2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 23-27, 2022, Kyoto, Japan

Towards Adaptive Continuous Control of Soft Robotic Manipulator using Reinforcement Learning

Yingqi Li, Xiaomei Wang, Ka-Wai Kwok, *Senior Member, IEEE*

***Abstract*—Although the soft robot is gaining considerable popularity in dexterous and safe manipulation, accurate motion control is still an open problem to be explored. Recent investigations suggest that reinforcement learning (RL) is a promising solution but lacks efficient adaptability for Sim2Real transfer or environment variations. In this paper, we present a deep deterministic policy gradient (DDPG)-based control system for the *continuous* task-space manipulation of soft robots. Domain randomization is adopted in simulation for fast control-policy initialization, while an offline retraining strategy is utilized to update the controller parameters for incremental learning. The experiments demonstrate that the proposed RL controller can track a moving target accurately (with RMSE of 1.26 mm), and accommodate to external varying load effectively (with ~30% RMSE reduction after retraining). Comparisons among the proposed RL controller and other supervised-learning-based controllers in handling additional tip load were also conducted. The results support that our RL method is appropriate for automatic learning such that there is no need of manual interference for data processing, particularly in cases with external disturbances and actuation redundancy.**

***Index Terms*—Deep reinforcement learning, Domain randomization, Incremental learning, Learning-based control, Soft robot control**

I. INTRODUCTION

OWING to the inherent compliance and softness, the soft robot has drawn increasing interest since its advent. There are emerging various applications of soft robots, such as the soft gripper suitable for delicate and deformable objects [1], flexible needle and endoscope in minimally invasive surgery (MIS)[2-4], and wearables or prostheses in rehabilitation [5]. However, due to the high nonlinearity of material and actuation, as well as the flexibility, the analytical model of soft robot cannot be fully defined, making its accurate control a challenging topic.

To tackle the problem of modeling, a variety of methods have been proposed to approximate the kinematics/dynamics model of continuum robots, generally divided into model-based and model-free solutions. Piecewise constant curvature (PCC) assumption has been widely applied in continuum robot modeling [6], and self-contained curvature sensors could assist in real-time closed-loop control[7]. Nonetheless, neglecting the dynamics behavior, the PCC-based controller can fail under external disturbance. Cosserat rod theory [8] takes forces into account, but the dynamics is modeled with a series of nonlinear partial differential equations (PDEs) which require large computational time. Finite element method (FEM) is also used to characterize the deformation [9], but the modeling is highly specific to the material and geometry, and also laborious. Avoiding model analysis of robots, model-free methods directly generate the mapping between actuation and task spaces. Since the mapping or control policy is trained using experimental sensory data, this method is also denominated as data-driven control. There have been various supervised learning algorithms successfully applied in continuum manipulator control, such as forward neural networks (FNNs) [10], locally weighted project regression (LWPR) [11] and locally Gaussian process regression (LGPR) [12]. Although these methods can train the global or local inverse kinematics (IK) mapping, which enable the soft manipulator to track goals and adapt to environment changes [13], they heavily rely on the quality of measured data. To avoid the problem of redundant mapping, careful elaboration on training data pre-processing is of importance, such as using sample pair filtering [12] and constrained optimization [14]. Usually, this requires manual adjustment on dataset distribution in a uniform pattern within the workspace, so as to alleviate the adverse data imbalance effect.

Recently, reinforcement learning (RL) has become an attractive strategy in soft continuum robot control [15]. Unlike supervised learning where the "answer key" (end-to-end data) is exploited to obtain the mapping of task-actuation space, RL could train a controller during automatic exploration capable of deciding the optimal action, given the state [16]. Referred to whether the model of robot motion is known, RL could be categorized into model-based and model-free algorithms. In soft robotics, it is challenging to obtain prior knowledge of robot structure or its interaction with the surrounding; therefore, model-free approaches are more prevalent in practice [13]. Q-learning, a popular model-free RL algorithm, including vanilla Q-learning [17], DQN [18] and DDQN [19], can store action-state values during training and also can pick the action with the highest value while testing. Such kind of methods has been validated in pneumatic artificial muscles (PAMs) to perform control tasks such as position tracking [20], path following with loads [21], and even complex interactions (e.g., drawer opening and handwheel rotating) [22]. However, the family of Q-learning can only estimate the state-action value in discrete space, which could sacrifice the control resolution and cannot be

This work was supported by the Research Grants Council (RGC) of Hong Kong (17209021, 17207020, 17205919), the Innovation and Technology Commission (ITC) (MRP/029/20X) of the HKSAR Government under the InnoHK initiative, Hong Kong, via Multi-scale Medical Robotics Center (MRC) Ltd. and Centre for Garment Production Limited.

Y. Li, X. Wang and K.W. Kwok are with Department of Mechanical Engineering, The University of Hong Kong, Hong Kong (corresponding author, Tel: +852-3917-2636; e-mail: kwokkw@hku.hk).

X. Wang is also with Multi-scale Medical Robotics Center (MRC) Limited.

2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 978-1-6654-7927-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/IROS47612.2022.9981335

applied in high-dimensional tasks. Deep deterministic policy gradient (DDPG), capable of continuous task-space control [23], was applied on pneumatic/hydraulic soft continuum arms [21, 24]. The "actor-critic" scheme in DDPG can approximate the environment and decision-maker, thus allowing the evaluation on continuous state-action pairs. Similar to supervised learning, model-free RL also relies on a large amount of training data. In practice, there are two main training approaches (i.e., acquiring the experience data) for RL-based controller: **i)** training in simulation [17, 18, 20], which is established using an ideal model such as PCC and Cosserat rod theory; **ii)** learning from scratch by real robots [25]. Simulation could generate abundant virtual data fast, therefore greatly shortening the training time, but it may suffer from a considerable Sim2Real gap [26]. Since model-free RL is sensitive to data, it is possible to break down when encountering a condition never seen in pre-training. The latter approach circumvents the gap, but obtains trial-and-error experience on a physical robot, particularly at the initial stage, thus could be low-efficient, costly, and even harmful to the operation environment or robot itself. Therefore, the combination of simulation and transfer adaptation is expected for efficient RL-centered soft robot control, which has not yet been exploited in the aforementioned works.

In this paper, we propose an adaptive RL-based control framework designed for soft continuum robots. The control policy is trained in a kinematics simulator first, then transferred to a real robot prototype with fine-tune technique to accommodate to motion disturbances. The core of adaptive learning can be described in two aspects. **i)** Domain randomization in simulation is to improve the controller's generalization ability, and **ii)** offline retraining is to facilitate continuous update of the controller. Consequently, precise tracking performance could be assured, and the advantage over supervised learning methods can then be demonstrated. The major work contributions can be concluded below:

1) Implementation of an RL-based controller in continuous task-space, which provides soft manipulators with *fast* initialization in simulation and *rapid* fine-tune of control policy in response to motion disturbance;
2) Adaptive learning strategy enhanced by incorporating both domain randomization and offline retraining, designed for soft robot controller to accommodate to Sim2Real deviation;
3) Experimental validations on accurate path-following ability, including comparison with supervised learning (FNNs, GPR) and simplified Jacobian model-based controllers.

II. Methodology

This section details the proposed RL control method for pneumatic-driven robots. Interaction of the robot with its surrounding is modelled by Markov decision process (MDP). The controller is initialized using DDPG algorithm in simulation. The strategy of domain randomization and offline retraining is employed to enhance the adaptability of RL-based controller.

A. Soft manipulator and task space definition

We use the soft manipulator in [27], accredited to its generic structure (**Fig. 1a**) for experimenting on its non-linear behavior. There distribute three cylindrical fluidic chambers spaced 120° apart from each other (**Fig. 1b**). Three independent actuators inflate these three chambers, enabling the robot to bend omnidirectionally under different combinations of inflation pressure. Locating the end-effector at the robot tip is convenient for observing the robot deformation effect due to any external interaction. A bellow sheath enclosing the robot body is not only to maintain the repeatability of robot operation by protecting the soft actuators from its material fatigue or damage, but also to reinforce the actuation withstanding high pneumatic pressure (max. 8 bar), thus exhibiting comprehensive non-linear behavior for validating learning-based controls for soft robots.

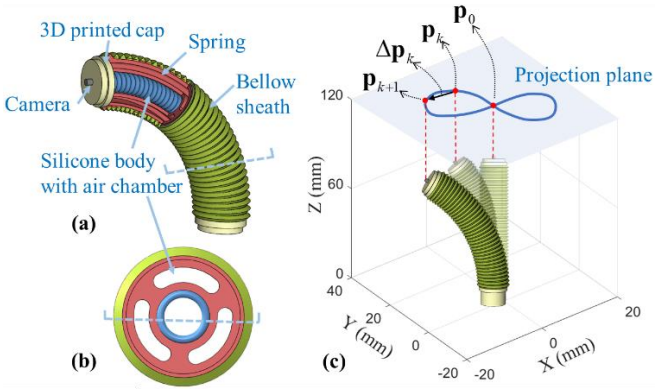

Fig. 1. (a) Structural diagram of the soft robot body; **(b)** Cross-section showing three air chambers for omni-directional robot bending; **(c)** Schematic of the robot path following task with the end-effector projection at time 0, k and k+1.

The actuator input at time step k (equilibrium) is denoted as $\mathbf{q}(k) \in \mathbb{R}^m$, where m expresses the dimension of actuation space (in this work, m = 3). The corresponding deformation/bending of robot could be represented as $[\mathbf{p}(k), \mathbf{n}(k)] \in \mathbb{R}^6$, where $\mathbf{p}(k)$ and $\mathbf{n}(k)$ are, respectively, the position and orientation of robot end-effector in world frame. The task space is defined in the projection plane normal to end-effector at initial state (**Fig. 1c**), with the incremental 2D displacement presented as $\Delta\mathbf{p}(k) = [\Delta p_x(k), \Delta p_y(k)]$. The objective of our study is to construct a controller, capable of instructing actuation command $\Delta\mathbf{q}(k)$ to accomplish the desired displacement $\Delta\mathbf{p}^*(k)$.

B. RL-based control

Here, we introduce the mathematical definition of parameters in RL and the training algorithm DDPG as follow.

i) *Markov Decision Process (MDP)*: *Markov Decision Process (MDP)* is used to characterize the interaction between the robot and the environment in our RL framework, involving state s, action a, state transition probability $\mathrm{P}(s' \mid a, s)$ and reward r, where s' is the next state after action execution. In terms of robot control, the process could be explained such that the controller perceives state information s, then decides the action a. Afterwards, the

environment feeds back the next state s' and reward r. Generally, reward r, state s and action a are defined empirically upon the required robot task. Transition probability $\mathrm{P}(s'|a,s)$ is counted on the environment. In other words, RL with specific definition of parameters would also be applicable to various robots and tasks.

Considering that the fluid-driven soft continuum robot is deformed by chamber inflation from m independent pneumatic/hydraulic actuators (here, m = 3), we define the action a as $a=\Delta\boldsymbol{\alpha}=[\Delta\alpha_1,\Delta\alpha_2,\Delta\alpha_3]$, where $\Delta\alpha_i\,(i=1,2,3)$ is the length change of each chamber. To stimulate the robot bending towards the target, the state is defined as $s=[\mathrm{p}_x,\mathrm{p}_y,\Delta\mathrm{p}_x,\Delta\mathrm{p}_y,F]$, where $\Delta\mathrm{p}_x$ and $\Delta\mathrm{p}_y$ are the errors between end-effector and target, respectively, in x and y directions. Binary variable $F\in\{0,1\}$ indicates whether the manipulator has reached the goal such that F = 1. The reward function is designed as:

$$r=\begin{cases}-b+err_{k-1}-err_k,\ err_k>\varepsilon\\ -b+err_{k-1}-err_k+c,\ err_k\le\varepsilon\end{cases}\tag{1}$$

where b is a penalty term on each transition, thus motivating the manipulator to reach the goal in as few steps as possible. And *err* is the distance error between robot position to target. The controller would be rewarded with a large value c when the manipulator touched the target within a threshold ε. The transition probability $\mathrm{P}(s'|a,s)$, namely the kinematics model of the soft robot is unknown.

With the defined parameters in MDP, the motion of soft robot guided by policy π_θ could be mathematically depicted as a procedure $\tau=(s_0,a_0,s_1,r_1,a_1,\cdots,s_n,r_n,a_n)$, where n is the motion step. The objective of RL is to maximize the cumulative reward during the transition, which is stated as:

$$\max\mathbb{E}_{\tau\sim\mathrm{P},\pi}\left[\sum_{t=0}^{\infty}\gamma^t r_{t+1}\right]\tag{2}$$

where $\gamma\in(0,1)$ is a discount factor. And the control policy π_θ^*, which is able to direct the soft robot to obtain the maximum reward, is defined as the optimal controller.

ii) *Deep Deterministic Policy Gradient (DDPG)*: is a model-free RL algorithm designed for continuous control, integrating policy gradient and value-based method. The state-action value, also named as Q-value, is used to assess the quality of action in the algorithm. The framework consists of an "actor" network $\mu(s|\theta^\mu)$ and a "critic" network $\omega(s,a|\theta^\omega)$, where "actor" receives the observation, then outputs action, while "critic" evaluates the action under the specific state. To avoid the problem of bootstrapping, new definitions of target "actor" $\mu'(s|\theta^{\mu'})$ and target "critic" $\omega'(s,a|\theta^{\omega'})$ are utilized. The update goal of "critic" is to minimize the approximation loss on state-action evaluation, i.e., to approach the real environment, which follows:

$$\theta^\omega\leftarrow\theta^\omega-\alpha L\tag{3}$$

$$L=\frac{1}{N}\sum_i\left(r_i+\gamma\omega'\left(s_{i+1},\mu'\left(s_{i+1}|\theta^{\mu'}\right)|\theta^{\omega'}\right)-\omega\left(s_i,a_i|\theta^\omega\right)\right)^2\tag{4}$$

where α is the learning rate for "actor" update, and N is the minibatch size. The objective of "actor" is to maximize the Q-value obtained from "critic", and its update follows:

$$\theta^\mu\leftarrow\theta^\mu+\beta\frac{1}{N}\sum_i\nabla_a\omega\left(s,a|\theta^\omega\right)|_{s=s_i,a=\mu(s_i)}\ \nabla_{\theta^\mu}\mu\left(s|\theta^\mu\right)|_{s_i}\tag{5}$$

where β is the learning rate for "critic" update. The update of target networks follows:

$$\theta^{\omega'}\leftarrow\eta\theta^\omega+(1-\eta)\theta^{\omega'}\tag{6}$$

$$\theta^{\mu'}\leftarrow\eta\theta^\mu+(1-\eta)\theta^{\mu'}\tag{7}$$

where η is the learning rate for target models.

C. Kinematics simulation with domain randomization

To obtain a rough kinematics control policy in a short time, a simulation environment is developed using constant curvature (CC) model [28]. The "interaction" in this study is defined as every time the robot acts, the virtual environment would feed back its state information and the reward. In other words, the transition probability $\mathrm{P}(s'|a,s)$ is given by CC assumption. The CC model theoretically defines the configuration of an ideal single-section robot. However, due to the deviations not considered in the assumption such as fabrication bias and material nonlinearity, it is almost impossible for the real robot to be deformed as same as the ideal model. Even though performing satisfactorily in simulation, the controller may result in a diverged performance break down in the real world. Domain randomization can improve the generalization ability of trained models by adding variations into simulation [29]. Here, observation and physical parameter randomizations are deployed: Gaussian noise is added to end-effector position **p**, to mimic the observation noise of sensors. Noise sampled from a continuous uniform distribution is added to the chamber length $\boldsymbol{\alpha}$, to simulate the actual deformation of real robots.

In the first step of training, a large number of goals would be generated randomly within the workspace, namely the goal pool. At the start of each episode, a new goal would be picked from the pool, and the policy would "learn" to approach the goal within certain steps. Once the target is reached or the accumulative number of movements exceeds the maximum, the episode is regarded as terminated. The procedure of pre-training strategy is summarized in **Algorithm 1**. In contrast to the training with a specified target path, such a method with random goals enables the controller to be applied in different position control tasks such as point targeting and various path following.

Algorithm 1: Controller Pre-training in Simulation using DDPG

Input: training episode M
maximum movement times T
Randomly generate virtual goal pool $\mathcal{G}$
for episode=1, M **do**
  Reset the goal by random picking from $\mathcal{G}$
  for step=1, T **do**
    Train using DDPG
    if goal is reached **then**
      break
  end for
end for

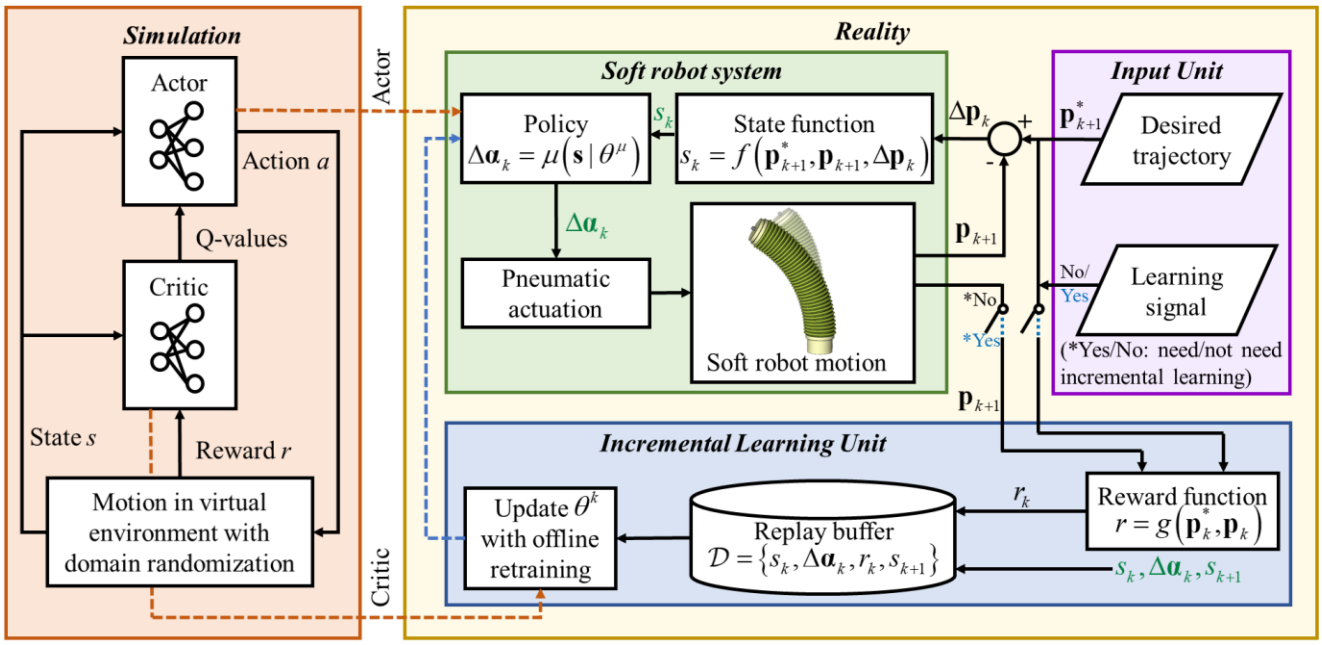

Fig. 2. Proposed RL-based robot control architecture. The controller, consisting of "actor" and "critic" models, is trained first in a virtual environment (Left), then transferred onto a real robot (Right). The "actor" model acts as controller generating the action command. The "critic" model evaluates the "actor", thus fine-tuning the "actor". The incremental learning unit is activated when encountering environment variations, providing "actor" and "critic" models with parameters update.

D. Incremental learning with offline retraining

In addition to the effort in policy pre-training, i.e., introducing random noises into the simulation, adaptive learning in this research also leverages the technique of fine-tuning during implementation on real robots. Offline retraining [30] is an encouraging strategy, enabling robots to adapt to new variations rapidly. The workflow of this method can be described as: First, collecting practical experience on real robots using pre-trained controller; Next, updating the pre-trained controller using the same algorithm as simulation, with a decreased learning rate; Then, collecting data using retrained policy for next update. This sampling-updating loop would not be ended until the performance is satisfying. The procedure of offline retraining is summarized in **Algorithm 2**. The control flowchart, **Fig. 2**, depicts the key procedure in the proposed control frame, including simulation, implementation on real robot and incremental learning.

Algorithm 2: Offline Retraining on Real Robot

Input: Pre-trained control policy θ_{init}
Empty execution control policy θ_{exu}
Retraining episode M
Empty replay buffer $\mathcal{D}$
Output: Fine-tuned control policy θ^*

```
Initialize θ_exu = θ_init
for episode=1, M do
    Empty D
    Collect experience data using θ_exu and store into D
    Offline update θ_exu
end for
Return θ_exu
```

III. Experiments, Results and Discussion

The presented RL-based controller is applied in various path following tasks with/without disturbance, showing the performance of how the robot could track the moving-target along the path. Comparison with simple Jacobian model-based and typical supervised learning-based controllers is also included.

A. Simulation

Pre-training of control policy has to be conducted on simulation, in prior to its implementation on the robot platform. Four FNNs work individually as (target) "actors" and (target) "critics", each of which is constructed with five hidden layers containing 128 neurons. The training was repeated for 5,000 episodes, and the tolerated maximum step in each episode was set to 200. Reaching threshold $\varepsilon = 0.5$ mm was chosen as 2.8% of the soft continuum diameter. The replay buffer acts as a memory pool, the size of which was set as 10,000. The training was implemented on computing workstation, AMD 3990X with graphic processing units (GPUs), NVIDIA RTX 3090. The trained results converged after ~500 episodes in ~3 hours. The trained RL-based controller was verified on an "∞"-shape path following task in virtual. The resultant root mean square error (RMSE) was 1.32 mm. The test time of each step was 1 ms, implying the controller can respond promptly once the state was given.

B. Experimental setup

A compact soft continuum manipulator (**Fig. 1a**, Ø18×70-mm) was made from a molded silicone (Dragon Skin20, Smooth-on Inc.). Its cylindrical surface was covered by a reinforced layer of bellow sheath. A closed-coil spring (Ø7 mm) was integrated as a robot skeleton, which would empower the robot dynamics [27]. Maximum robot bending

angle was confined below 90° (corresponding actuation pressure: ~2.5 bar), projecting the 2D plane within 35×35 mm. An endoscopic camera (Ø3.5 mm, Shanghai Yanshun Co., Ltd) was mounted on the robot tip cap for the purpose of visual exploration. An electromagnetic (EM) tracking system (NDI Medical Aurora) was used to provide position feedback to the presented controller. Two EM coils were attached onto the robot tip. All the processes, including sensing and control computation, were implemented on Python, so as to avoid communication time among different compliers. Running time intervals for sensing and control were minimized to 24 ms and 1 ms, respectively.

The control policy will determine the *action* for each step in the form of robot chamber length change $\Delta\boldsymbol{\alpha}$. However, the actuator (i.e., stepper motors in our setup, connected to cylinders and then soft robot) can only be commanded by motor position $\mathbf{q}$, which adjusts the chamber length by pushing cylinders and varying the pressure inside chambers. Therefore, the relation between chamber length change $\Delta\boldsymbol{\alpha}$ and motor position $\mathbf{q}$ needs to be established. We approximated it by means of fitting several sample pairs ($\Delta\boldsymbol{\alpha}$ to $\mathbf{q}$) using cubic spline data interpolation. The fitting result shows that their relation can be represented by a nonlinear curve upward. The larger the motor position, the smaller the slope.

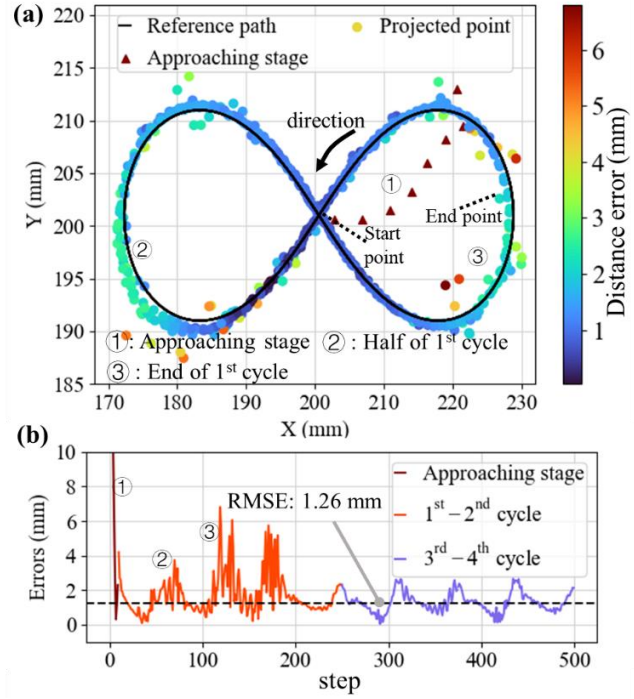

Fig. 3. Path "∞" following performance of the pre-trained controller. **(a)** Robot tracking started from the center [200, 200]. Warmer the color, bigger the deviation from the path; **(b)** Tracking error in two cycles of "∞". The RMSE is 1.26 mm after the approaching stage.

C. Path following using pre-trained controller

To evaluate the training effectiveness, the pre-trained control policy was implemented on the soft manipulator in path following task. In this task, the robot is aimed at tracking a point moving along a 2-D reference "∞" path within 56×20 mm (**Fig. 3a**). The tracking speed is expected to be constant. Four cycles (~500 time steps in total) were carried out in order to observe the resultant robot motion behaviors varied with the chamber pressures. The robot started from the center of projection plane [200, 200] and tracked on the moving goal shifting in each time step. Note that the actual trajectory of robot's end-effector takes place in a spatial cambered surface, and the 2D projection on the *x*-*y* plane is regarded as the trajectory to be evaluated. Euclidean distance error at each time step is the distance measured from the end-effector position to the moving goal along the path. RMSE is used to indicate the overall accuracy for a period of tracking.

Fig. 3a and **3b** indicate, respectively, the tracking footprint and error. It took ~10 time steps, namely the approaching stage, for the robot to catch up the moving goal originated from the start point. Afterwards, the robot could follow the target closely. The RMSE counted, excluding the approaching stage, is 1.26-mm, which is ~6.8% of the robot tip diameter. It can be seen that the tracking performance in different cycles is not identical at the same place. For instance, at the 1st and 2nd cycle, the robot tip would oscillate with a maximum distance error of 6 mm (time step: 125~180). But when operating at the 3rd cycle, it would move smoothly with the maximum error of 3 mm (time step: 250~305). It is generally caused by the nonlinear correlation between the chamber length and pressure. At the beginning, a large chamber pressure variation is needed but results in only a little change of the chamber length. Thus, the swift pressure variation might induce the robot tip shaking back and forth. This experiment demonstrates that the initial controller only trained in virtual environment can accomplish high-accuracy path following task in the reality. Furthermore, such controller performance also indicates that the strategy of domain randomization can facilitate to bridge the Sim2Real gap.

D. Path following under varying load

Varying load on the robot tip was applied so as to validate if the controller learning could enable the robot control adapted with such varying dynamic disturbance. First, the controller was pre-trained in simulation only taking account of soft robot kinematics. Then, while testing in the reality, a balloon cap that can hold water was mounted on the robot tip. As displayed in **Fig. 4a**, the load could be variable by injecting water into the balloon. The balloon setup weighing 4 g could contain up to 15-g water. Namely, the controller needs to deal with a varying tip payload ranging from 4 g to 19 g through its incremental learning. The robot was commanded to follow the path in two cycles, in each of which, water was injected and extracted in a regular rhythm and volume (**Fig. 4b**). Every injection/extraction would complete in few seconds. Once the 1st cycle is completed, the controller would update for 50 epochs offline using the collected data. The update took <5 s only. The tracking data in 1st and 2nd cycles were used to display the performance of pre-trained and updated controllers, respectively.

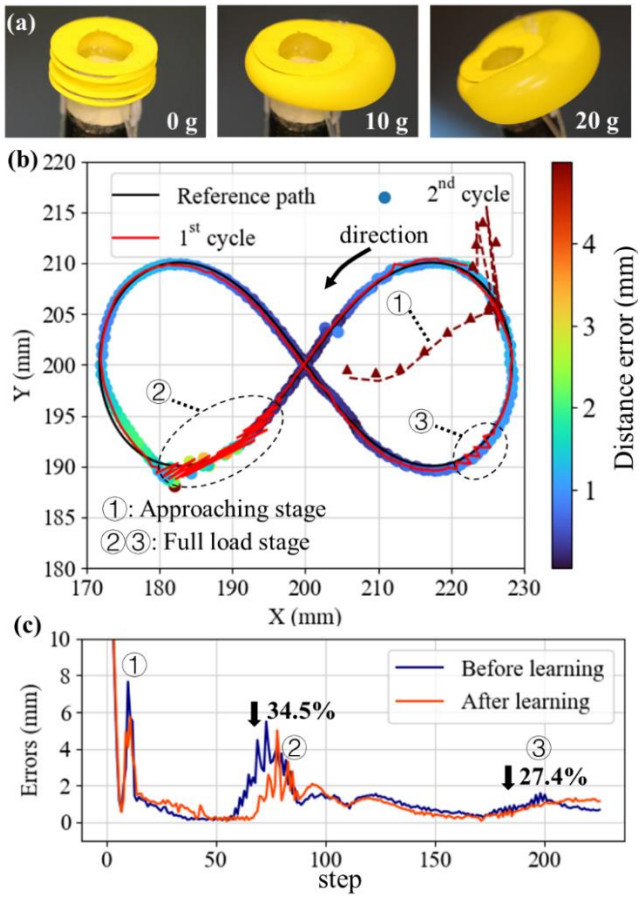

Fig. 4. Two-cycles path following of the fine-tuned controller under varying load. A 4 g balloon cap at the robot tip can be injected by water, changing the payload from 4 to 19 g. **(a)** Water injection at three stages: none, half load and full load; **(b)** Controllers got retrained in between the first and second cycles; **(c)** Corresponding errors of pre-trained and fine-tuned controllers.

The tracking trajectory and error are plotted in **Fig. 4b** and **4c**, respectively. During the two stages with full load (marked as ② and ③) in the 1st cycle, the *action* bounced between the upper and lower limits, causing the robot tip oscillated along the reference path with a maximum deviation of 5.48 mm. At the same stages in the 2nd cycle, the oscillation was diminished or eliminated obviously, within a reduced deviation of 5.0 mm. For the 1st full-load stage (time step: 62~90), the RMSE decreased from 2.06 mm to 1.35 mm (34.5% reduction); for the 2nd full-load stage (time step: 192~200), the RMSE declines from 0.84 mm to 0.61 mm (27.4% reduction). It can be summarized that our controller is able to keep tracking with less oscillation after offline retraining. Such controller updates can act every time when a cycle/period is finished, being helpful for accuracy enhancement in position control tasks featured with repeated workspace. However, the controller after incremental learning can only accommodate to new conditions within limited range, instead of eliminating oscillation thoroughly. Due to the nonlinearity of robot actuation, the robot tends to execute large action when the chamber pressure is close to its initial state (i.e., minimal value). When tracking without external disturbances, the large action would only cause little oscillation, whereas the full load could aggravate the oscillation. The RL-learned control mechanism gradually approaches the actual robot features (including fabrication bias of robot, and impact of the surrounding) by means of incremental learning.

E. Comparison among: RL-, model- and supervised learning-based controllers

Experimental comparison among our RL-controller, model-based and two kinds of supervised learning-based controllers is carried out. It aims to explore an appropriate work condition for RL-based controller, also to optimize its strength in soft robot control. The model-based controller was constructed using CC assumption (aka. Jacobian model- or model-based controller in this study). It can assure a relative stable performance when the robot motion keeps in the quasi-static state. Both NN- and GPR-based controllers are in category of supervised learning methods. A 3-hidden-layer FNN model was used in the NN-based controller. Target position is taken as input and the chamber length of soft robot as output, and input-output pairs were sampled in virtual environment. In fairness to the comparison, no post-processing was followed. In GPR algorithm, data clustering was also disregarded. The training was run with the tool of scikit-learn in Python. In prior to its validation on the real robot prototype, these three controllers were tested in the circle path (30×30 mm) following task in simulation. All the four controllers can track/follow the moving goal smoothly, with the RMSE of 1.44 mm, 0.84 mm and 1.07 mm, respectively, using model-, NN- and GPR-based controllers. The promising performance in simulation indicates the training of NN- and GPR-based control policy is correct. In regard to the experiment on real robot, the control performance was evaluated by the circle path following tasks with and without a constant load (25.3 g) applied. There were 210 time steps in one cycle, and each test includes two cycles for observing whether the deviation would be reduced in the 2nd cycle.

Fig. 5. shows the tracking performance with and without the load. The RL- and model-based controllers made the tracking started from the center, then catching up the goal within 10 steps only. Similar to previous experiments (in **Section III. C** and **D**), there appear oscillations in actual trajectories. The slight deviation (<0.85 mm) of the RL-based controller indicates its fast convergence ability (**Fig. 5a**). However, the robot using model-based controller still moved with obvious vibration, resulting in a zig-zag trajectory (**Fig. 5b**). Both the NN- and GPR-based controllers, despite their promising tracking results in simulation, cannot catch up the goal point precisely (**Fig. 5c**), with RMSE of 10.78 mm and 11.12 mm, respectively. Due to their unacceptable performances (>10 mm), we skip them and just do the comparison between RL-based and model-based controllers in our further experiments.

Fig. 5a and **5b** plot the tracking performance of RL- and model-based controllers with and without the load. Although the extra load induced larger amplitude of oscillations, the robot using RL-based controller was still capable to follow the circle with the RMSE of 1.70 mm only. Similarly in using

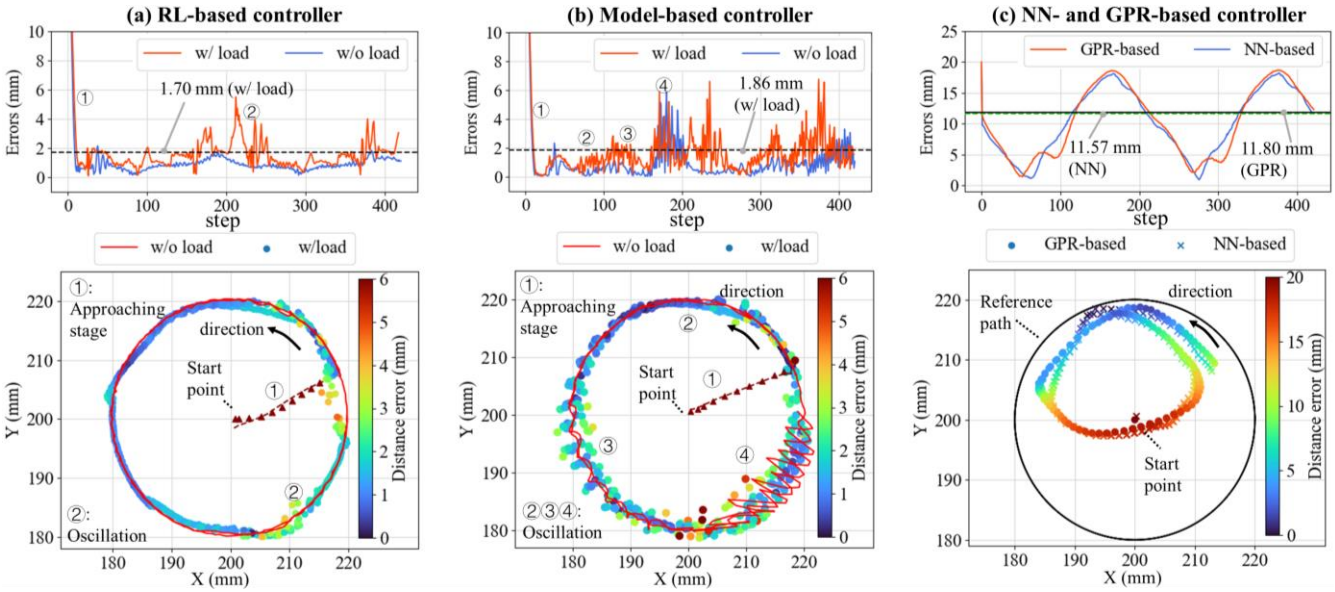

Fig. 5 Circle path following in two cycles (21 s per cycle). Besides free-space tests without load, a set of tests with an additional constant load (25.3 g) was conducted when using the RL-based and Jacobian model-based controllers. Tracking errors (upper) and trajectories (lower) of **(a)** RL-based controller, **(b)** Jacobian model-based controller, **(c)** NN- and GPR-based controllers. The tracking RMSE of four controllers are annotated in tracking error plots.

model-based controller, the load also intensified the oscillation/shake along the path, the overall RMSE of which was 1.86 mm. It can be observed that the trajectory in **Fig. 5a** is smoother and more stable than the one in **Fig. 5b** (marked as ② ③ ④). To further compare the two controllers' performance in regard of oscillation, the RMSE of a particular period (time step: 320~420, typical oscillation period, marked as② in **Fig. 5a** and ④ in **Fig. 5b**) was calculated. Whether with or without the constant load, the RL-based controller could promote the tracking performance (~20% reduction in RMSE, 18.5%~63.6% reduction in maximum deviation error) compared with the model-based controller. Detailed experimental results are summarized in **Table 1**.

TABLE 1. RMSE AND MAXIMUM DISTANCE ERROR COMPARISON WITH OTHER THREE TYPES OF CONTROLLERS IN CIRCLE PATH FOLLOWING TASK.

	w/o load		w/ load	
	RMSE*	Max. err.	RMSE*	Max. err.
Model-based	1.35	5.88	2.38	6.75
NN-based	11.57	18.23	N/A	
GPR-based	11.80	18.70		
Our RL-based	1.09	2.14	1.85	5.5
Improv.**	**19.3%**	**63.6%**	**22.3%**	**18.5%**

* The RMSE during time step 320~420 is presented for comparison.
** "Improv." denotes the improvement of RL- over the model-based controllers.

In addition to accuracy validation tests, we also applied the soft manipulator with RL-based controller in a vision task. The soft robot was fixed downward, with the tip camera viewing a printed badge picture (**Fig. 6a**). Its camera field of view (FoV: 120°) would visualize only part of the badge at one shot. The soft robot was commanded to follow a circle path (10×10 mm) and save an image every step, in order to capture and reconstruct the full view of the badge. A series of images were mosaicked (**Fig. 6b**) using the Computer Vision Toolbox in MATLAB. The reconstruction of 2D scene using mosaicking is more or less conform to the real scenario, which indicates that the proposed controller enables soft robot to move swimmingly.

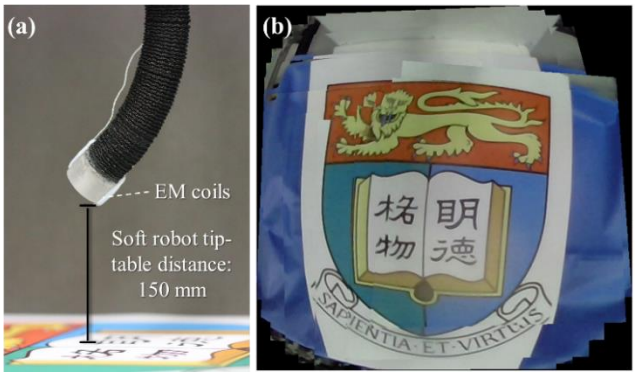

Fig. 6. (a) Camera at the robot tip viewing downwards a printed HKU badge. Two electromagnetic (EM) positional tracking coils were attached at the tip, providing the controller with sensing feedback; **(b)** Mosaicked image result.

IV. CONCLUSION

This paper proposes an RL-based adaptive control framework, which enables the fluid-driven soft robot well-controlled in continuous task-space. Provided with sufficient exploration in simulation, where domain randomization is applied, our RL controller could "learn" to track targets precisely even encountering large Sim2Real gap. The offline retraining scheme encourages the coarse control policy to fine-tune itself towards specific task, thus adapting to the environment variations.

In view of the strong durability, the soft robot consisting of bellow sheath and spring is chosen for demonstration. Our roughly pre-trained RL controller is first verified on the soft robotic manipulator for path following. The acceptable tracking accuracy (1.26 mm in RMSE) was obtained. It supports that the technique of domain randomization used in simulation effectively can improve the generalization ability of control policy. In the path following task varied with payload (4~19 g), the controller could weaken the oscillation at RMSE reduction of ~30%, while maintaining the soft robot's movement in a stable and smooth state. In comparison

with model-based, NN-based and GPR-based controllers, our RL controller could reduce the tracking error by >20%, proving its eligibility for redundant control and biased dataset.

In future work, we will further explore the adaptive learning ability of RL-based control policy. In terms of simulation-based model training, more effective Sim2Real transfer strategies will be studied, such as domain adaptation [26], which would improve the controller's adaptability even with more sudden changing interaction. Additionally, we will employ computational mechanics in accounts for more realistic hyper-elastic deformation or dynamics, e.g., FEM and Cosserat rod theory. In regards of implementation on real robots, we intend to investigate incremental learning algorithms such as policy relaxation and importance weighting [31], which would speed up the control adaptation with varying disturbance by the robot surroundings. Moreover, advanced self-contained curvature sensors such as fiber Bragg gratings (FBGs), could be applied to offer precise chamber length feedbacks as well as end-effector pose, since they are capable for configuration-related measurement for soft robots. The soft continuum robot with outstanding adaptive ability would be considered to perform more intelligent manipulation tasks in unstructured environments, such as catching a randomly-walking object. The strong adaptability would also enable our RL-based control framework to generalize on soft robots with various morphology or multiple end-effectors, even micro-scale medical robots such as [32]. After setting specific kinematics or FEA model for the robot to be used, as well as the desired task space, automatic learning can be conducted to generate a corresponding RL controller.

REFERENCES

[1] Z. Gong *et al.*, "A soft manipulator for efficient delicate grasping in shallow water: Modeling, control, and real-world experiments," *The International Journal of Robotics Research,* vol. 40, no. 1, pp. 449-469, 2021.

[2] G. Fang *et al.*, "Soft robotic manipulator for intraoperative MRI-guided transoral laser microsurgery," *Science Robotics,* vol. 6, no. 57, p. eabg5575, 2021.

[3] K.-W. Kwok, H. Wurdemann, A. Arezzo, A. Menciassi, and K. Althoefer, "Soft Robot-Assisted Minimally Invasive Surgery and Interventions: Advances and Outlook," *Proceedings of the IEEE,* 2022.

[4] V. Vitiello, K.-W. Kwok, and G.-Z. Yang, "Introduction to robot-assisted minimally invasive surgery (MIS)," in *Medical Robotics*: Elsevier, 2012, pp. 1-P1.

[5] M. Zhu, S. Biswas, S. I. Dinulescu, N. Kastor, E. W. Hawkes, and Y. Visell, "Soft, Wearable Robotics and Haptics: Technologies, Trends, and Emerging Applications," *Proceedings of the IEEE,* 2022.

[6] 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.

[7] Z. Dong *et al.*, "Shape tracking and feedback control of cardiac catheter using MRI-guided robotic platform—Validation with pulmonary vein isolation simulator in MRI," *IEEE Transactions on Robotics,* 2022.

[8] A. A. Alqumsan, S. Khoo, and M. Norton, "Robust control of continuum robots using Cosserat rod theory," *Mechanism and Machine Theory,* vol. 131, pp. 48-61, 2019.

[9] T. M. 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, no. 3, pp. 348-364, 2018.

[10] M. Giorelli, F. Renda, G. Ferri, and C. Laschi, "A feed-forward neural network learning the inverse kinetics of a soft cable-driven manipulator moving in three-dimensional space," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013: IEEE, pp. 5033-5039.

[11] K.-H. Lee *et al.*, "Nonparametric online learning control for soft continuum robot: An enabling technique for effective endoscopic navigation," *Soft robotics,* vol. 4, no. 4, pp. 324-337, 2017.

[12] G. Fang *et al.*, "Vision-based online learning kinematic control for soft robots using local gaussian process regression," *IEEE Robotics and Automation Letters,* vol. 4, no. 2, pp. 1194-1201, 2019.

[13] X. Wang, Y. Li, and K.-W. Kwok, "A Survey for Machine Learning-Based Control of Continuum Robots," *Frontiers in Robotics and AI,* p. 280, 2021.

[14] J. D. Ho *et al.*, "Localized online learning-based control of a soft redundant manipulator under variable loading," *Advanced Robotics,* vol. 32, no. 21, pp. 1168-1183, 2018.

[15] S. Bhagat, H. Banerjee, Z. T. Ho Tse, and H. Ren, "Deep reinforcement learning for soft, flexible robots: Brief review with impending challenges," *Robotics,* vol. 8, no. 1, p. 4, 2019.

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

[17] X. You *et al.*, "Model-free control for soft manipulators based on reinforcement learning," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017: IEEE, pp. 2909-2915.

[18] Q. Wu *et al.*, "Position control of cable-driven robotic soft arm based on deep reinforcement learning," *Information,* vol. 11, no. 6, p. 310, 2020.

[19] H. You, E. Bae, Y. Moon, J. Kweon, and J. Choi, "Automatic control of cardiac ablation catheter with deep reinforcement learning method," *Journal of Mechanical Science and Technology,* vol. 33, no. 11, pp. 5415-5423, 2019.

[20] 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: IEEE, pp. 5133-5139.

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

[22] H. Jiang *et al.*, "Hierarchical control of soft manipulators towards unstructured interactions," *The International Journal of Robotics Research,* vol. 40, no. 1, pp. 411-434, 2021.

[23] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971,* 2015.

[24] Y. Zhang, T. Wang, N. Tan, and S. Zhu, "Open-Loop Motion Control of a Hydraulic Soft Robotic Arm Using Deep Reinforcement Learning," in *International Conference on Intelligent Robotics and Applications*, 2021: Springer, pp. 302-312.

[25] 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: IEEE, pp. 141-148.

[26] W. Zhao, J. P. Queralta, and T. Westerlund, "Sim-to-real transfer in deep reinforcement learning for robotics: a survey," in *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2020: IEEE, pp. 737-744.

[27] H.-C. Fu *et al.*, "Interfacing soft and hard: a spring reinforced actuator," *Soft robotics,* vol. 7, no. 1, pp. 44-58, 2020.

[28] X. Wang *et al.*, "Learning-based Visual-Strain Fusion for Eye-in-hand Soft Robot Pose Estimation and Control," *IEEE Transactions on Robotics,* (under review), 2022.

[29] O. M. Andrychowicz *et al.*, "Learning dexterous in-hand manipulation," *The International Journal of Robotics Research,* vol. 39, no. 1, pp. 3-20, 2020.

[30] R. Julian, B. Swanson, G. S. Sukhatme, S. Levine, C. Finn, and K. Hausman, "Never stop learning: The effectiveness of fine-tuning in robotic reinforcement learning," *arXiv preprint arXiv:2004.10190,* 2020.

[31] Z. Wang, H.-X. Li, and C. Chen, "Incremental reinforcement learning in continuous spaces via policy relaxation and importance weighting," *IEEE Transactions on Neural Networks and Learning Systems,* vol. 31, no. 6, pp. 1870-1883, 2019.

[32] L. Lindenroth, S. Bano, A. Stilli, J. G. Manjaly, and D. Stoyanov, "A fluidic soft robot for needle guidance and motion compensation in intratympanic steroid injections," *IEEE Robotics and Automation Letters,* vol. 6, no. 2, pp. 871-878, 2021.