Periodic movement learning in a soft-robotic arm

Paris Oikonomou, Mehdi Khamassi, Costas S Tzafestas

Periodic movement learning in a soft-robotic arm*

Paris Oikonomou[1], Mehdi Khamassi[1,2] and Costas S. Tzafestas[1]

Abstract— **In this paper we introduce a novel technique that aims to dynamically control a modular bio-inspired soft-robotic arm in order to perform cyclic rhythmic patterns. Oscillatory signals are produced at the actuator's level by a central pattern generator (CPG), resulting in the generation of a periodic motion by the robot's end-effector. The proposed controller is based on a model-free neurodynamic scheme and is assigned with the task of training a policy that computes the parameters of the CPG model which generates a trajectory with desired features. The proposed methodology is first evaluated with a simulation model, which successfully reproduces the trained targets. Then experiments are also conducted using the real robot. Both procedures validate the efficiency of the learning architecture to successfully complete these tasks.**

Index Terms— **Reinforcement learning, Central pattern generators, Soft Robotics, Rhythmic movements**

I. Introduction

OVER the last years, continuum bio-inspired manipulators based on soft robotic technologies have drawn increasing attention of the robotics community [1]–[3]. These soft robots open many novel research and application routes, and present specific advantages related to their natural compliance and biomorphic control properties. However, despite such beneficial properties, all these systems present specific problems and limitations that still prevent their widespread use in many application domains. These are mostly related to the difficulty regarding the design of efficient control schemes that can provide accuracy and robustness in dynamic tasks. This is due to the non-linearity and the high complexity introduced by the multiple DoFs, preventing the computation of a mathematical model that simulate the system dynamics, or the design of a controller based on classical methods. Nevertheless, overcoming these limitations would permit to benefit from the flexibility of soft robots when used in environments where the target is unreachable by rigid arms, as well as benefiting from their inherent safety, which may be critical during interaction for specific applications such as in the medical domain.

Recent research work specifically addressed the problems related to the use of continuum robots to perform dynamic control through model-based approaches. For instance, [1] introduces a dynamic model based on a nonlinear model-based strategy for curvature space control that is applied to robots with certain properties, such as extension, contraction, or omnidirectional bending. A similar approach is presented in [2], where model-based controllers are developed based on suitable models using a combination of feedforward control and decoupled PD-controllers, applied to a pneumatically actuated manipulator with multiple actuators.

A different approach is proposed in [3], based on the design of open-loop predictive controllers directly from the actuation to the task space. In particular, a machine learning-based approach is used in order to learn the dynamic models, while a trajectory optimization method is performed to compute the control policy. Based on a different set of techniques, the work presented in [4] applies novel spatial dynamics to variable length multisection continuum arms which, as opposed to other continuum robots, are actuated by multiple variable length actuators, assuming circular arc deformation of continuum sections without torsion. A relevant approach is presented in [5] where the authors use a feedforward neural network component to compensate for dynamic uncertainties.

In this paper, we present an architecture that aims to dynamically control a modular bio-inspired soft-robotic arm. Our approach uses a suitably designed controller based on a neurodynamic adaptive control scheme that provides the soft robot's end-effector with the ability to learn a policy that generates cyclic periodic trajectories of desired features, defined by the user. In particular, our approach combines: (a) a Reinforcement Learning (RL) algorithm that handles continuous states and actions, while providing adaptation and robustness to variations in the environment as well as in the dynamics of the robotic system during operation, and (b) a Central Pattern Generator (CPG) model that is applied to the motors, resulting in the generation of oscillatory motion of appropriate features, which are provided by the RL stage. The efficiency of the proposed architecture is initially evaluated on a simulation model, in terms of its convergence, repeatability, and generalizability, while an experimental setup with the real robot validates the proposed methodology. The following section summarizes related work and highlights the main novelty and contribution of the proposed approach.

II. Related Work

CPGs are neural circuits that are capable of producing rhythmic coordinated patterns of high-dimensional rhythmic output signals [6]. CPGs' periodic movements are usually operated without receiving any rhythmic input from sensory feedback, but only simple low-dimensional input signals. CPGs have been found in animals' nervous system, underlying many rhythmic activities like chewing or breathing, but more importantly they contribute to their locomotor

* This work has been partially funded by the EU project I-SUPPORT (grant agreement no. 643666) and by the French Centre National de la Recherche Scientifique (CNRS)'s PICS international scheme no. 279521.

[1]All authors are with the School of Electrical and Computer Engineering, National Technical University of Athens, Greece.

[2]Mehdi Khamassi is also with Sorbonne Université, CNRS, Institute of Intelligent Systems and Robotics, Paris, France.

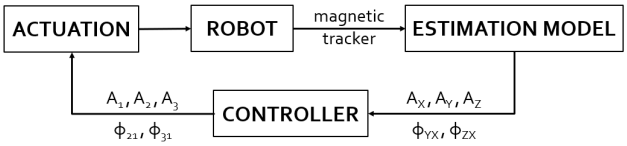

Fig. 1: Block Diagram

neural circuits. Their interesting properties, like the ability to modulate the features of the produced rhythmic output (e.g. frequency) by using simple control signals, has represented a major potential source of scientific research. This led to the design of neurobiological mathematical models that approximate their functionality. These models have been extensively analysed and used with success in the field of Robotics, focusing especially on the design of locomotion controllers in robots, like in [7] where the authors created artificial CPGs in order to simulate swimming movements in a lamprey-like substrate. In more recent work, CPG controllers have been used in combination with various RL schemes to enhance their adaptive properties. This is the case in [8] where a natural CPG-Actor-Critic architecture was implemented on a 3D-simulated humanoid in a relatively high-dimensional state space, or in [9] where an actor-critic architecture permitted to find a good reshaping function, when it was used in combination with a control scheme composed of CPGs and Dynamic Motor Primitives. However, to our knowledge none of these CPG implementations have been applied in combination with an RL controller to control a soft robotic arm, in a scheme that exploits the property of the CPG to generate complex signals out of simple commands, and the ability of a soft-robot to produce complex motions due to its dexterity.

Other work addressed closed-loop dynamic control with application to soft-robotic manipulators. In [10], a closed-loop predictive controller was implemented with a model-based policy learning algorithm and trained using trajectory optimization and supervised learning. Such control schemes, however, are not suitable for the execution of dynamic motion patters, such as the cyclic trajectories considered in this paper. In contrast, the contribution of our work consists in exploiting the simplicity of a CPG model to generate complex rhythmic patterns. The RL-based controller proposed in this paper is designed to learn simple signals that control the parameters of the CPG implementation applied to the motors, and hence producing trajectories of desired features.

III. Learning Architecture

This section describes the mathematical formulation underlying the proposed methodology. Starting from the tools that are used to test the architecture (Fig. 1), it continues to the presentation of the CPG model applied to the actuators, and concludes with the description of the RL-based controller.

A. Preliminaries

The goal of the present work being to enable a soft robotic manipulator to learn cyclic movements, this section briefly describes the mathematical model that is used to approximate the forward kinematics of the real soft-robotic arm in a simulated environment. We also present the experimental setup with the real robot where some preliminary results were derived in order to evaluate the performance of the proposed methodology.

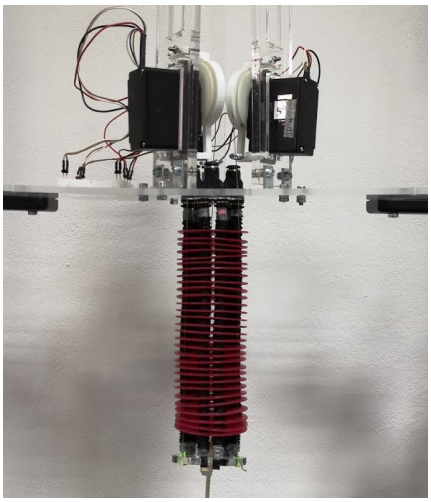

Fig. 2: The soft robotic arm module

1) Constant Curvature Approach for Simulation Model: The present simulation model is based on a piece-wise constant curvature (PCC) approximation that models the forward kinematics of Festo's "Bionic Handling Assistant" [11]. The motion of each physical module constituting the robot is driven by three radially symmetric arranged independent actuators, which change the configuration of the module after modifying their length, resulting in extension, contraction, or omnidirectional bending.

2) Experimental Setup: The performance of the proposed learning controller was evaluated experimentally on a modular soft manipulator (Fig. 2), which is described in [12], [13]. This robotic module has been adapted from the soft-arm which was developed by the Biorobotics Institute (Pisa, Italy) in the frames of the EU project I-SUPPORT. The modules comprising the robot are made up of hybrid actuation, including three radially symmetric tendons driven by three motors, in combination with pneumatic chambers, whose actuation is considered to be fixed in this work. Therefore, the real robot actuation is based on three inputs at the motor control level. In addition, during the experiments conducted on the real setup, the module that is proximal to the base was used (shown in Fig. 2) and was controlled using the same methodology that was first applied in simulations, as will be described in later sections. Regarding position feedback, a 3D magnetic tracker (3D Guidance trakSTAR Class 1 Type B by Ascension Technology Corp.) was used, whose 6-DoF electromagnetic probe is attached at the end of the manipulator, providing its position and orientation.

B. CPG-based Actuation

In this work, the motors of the soft-robotic arm can be seen as oscillators that produce appropriate rhythmic signals, resulting in the generation of cyclic periodic trajectories by the

end-effector. However, the generation of rhythmic patterns of desired features by the robot requires the cooperation of all motors, and thus their operation in a coupled scheme. In this scenario, the signal produced by each actuator is determined by an amplitude value and a phase difference with respect to the signal of an actuator that is predefined as reference.

In [14] the authors proposed a control architecture based on a CPG model implemented as a system of coupled nonlinear oscillators for locomotion control in an amphibious snake robot. The presented scheme provides an ideal building block for doing online generation, while it offers high maneuverability when its control parameters are interactively modulated. Because the properties of their architecture ideally satisfy the requirements of the present soft-robotic arm's actuators, we use a similar CPG model for generating rhythmic input signals.

In particular, an oscillator is assigned to each motor of the soft-robotic arm. Thus the present CPG is implemented as a system of 3 coupled oscillators:

$$\begin{aligned} \dot{\theta}_i &= 2\pi v_i + \sum_j w_{ij} \sin\left(\theta_j - \theta_i - \phi_{ij}\right) \\ \ddot{r}_i &= \alpha_i \left(\frac{\alpha_i}{4} \left(R_i - r_i\right) - \dot{r}_i \right) \\ x_i &= r_i \left(1 + \cos\theta_i\right) \end{aligned} \tag{1}$$

where the state variables θ_i and r_i represent, respectively, the phase and the amplitude of the i^{th} oscillator, and are computed iteratively using a numerical method that allows us to approximate solutions to differential equations. Besides, the parameter v_i is the intrinsic frequency that is assumed to be fixed in our work, while R_i and ϕ_{ij} denote the intrinsic amplitude and the phase biases between oscillators i and j, respectively, and are determined by a learning policy that is described later. Together with the weights w_{ij}, these variables define the coupling between the oscillators, while α_i is a positive constant. The variable x_i is the rhythmic and positive output signal extracted from oscillator i.

To reflect the symmetries of the robot, some parameters are set to the same values for all oscillators. Starting from the intrinsic frequency, its value is determined by the desired time period $1/v$ of the targeted cyclic periodic trajectory produced by the end-effector, which must be the same for all oscillators, i.e. $v_i = v$. Analogously, $\alpha_i = \alpha$ and $w_{ij} = w$ for all connections.

C. Estimation Model

Using the CPG architecture presented in the previous section results in the generation of cyclic periodic trajectories by the end-effector of the soft-robotic arm. Subsequently, the generated path must be compared to the desired one, initially defined by the user. Therefore, we need to evaluate the produced rhythmic patterns in order to assess the efficiency of the whole methodology. Such a trajectory comparison will moreover also determine the decisions taken by the learning policy through the reward function, as will be seen later.

One effective way to evaluate produced movements consists in projecting the generated trajectory to the targeted one, and then designing a suitable heuristic function that quantifies their deviation. The implementation of this approach could be based on a mapping between the points of the two paths. However, this idea comes up with some difficulties, related to the rule that matches the points of two different trajectories.

Instead, we attempted to decompose each trajectory into a set of unique features that characterize it. It is assumed that the position of the end-effector of the soft-robotic arm in the 3D space at every time-step is the only sensory feedback that is available to the algorithm, and thereby to the user. It was also observed through preliminary experiments that a cyclic periodic trajectory can be seen as the result of coupled oscillatory motion on each one of the 3 axes $< x, y, z >$. In particular, regarding each axis, in case of the existence of 1 oscillator, the obtained signal can be fully described using only 2 variables: its amplitude and its phase difference with respect to a reference oscillating signal. Therefore, in our work, where the single frequency assumption was used for the motion on each axis, the generated rhythmic pattern can be described using 5 variables that must be estimated: 3 amplitudes (1 on each axis), and 2 phase differences on axis $< y, z >$ (using the signal of the x-axis as reference).

In [15] the authors propose a system of coupled adaptive nonlinear oscillators that is capable of learning and reproducing arbitrary periodic signals. In this supervised learning framework, the system initially exploits the information provided by the teaching signal, adjusting its parameters such as intrinsic frequencies, amplitudes, and phase differences. The system acts as a dynamic Fourier series representation, where each oscillator encodes one frequency component. After the adjustment phase, the system of differential equations itself is able to replicate the teaching signal without using any external optimization algorithm.

The first part of the algorithm (which includes the dynamic Fourier series representation) can be used in this work in order to estimate the variables that describe the cyclic rhythmic patterns generated by the end-effector. The main difference between the algorithm proposed by [15] and our implementation is that we are aware of the main intrinsic frequency ω_i of the received signals, assuming that it is the same as the frequency defined for the operation of the CPG model in the motors. In addition, we have chosen to approximate the rhythmic signal corresponding to each axis using 2 oscillators that encode its 2 main frequencies $(\omega_i, 2\omega_i)$, even though we are only interested in ω_i.

Regarding the estimation of the amplitude on each axis, the equations describing the dynamic Fourier series representation, in absence of the one that corresponds to the known intrinsic frequency, are as follows:

$$\begin{aligned} \dot{x}_i &= \gamma \left(\mu - r_i^2\right) x_i - \omega_i y_i + \epsilon F(t) + \tau \sin\left(\theta_i - \phi_i\right) \\ \dot{y}_i &= \gamma \left(\mu - r_i^2\right) y_i + \omega_i x_i \\ \dot{\alpha}_i &= \eta x_i F(t) \\ \dot{\phi}_i &= \sin\left(\frac{\omega_i}{\omega_0}\theta_0 - \theta_i - \phi_i\right) \end{aligned} \tag{2}$$

with

$$\begin{array}{lcl} r_i & = & \sqrt{x_i^2 + y_i^2} \\ \theta_i & = & sgn(x_i) \arccos\left(-\frac{y_i}{r_i}\right) \\ F(t) & = & P_{teach}(t) - Q_{learned}(t) \\ Q_{learned}(t) & = & \sum_{i=0}^{N} \alpha_i x_i \end{array} \tag{3}$$

where $i = \{0, 1\}$ is the index of the oscillator, τ and ϵ are coupling constants and η is a learning constant. $P_{teach}(t)$ denotes the teaching signal, while α_i represents the amplitude associated to the frequency ω_i of oscillator i, and ϕ_i is the phase difference between oscillator i and 0.

The equations of coupling between the oscillators of different axes and the learning rule for their phase difference are:

$$\begin{array}{l} \dot{x}_{0,k} = \left(\mu - r^2\right) x_{0,k} - \omega_{0,k} y_{0,k} + \tau \sin\left(\theta_{0,k} - \phi_{0,k}\right) \\ \dot{\phi}_{0,k} = \sin\left(\theta_{0,k-1} - \theta_{0,k} - \phi_{0,k}\right) \end{array} \tag{4}$$

where $(0, k)$ denotes the first oscillators of the kth CPG. Using the results of this system, the phase differences of the rhythmic signal on y- and z-axes, with respect to the oscillations on the x-axis, respectively, are as follows:

$$\begin{cases} \phi_{YX} & = \phi_{0,X} - \phi_{0,Y} \\ \phi_{ZX} & = \phi_{0,X} - \phi_{0,Z} \end{cases} \tag{5}$$

D. Learning Policy

The individual algorithms analysed previously depend on a well-designed mechanism whose output is sent to the actuators, while receiving feedback related to the motion of the robot's end-effector. More precisely, the system, whose input is a targeted cyclic trajectory defined by the user, must be capable of providing the CPG model used for the motors' actuation with appropriately computed parameters obtained by a policy whose update rule depends on its performance. The evaluation process quantifies the deviation between the targeted and the generated trajectories, and is implemented using the dynamic Fourier series representation discussed above.

In order to choose an appropriate class of learning algorithm, the following requirements must be taken into account. First of all, the use of a classical control scheme is not recommended because of the robot's nonlinear properties that limit the accuracy and the efficiency of a fixed mathematical model. One of the main requirements for the designed mechanism is the capability to adapt online to any potential change of the robot's dynamics, which constitutes a usual phenomenon in bio-inspired systems. In addition, the designed policy should be able to generalize its knowledge while working in an unsupervised framework. The operation of a system under these properties imply the use of a reinforcement learning (RL) control scheme [16].

Nevertheless, the need to learn a policy in a continuous environment limits the range of possible RL algorithms. In [17] the authors proposed a new class of model-free algorithms named Continuous Actor-Critic Learning Automaton (CACLA), that is used in combination with Gaussian exploration and can handle continuous variables in both state and action spaces. Regarding its performance, it is mentioned that the presented implementation ensures the ability to find real continuous solutions, while it provides good generalization properties and fast action selection.

Here the state-space was chosen to be composed of the 5 continuous features that represent the desired cyclic periodic trajectory:

$$s_t = \{A_x, A_y, A_z, \phi_{YX}, \phi_{ZX}\}_{desired} \tag{6}$$

where A_x, A_y and A_z are the amplitudes of the target trajectory on axes x, y, and z, respectively, while ϕ_{YX} and ϕ_{ZX} denote the phase differences of the signals on y and z-axes with respect to the one on the x-axis.

Besides, the action-space is composed of the following 5 continuous variables, used by the CPG model that has been implemented in the 3 actuators:

$$a_t = \{A_1, A_2, A_3, \phi_{21}, \phi_{31}\} \tag{7}$$

where A_k for $k = \{1, 2, 3\}$ represents the amplitude sent to motor k, while ϕ_{21} and ϕ_{31} are the phase differences of the oscillators in motors 2 and 3, respectively, with respect to the oscillation in motor 1.

To deal with the continuity of both the state and action spaces, radial basis functions with equally distributed Gaussian functions are used as linear function approximators (FA). This means that a parameter vector θ^V is assigned to the critic FA, while a similar θ^{Ac} is used for the actor FA. As in the original CACLA algorithm, the parameters of each FA are updated based on the following equations:

$$\theta^V_{i,t+1} = \theta^V_{i,t} + \alpha_V \delta_t \frac{\partial V_t(s_t)}{\partial \theta^V_{i,t}} \tag{8}$$

$$\text{IF } \delta_t > 0\text{: } \theta^{Ac}_{i,t+1} = \theta^{Ac}_{i,t} + \alpha_{Ac}\left(a_t - Ac_t(s_t)\right) \frac{\partial Ac_t(s_t)}{\partial \theta^{Ac}_{i,t}} \tag{9}$$

with

$$\begin{array}{lcl} V_{t+1}(s_t) & = & V_t(s_t) + \alpha_t \delta_t \\ \delta_t & = & R_t + \gamma V_t(s_{t+1}) - V_t(s_t) \end{array} \tag{10}$$

where s_t is the state at time t, and $V_t(s_t)$ represents its state value function. α_V is a learning rate, while γ is a discount factor. The latter is here fixed to 0 since we expect the soft-robotic arm's end-effector to generate the desired trajectory as a unitary movement, without learning a sequence of actions. Regarding the actor, α_{Ac} is a learning rate, while Ac_t denotes the action that the actor's FA outputs at time t. a_t is the action that is sampled from a Gaussian distribution with mean Ac_t. Using this kind of exploration method, the policy is defined by:

$$\pi_t(s_t, a_t) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(a - Ac_t(s_t))^2/(2\sigma^2)} \tag{11}$$

where σ denotes the standard deviation of the Gaussian exploration and depends on the obtained reward through a logistic function:

$$\sigma(R) = \frac{h}{1 + e^{c(R+b)}} \tag{12}$$

where h, b and c are positive constants determining its maximum value, the bias on the horizontal axis, and its width, respectively. It is evident that when the reward is small, implying a considerable deviation from the desired state, the Gaussian's width is large enhancing the exploration, while in the opposite case a large reward results in an action that is close to the actor's output with high probability.

The reward function has been chosen to be a heuristic that represents the negative deviation between the desired and the generated cyclic periodic trajectory, and is given as follows:

$$R = -3\sqrt{\frac{(\Delta A_x)^2 + (\Delta A_y)^2 + (\Delta A_z)^2}{D^2(A_x) + D^2(A_y) + D^2(A_z)}} - \\ - \frac{1-\cos(\Delta\phi_{YX})}{2} - \frac{1-\cos(\Delta\phi_{ZX})}{2} \quad (13)$$

with

$$\begin{aligned} (\Delta A_k) &= A_{k_{desired}} - A_{k_{real}} \\ D(A_k) &= \max A_k - \min A_k \\ (\Delta\phi_{kX}) &= \phi_{kX_{desired}} - \phi_{kX_{real}} \end{aligned} \quad (14)$$

where $k = \{x, y, z\}$, and A_k represents the amplitude on the k-axis, while ϕ_{kX} denotes the phase difference of a signal between k- and x-axis. It can be easily seen that the maximum reward is 0 when the desired and generated trajectory have the same features, while its minimum value is -5 and corresponds to the case where all features differ the most.

It should be noted that although the global reward implies the existence of only 1 critic, the presence of 5 variables/agents in the action-space leads to the design of 5 actors, one for each agent that will update its parameter vector independently from the others. However, each update takes place only if $\delta_t > 0$, depending on the global reward, and hence obtained after the cooperation of all actors. Regarding this aspect, if an agent acts correctly but all the others make bad decisions, coming up with a negative δ_t, the first agent is not rewarded for his right choice.

IV. Experiments

The experimental procedure as well as the derived results are presented in this section, which is divided into two parts. At the beginning, the performance of the proposed controller is evaluated on a simulation model, while in the second part, an experimental setup of the real robot validates the research findings. Both setups are described in section III-A.

In both cases, prior to the integration of the RL-based controller, the motors are fed with arbitrary oscillatory signals, whose parameters lie within a range of values, and the features of the cyclic periodic trajectories produced by the end of the manipulator are stored in a dataset S_D, performing a mapping between input and output. This process is of great importance since it results in the definition of the workspace, determining the rhythmic trajectories that are feasible to be executed by the soft-robotic manipulator.

During the experiments, the biased actuation is considered to be fixed with the same value for all actuators, resulting in the generation of periodic signals, and thus cyclic rhythmic trajectories around a prefixed bias configuration.

A. Simulation results

Although the simulation model constitutes only a coarse approximation of the real robot, it is still considered to be a good approach for the forward kinematics. At this point, we evaluate the performance of the proposed architecture in terms of its convergence, repeatability, and generalizability.

For all the results figures hereafter, a sliding window function of fixed width is applied to the data, computing the mean of those between its edges, in order to achieve smoother representation of the results.

The algorithm is tested on only one module, whose length and radius are set to 0.25m and 0.05m, respectively. As for the action-set, the amplitude A_k of the oscillation generated by the CPG model in motor k is limited between 0.005m and 0.02m, while the phase differences ϕ_{21} and ϕ_{31} lie within the intervals $[100°, 140°]$, and $[220°, 260°]$, respectively.

1) Training Procedure: After completing the initial procedure described at the beginning of this section, which aims at the definition of the workspace, the analysis of the training process follows. Starting with the determination of the targets that are learned in this part, we firstly split the dataset S_D into two subsets, with a ratio of 9:1. The first subset S_{D_1}, containing 90% of the feasible targets (almost 2900), is used during the training process, while the rest 10% (approximately 300 target points) defining the S_{D_2} subset proves the ability of the algorithm to generalize during the test process.

During the training process, each target is set as input sequentially while the controller is looking for the appropriate combination of actions that maximizes the obtained reward. Another target follows only if a maximum number of iterations has been reached, or the error lies below a predefined threshold. Each target point is drawn from S_{D_1} by a random picker which chooses each unique target about 12 times. As a result, the RL controller is trained for approximately 35000 samples. At this point, we should point out that in the experimental analysis that follows and the respective plots, the error refers to the absolute value of the reward function R, which is a dimensionless measure of the distance between the features of the desired and the generated trajectories (as defined in Eqs. 13,14). Therefore, its maximum value is 5, while the minimum is 0.

During the first epoch where the targets are chosen for the first time to train the proposed RL scheme, each target point is trained for 1000 iterations with a learning rate 0.1 before the arrival of another target, since the system is learning from scratch. After the first epoch, each sample is trained for 50 iterations with a learning rate 0.01. The left side of Fig. 3 shows the convergence of the algorithm, presenting the error of each sample point after its last iteration.

2) Repeatability: After the end of the training process, we proceed to the validation of the RL controller's repeatability, that assess its capability to reproduce the learned targets instantly, exploiting the learned policy. During this process, each point of S_{D_1} is set as input, while the algorithm executes only 1 iteration attempting to output an action that approaches the target point, without updating its parameters

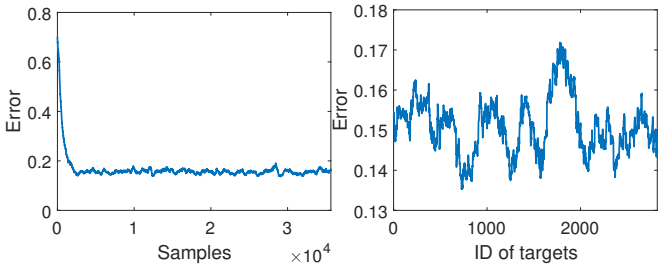

Fig. 3: Convergence - Repeatability

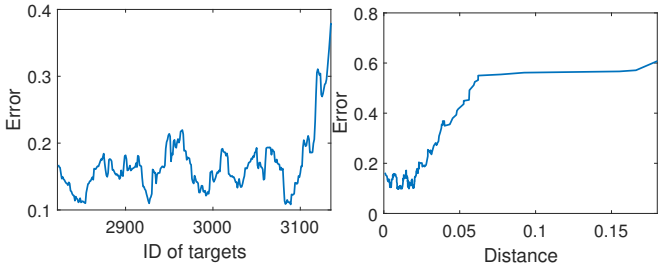

Fig. 4: Generalizability

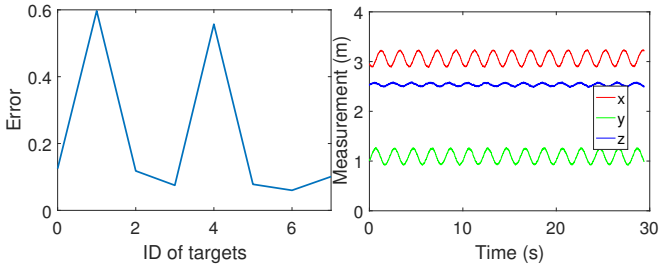

Fig. 5: Repeatability - Measurements

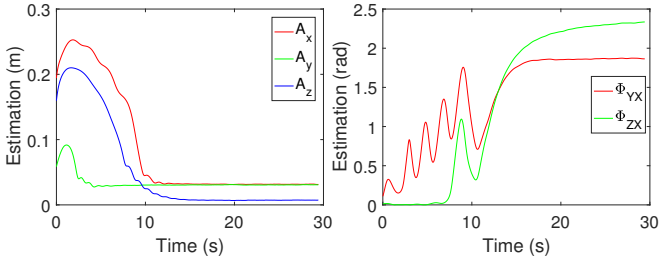

Fig. 6: Estimations

and thus re-converging. Also, the parameter σ of the Gaussian distribution is set to a negligible value in order to limit the exploration. The error obtained for each target point is illustrated in the right side of Fig. 3. The mean value for all samples is 0.1508.

3) Generalizability: The last step of the assessment is to evaluate the ability of the methodology to generalize to the targets of S_{D_2} that have not been used during the training process. We follow the same procedure as in repeatability, resulting in the error presented on the left part of Fig. 4, whose mean value is 0.1672. The right part of Fig. 4 shows the error of each testing target of S_{D_2} with respect to its distance from the closest training point of S_{D_1}, indicating that distant targets are not approached as well as the proximal ones. The distance is computed as the absolute value of the function described in Eq. 13, after replacing the features of the generated trajectory with those of the closest training point of S_{D_1}.

B. Experimental results with the real robot

In this part, we present some preliminary experiments that validate the results obtained by the simulation model, showing that the proposed controller is able to cope with the challenges introduced by the structural properties of the soft-robotic arm.

The experiments are conducted on the module that is proximal to the base. As for the set of actions that were used for the definition of the workspace, the amplitude A_k of the oscillatory motion on motor k is limited between 1750 and 2000 ticks per motor cycle, while the phase differences ϕ_{21} and ϕ_{31} lie within the intervals $[110°, 130°]$, and $[230°, 250°]$, respectively. Eventually, 8 target points were picked randomly from the set S_D.

1) Repeatability: Regarding the ability of the RL controller to reproduce the training targets using the learned policy, we apply the same procedure as in simulation. The left part of Fig. 5 shows the error for each target learned during the training process after testing it to the converged algorithm, resulting in an average of 0.2146.

2) Estimation Model: One of the things that are not tested during the simulations is the convergence of the estimation model. For this purpose, oscillatory signals of arbitrary parameters are generated by the CPG model applied to the motors, resulting in rhythmic trajectories at the end-effector's level, whose features are estimated by the dynamic Fourier series representation proposed in section III-C. The right part of Fig. 5 illustrates the signals received by the magnetic tracker, while Fig. 6 shows the time evolution of the corresponding estimation for each feature, resulting in convergence after about 20 seconds of periodic motion.

V. Conclusion and Discussion

In this work we have presented a methodology that uses a neurodynamic adaptive controller in combination with a CPG model aiming at learning a policy that generates rhythmic trajectories. The key principle is here to progressively build up skills by leveraging on simulated or experimental data while avoiding the use of complex fixed models. The novelty relies on learning periodic movements on a robotic soft arm. The results show that the proposed architecture satisfies the requirements that have been set from the beginning regarding the ability of the algorithm to reproduce both trained and similar generalized trajectories. On the other hand, the estimation model has been proven to be reliable, producing the features of the generated trajectory with fast convergence in simulation, even if the measurements are noisy. In future work, we plan to improve the RL controller, accelerating the convergence in order to learn faster when using the real robot. Furthermore, a long-term goal is to expand the methodology to the coordination of multiple physical modules constituting the robotic arm. Eventually, the action-set could also be extended to include the pneumatic actuation, offering the ability to physically interact with the environment, handling external loads and applying forces.

REFERENCES

[1] A. Kapadia, I. Walker, D. Dawson, and E. Tatlicioglu, "A model-based sliding mode controller for extensible continuum robots," pp. 113–120, 02 2010.

[2] V. Falkenhahn, A. Hildebrandt, R. Neumann, and O. Sawodny, "Dynamic control of the bionic handling assistant," *IEEE/ASME Transactions on Mechatronics*, vol. 22, no. 1, pp. 6–17, Feb 2017.

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

[4] I. S. Godage, G. A. Medrano-Cerda, D. T. Branson, E. Guglielmino, and D. G. Caldwell, "Dynamics for variable length multisection continuum arms," *The International Journal of Robotics Research*, vol. 35, no. 6, pp. 695–722, 2016. [Online]. Available: https://doi.org/10.1177/0278364915596450

[5] D. Braganza, D. M. Dawson, I. D. Walker, and N. Nath, "A neural network controller for continuum robots," *IEEE Transactions on Robotics*, vol. 23, no. 6, pp. 1270–1277, Dec 2007.

[6] A. J. Ijspeert, "Central pattern generators for locomotion control in animals and robots: A review," *Neural Networks*, vol. 21, no. 4, pp. 642 – 653, 2008, robotics and Neuroscience.

[7] A. J. Ijspeert and J. Kodjabachian, "Evolution and development of a central pattern generator for the swimming of a lamprey," *Artif. Life*, vol. 5, no. 3, pp. 247–269, Jun. 1999.

[8] C. LI, R. Lowe, and T. Ziemke, "Humanoids learning to walk: A natural cpg-actor-critic architecture," *Frontiers in Neurorobotics*, vol. 7, p. 5, 2013.

[9] C. Li, R. Lowe, and T. Ziemke, "A novel approach to locomotion learning: Actor-critic architecture using central pattern generators and dynamic motor primitives," *Frontiers in Neurorobotics*, vol. 8, p. 23, 2014.

[10] 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, Feb 2019.

[11] M. Rolf and J. J. Steil, "Constant curvature continuum kinematics as fast approximate model for the bionic handling assistant," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2012, pp. 3440–3446.

[12] 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.

[13] 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)*, June 2016, pp. 833–838.

[14] A. J. Ijspeert and A. Crespi, "Online trajectory generation in an amphibious snake robot using a lamprey-like central pattern generator model," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, April 2007, pp. 262–268.

[15] L. Righetti and Auke Jan Ijspeert, "Programmable central pattern generators: an application to biped locomotion control," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, May 2006, pp. 1585–1590.

[16] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*, ser. Adaptive Computation and Machine Learning series. MIT Press, 2018.

[17] H. van Hasselt and M. A. Wiering, "Reinforcement learning in continuous action spaces," in *2007 IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*, April 2007, pp. 272–279.