Task Driven Skill Learning in a Soft-Robotic Arm

Paris Oikonomou, Athanasios Dometios, Mehdi Khamassi, Costas S Tzafestas

Task Driven Skill Learning in a Soft-Robotic Arm

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

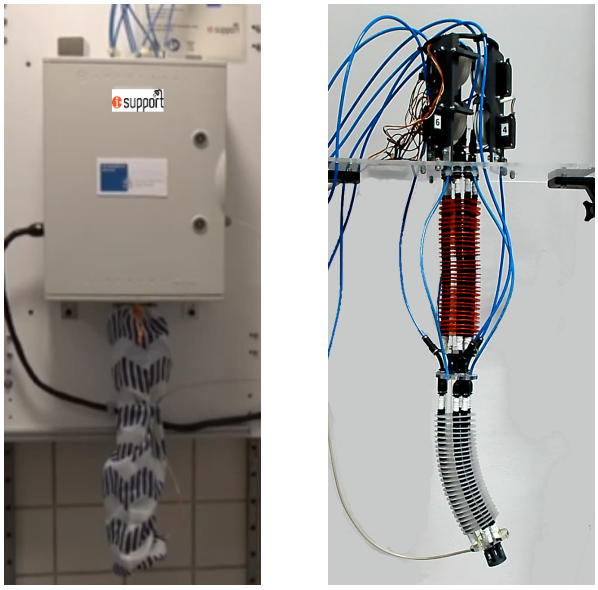
***Abstract*— In this paper we introduce a novel technique that aims to dynamically control a two-module bio-inspired soft-robotic arm in order to qualitatively reproduce a path defined by sparse way-points. The main idea behind this work is based on the assumption that a complex trajectory may be derived as a combination of a discrete set of parameterizable simple movements, as suggested by Movement Primitive (MP) theory. Capitalising on recent advances in this field, the proposed controller uses a Probabilistic MP (ProMP) model which initially creates an abstract mapping in the primitive-level between the task and the actuation space, and subsequently guides the movement's composition by exploiting its unique properties - conditioning and blending. At the same time, a learning-based adaptive controller updates the composition parameters by estimating the inverse kinematics of the robot, while an auxiliary process through replanning ensures that the trajectory complies with the new estimation. The learning architecture is evaluated on both a simulation model, and a real soft-robotic arm. The research findings show that the proposed methodology constitutes a novel approach that successfully manages to simplify the trajectory control task for robots of complex dynamics when high-precision is not required.**

***Index Terms*— Robot Learning, Probabilistic Movement Primitives, Reinforcement learning, Soft Robotics**

I. Introduction

Well organized health care systems are increasing the life expectancy of modern societies, according to World Health Organization's research on health and aging [1]. Personal care (showering or bathing) is included among the first Activities of Daily Living (ADL), which incommode an elderly's life [2]. The present work capitalises on the outcomes of the I-SUPPORT project (EU H2020 grant agreement no. 643666), which contributed to robotics research effort for daily assistance through the development of an innovative, modular robotic system integrating a soft-robotic arm (Fig. 1), that aims at supporting frail older adults to safely and independently complete various physically and cognitively demanding bathing tasks, such as properly washing their back and their lower limbs.

In this context, bathing tasks should be executed in a human-friendly way in order to increase the comfort of an elderly user. Thus, proper washing motions for each task should be learned through demonstrations by health care experts. Expert demonstration in the task space might raise some requirements for each task, in terms of execution time and motion complexity. However, decomposition into simpler primitive motions is necessary for a robotic device for technical reasons. The fusion of such primitive actions with different parameters (e.g. duration, amplitude etc.) can reproduce more delicate and human-friendly actions. Dynamic Movement Primitives (DMPs) proposed in [3] are nonlinear attractor systems which are mostly used for trajectory control since they provide stability and scalability properties. A more recent work described in [4] and [5] presents the concept of Probabilistic Movement Primitives (ProMPs) which acts as a probabilistic framework with enhanced properties, such as conditioning or blending, that allows the exploitation of the properties of trajectory distributions for representing and learning primitives actions.



(a) Soft-Robot in Clinical Environment

(b) Soft-Robot in Lab

Fig. 1: The two module Soft-Robotic manipulator developed during the i-Support project, (a) installed in a clinical environment and (b) a replicated version in the lab.

In addition, an interactive bathing application, which involves direct human-robot physical contact, is way more demanding in terms of safety than other assistive tasks. The execution of such tasks by a rigid robotic manipulator is considered risky, hence the research attention has shifted to soft robots [6] with inherent or structural compliance, which gives them the ability to actively interact with the environment and undergo large deformations. Many continuum manipulators have already been presented with tendon [7] or pneumatic actuation [8] or a combination of those [9]. The more complex the mechanical and actuation structure of the robot is, the more sophisticated kinematic analysis and control scheme is required. Although, analytic kinematic

[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.

Email: oikonpar@mail.ntua.gr

models based on constant curvature assumption have been established [10], powerful control strategies for continuum manipulators are still being developed [11].

Recent research work [12] specifically addressed the problems related to the use of continuum robots to perform dynamic control through model-based approaches. A similar approach is presented in [13], 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. A different approach is proposed in [14], based on the design of open-loop predictive controllers and machine learning dynamic models directly from the actuation to the task space. Based on a different set of techniques, the work presented in [15] applies novel spatial dynamics to variable length multisection continuum arms assuming circular arc deformation of continuum sections without torsion. A relevant approach is presented in [16] where the authors use a feedforward neural network component to compensate for dynamic uncertainties.

In the literature, the number of control approaches that are applied on the same soft-robot, allowing for a fair performance comparison, is limited and only two of them focus on dynamic control. In our previous work [17], a model-free neurodynamic control scheme is proposed in combination with Central Pattern Generators (CPGs) assigned with the task of generating cyclic motion patterns of desired features by the end-effector of the soft-robotic arm. Despite the complexity of the achieved task, this implementation is strictly limited to produce periodic movements in the task space. In this work, we focus on a more robust implementation that extends and broadens the capabilities of the approach proposed in [17], enabling the soft-arm to follow a more generic path.

Another control architecture evaluated on the same hardware is proposed in [18], where a closed-loop predictive controller was implemented with a model-based policy learning algorithm and trained using trajectory optimization and supervised learning. However, such control schemes require high computational resources, big amount of data, and a lot of iterations in order to successfully reproduce a single trajectory, since they focus on accuracy at every time step, while convergence during training is time-consuming. As a consequence, it is hard to generalize their performance to unseen trajectories, thus failing to ensure robustness. Besides, in most applications that use soft-robots, high accuracy in the trajectory level is not a principal requirement, while a qualitative reproduction with similar motion properties of the demonstrated trajectory is preferred, which constitutes the main focus of this work. Additionally, in [18], achieving some key motion execution properties, such as the capability to control the completion of the trajectory execution within a desired duration, also known as temporal scalability, is not evident.

In this paper, we present an architecture that aims to dynamically control a two-module bio-inspired soft-robotic arm. Our approach focuses on the implementation of a learning-based control scheme inspired by motor control theory, that has the ability to qualitatively reproduce a path defined by sparse way-points. This simplification of the trajectory control task is proved to be useful for robots of complex unmodeled dynamics. In particular, our approach combines: (a) Movement Primitive (MP) theory which provides a mathematical model capable of building a mapping in the trajectory-level between task and actuation spaces, while also producing complex movements as the result of the combination of primitives that are formed by simple demonstrations, and (b) a Reinforcement Learning (RL) algorithm assigned with the task of approaching the unknown inverse kinematics of the soft-robot in a multi-task learning manner as it is required by the nature of the task. In parallel, an extension of the ProMP's properties is designed that provides the ability to replan the route of the trajectory during the course of the execution. The efficiency of the complete methodology is evaluated in both a simulation model and a real soft-robot. The set of preliminary experimental results presented in this paper highlights the ability of the proposed framework to dynamically control a complex robotic system with unknown dynamics.

The following section introduces the reader to the problem statement before proceeding to the description of the control architecture in Section III. Experimental evaluation results are presented in Section IV, while concluding remarks together with indicative future research directions are provided in Section V.

II. Problem Statement

A soft-robot like the one examined in the present work (Fig. 1), is not often assigned with the task of path following, especially when high-precision is required; the mechanical properties of its design is mostly exploited in tasks where safety must be ensured, such as those involving human-robot interaction or manipulation of fragile materials. The task examined in this paper does not differ from this class of applications, hence our focus lies on the qualitative reproduction of primitive actions defined in a subspace of a soft robot's workspace. The desired motion's properties are derived by analysis and decomposition of human expert's demonstrations which are usually conducted in the task space. In some approaches, kineasthetic teaching [19] of the robot by a user is employed, in order to overcome difficulties related to the demonstration's mapping to the actuation space. However, in the present application where kineasthetic teaching is impossible due to the mechanical structure of the soft-robot, generation from demonstrations should be redefined to cover the required subspace.

The mapping between the task and joint space might be provided by an inverse kinematic model, like in [5] where a model is extracted based on the known geometry of a rigid manipulator. However, in cases where the design of such a mathematical model is difficult, because of the complexity or the stochasticity introduced by the robot's dynamics, different methods are recommended. In recent years, common approaches that cope with the difficulty to map the task-space to the actuation-level, involves the design

of a learning-based controller that focuses on model learning [20], [21]. The methods proposed in these papers, either use supervised-learning techniques training a model offline without being capable of adjusting online its parameters and hence its behavior, or focus on extracting a dynamic model for the robot, which constitutes a different problem since the system usually is highly dependent on previous states.

As already stated in [17], the non-linearity as well as the stochasticity of a soft robot, introduced by its mechanical structure, and soft-properties do not encourage the use of a classical control scheme and the design of a fixed mathematical model that predicts its behavior, thus the design of a learning-based controller is the only way forward. At the same time, two of the main requirements that must be satisfied by the chosen class of learning algorithms are the generalizability and the online adaptability to any potential change of the robot's dynamics - which constitute a usual phenomenon in bio-inspired systems. Under these conditions, a Reinforcement Learning (RL) control scheme [22] seems to be suitable for approximating the inverse static model under an unsupervised framework.

III. Control Architecture

A. Preliminaries

In this section, we briefly describe the simulation model, while 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.

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" [23]. 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 two-module soft manipulator (Fig. 1), which is described in [24], [25]. This robotic module has been adapted from the soft-arm which was developed by the Biorobotics Institute (Pisa, Italy) in the frame of the EU project I-SUPPORT. Each module comprising the robot is 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 six inputs at the motor control level. 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. Trajectory extraction from demonstrations

One of the main ideas behind the ProMP theory introduced in [4] that is exploited in this work, is based on its capability

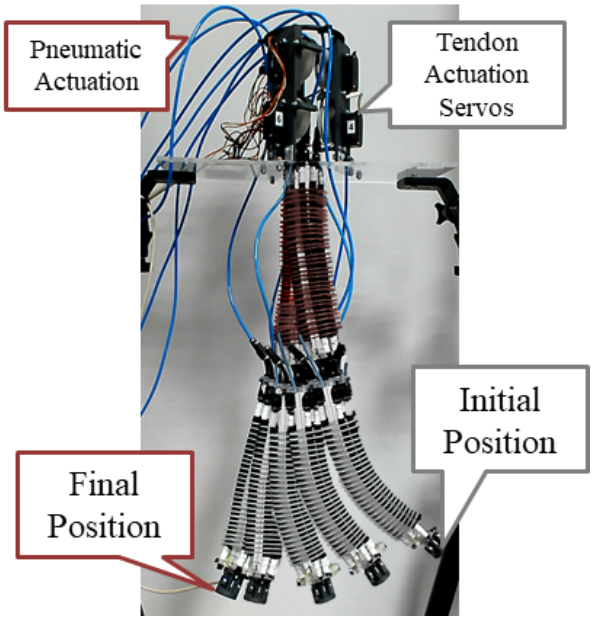


Fig. 2: Execution of a right to left demonstration motion generated with coordinated activation and motion of all motors simultaneously. Various demonstrations are generated in order to cover the desired subspace of the robot's workspace.

to form a single primitive from demonstrations that are not identical but they accomplish similar tasks, due to its property to model sequence of states as a trajectory distribution under a probabilistic framework. When experimenting with soft-robots, this seems to be quite convenient since the stochasticity introduced by their soft-properties does not guarantee reproduction of the same result even when the same actuation is applied. In addition, the ability to reach novel targets by conditioning and blending multiple primitives could cope with the requirement to pass through some pre-determined way-points at specific timestamps during the execution of the trajectory.

In some robotic applications, the design of a dynamic control scheme based on classical optimal control theory is not feasible due to the absence of a fixed mathematical model that approximates robot's dynamics - the same phenomenon is observed in the present case. Movement Primitive (MP) theory could provide a straightforward solution that focuses on building a mapping between task and actuation spaces in a trajectory-level, in which a demonstration in the first corresponds to a sequence of actions in the motors, and vice-versa. Exploiting the idea of ProMPs where a single primitive consists of multiple similar demonstrations performing the same task, a correspondence between primitives in both spaces - task and actuation - is formed. Following this strategy, a task-driven skill learning process is implemented in the sense that the composition of the requested trajectory is performed at a high-level using the properties provided by ProMP theory - conditioning, combining and blending primitives. Subsequently, the composition's parameters as extracted in the previous step are transferred unchanged in the actuation space, where the corresponding primitives are activated accordingly, resulting in the generation of a

trajectory by the robot's End-Effector (EE) which is similar to the desired one.

The training process of ProMP model and more specifically the extraction of demonstrations is crucial for the proposed methodology's performance. At the beginning, a subspace within the workspace is defined as a region of interest for the requested trajectories, where the demonstrations should lie in. Subsequently, each motor is assigned with the task to start moving from a random point defined by a mean and a standard deviation, towards another end-point defined accordingly as shown in Fig. 2. It is evident that both extreme points should lie within the subspace. Randomness between demonstrations is enhanced by forcing the motors to pass through some median random points, lying between the two extremes, while smooth transition between consecutive points is guaranteed with linear interpolation. Therefore, a demonstration is derived as the coordinated activation and motion of all motors simultaneously. The same process is followed for the extraction of various demonstrations, ensuring that all of them results in movements of similar directions in the task-space. Before proceeding to training, the resulting trajectories are grouped into classes, under a similarity criterion. In the present case, it is assumed that all demonstrations generated by a distribution around the same extreme points in the actuation-space, belong to the same primitive. During the last step, the estimation of each class' parameters takes place so that each primitive, defined as trajectory distribution, represents the variations of all corresponding demonstrations.

Algorithm 1: CACLA-MTL$(n, n_+, a^{(n)}, P_{cond}[n_+])$

Def `__init__`$(n, N, P_{cond}[n])$**:**
- $k = 0$
- $\theta_k^V = RestoreCriticTheta()$
- $\theta_k^{Ac} = RestoreActorTheta()$
- $k_n = 0$, for all $n = 1, .., N$
- $g_{k_n}^{(n)} = 0$, for all $n = 1, .., N$
- $a^{(n)} = Ac_k(P_{cond}[n])$
- **return** $a^{(n)}$

end

Def `__call__`$(n, n_+, a^{(n)}, P_{cond}[n_+])$**:**
- $k = k + 1$
- $k_n = k_n + 1$
- `// For action` $a^{(n)}$ `::`
- Observe & store current state for point n $s_{k_n}^{(n)}$
- Compute & store error $g_{k_n}^{(n)}$ (Eq. 9)
- Compute reward $r_k^{(n)}$ (Eq. 10)
- Compute TD-error δ_k (Eq. 6)
- Update Critic θ_k^V (Eq. 4)
- **if** $\delta_k > 0$ **then**
  - Update Actor θ_k^{Ac} (Eq. 5)
- **end**
- `// For target` $P_{cond}[n_+]$ `::`
- Sample $a^{(n_+)}$ (Eq. 7)
- **return** $a^{(n_+)}$

end

C. Learning-based controller for inverse static model

In the frame of this work, a variation of Continuous Actor-Critic Learning Automaton (CACLA) proposed in [26], [27] has been designed, due to its ability to handle continuous variables in both state and action spaces. The goal of the present application is to execute a trajectory whose only requirement - apart from the evident that must be similar to the paths obtained by demonstrations - is that the robot's EE passes through the consecutive way-points defined by the user in the task-space. Before proceeding to the description of the CACLA's variation, it should be clarified that each way-point $P_{cond}[n]$ constitutes a target, for which the learning-based controller is assigned to the task to find the corresponding actuation $a^{(n)}$; n is the id of the conditioning point in set P_{cond}. This fact implies that, during the execution of the trajectory the target is changed right after the EE passes through a way-point at the corresponding timestamp, so the actuation of the succeeding way-point is requested. In this way, the conditioning points are assigned periodically as targets, assuming that the each trajectory is executed more than once until the proximity to the way-points satisfies a convergence criterion.

To address the constant change of controller's target, a modified version of CACLA is designed, named Continuous Actor-Critic Learning Automaton for Multi-Task Learning (CACLA-MTL) which differs from the original in the sense that it keeps track of the current target's state in order to recall it when the same target is assigned to the controller. At the same time, the actuation of the succeeding target is computed after updating the actor-critic model for the current one. A main requirement that must be satisfied is that the multiple parallel tasks must concern the same plant since all targets share the same actor-critic model $\left(\theta^V, \theta^{Ac}\right)$. The steps of the proposed CACLA-MTL written in pseudo-code is presented in Alg. 1.

Note that, CACLA-MTL presented in this paper refers and is applied only to the conditioning points of the ProMP model. Thus in the rest of the current paragraph, subscript $cond$ is omitted from all variables. Also, for reasons of simplicity, the following applies:

$$\begin{aligned} V_k(s_{k_n}^{(n)}) &= V(s_{k_n}^{(n)}, \theta_k^V) \\ Ac_k(s_{k_n}^{(n)}) &= Ac(s_{k_n}^{(n)}, \theta_k^{Ac}) \end{aligned} \tag{1}$$

where subscripts k denotes the global time-step, while k_n is the time-step at which the update of the conditioning point n takes place.

Here, the state-space of the conditioning point n of the ProMP model at time k_n was chosen to be composed of 6 continuous features in the task-space level; 3 of them represent the desired EE's position of the conditioning point n, while the remaining indicate the corresponding current EE's position:

$$s_{k_n}^{(n)} = \left\{ \boldsymbol{p}_{desired}, \boldsymbol{p}_{current} \right\}_{k_n}^{(n)} \tag{2}$$

where $\boldsymbol{p} = \{p_x, p_y, p_z\}$ denotes the EE's position along the x,y,z-axis, respectively. As already implied, in this application the desired position in the task-space coincides to the way-point $P_{cond}[n]$.

At this point, it should be noted that the critic model is dependent of all the variables of the state-space as presented above, while the actor model uses only the first component - the $\mathbf{p}_{desired}$, indicating the desired EE's position; we expect the proposed learning algorithm to compute directly the actuation that corresponds to the desired EE's position as a unitary movement, without depending on its current position, and/or learning a sequence of actions to reach the target.

Besides, the conditioning point n of the ProMP model in the actuation level is composed of the following 6 continuous variables, each one of which refers to an actuator of the robot:

$$a^{(n)} = \{a_{11}, a_{12}, a_{13}, a_{21}, a_{22}, a_{23}\}^{(n)} \tag{3}$$

where a_{ij} for $i = \{1, 2\}$ and $j = \{1, 2, 3\}$ represents the actuation of motor j that belongs to module i of the robot.

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, here the parameters of each FA are updated based on the following equations, with the only difference being the introduction of superscript (n), which indicates that a variable refers to the conditioning point n, as shown below:

$$\theta_k^V = \theta_{k-1}^V + \alpha_V \delta_k \nabla_{\theta^V} V_{k-1}(s_{k_n-1}^{(n)}) \tag{4}$$

$$\theta_k^{Ac} = \theta_{k-1}^{Ac} + \alpha_{Ac} \left(a^{(n)} - Ac_{k-1}(s_{k_n-1}^{(n)})\right) \nabla_{\theta^{Ac}} Ac_{k-1}(s_{k_n-1}^{(n)}) \tag{5}$$

with

$$\delta_k = r_k^{(n)} + \gamma V_{k-1}(s_{k_n}^{(n)}) - V_{k-1}(s_{k_n-1}^{(n)}) \tag{6}$$

where $s_{k_n}^{(n)}$ is the state of conditioning point n at time-step k_n, and $V_k(s_{k_n}^{(n)})$ represents its state value function. α_V is a learning rate, while γ is a discount factor. Regarding the actor, α_{Ac} is a learning rate, while Ac_k denotes the action that the actor's FA outputs at time-step k. $a^{(n)}$ is the action that is sampled from a Gaussian distribution with mean Ac_k, and corresponds to the conditioning point n. Using this kind of exploration method, the policy is defined by:

$$\pi_k\left(s_{k_n}^{(n)}, a^{(n)}\right) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\left(a^{(n)} - Ac_k(s_{k_n}^{(n)})\right)^2}{2\sigma^2}\right) \tag{7}$$

where σ denotes the standard deviation of the Gaussian exploration through a logistic function:

$$\sigma(g) = \frac{h}{1 + e^{-c(g+b)}} \tag{8}$$

where h, b and c are positive constants determining its maximum value, the bias on the horizontal axis, and its width, respectively. σ is also dependent on the variable g, which is the last observed error for conditioning point n in the task-space, and is defined as the corresponding Euclidean distance between the desired and the current EE's position:

$$g_{k_n}^{(n)} = \|\boldsymbol{p}_{k_n,desired}^{(n)} - \boldsymbol{p}_{k_n,current}^{(n)}\| \tag{9}$$

It is evident that when the error is large, implying a considerable deviation from the target, the Gaussian's width is large, thus enhancing the exploration, while in the opposite case a small error results in an action that is close to the actor's output with high probability.

The reward function at time-step k has been chosen to be a heuristic that represents how close to, or how far from the target position of the conditioning point n, the EE has been moved in relation to its previous state $s_{k_n-1}^{(n)}$, and is given as follows:

$$r_k^{(n)} = g_{k_n-1}^{(n)} - g_{k_n}^{(n)} \tag{10}$$

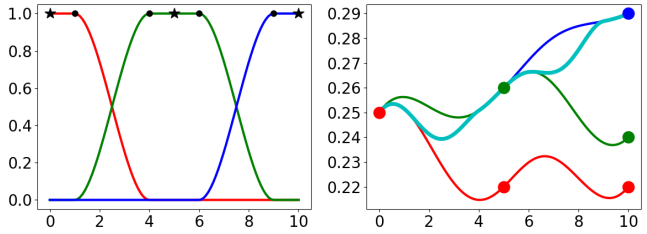
It can be easily seen that the instant reward could be either positive or negative, in case the EE at the conditioning point n moved towards or away from the desired position, respectively.

D. Re-planning in the ProMP-level

As stated earlier in this section, the proposed learning-based controller CACLA-MTL is assigned with the task to provide the ProMP model with an estimation of the actuation that corresponds to the desired way-point in the task-space, as the latter is defined by the user. The update of the inverse statics' estimation takes place during the execution of the trajectory by the soft-robot, implying that a new estimation of the succeeding actuation is available right after the sequence of actuations passes through a conditioning point. On the other hand, the execution of the trajectory is preceded by the weight formation of the ProMP framework which ensures that the sequence of actions taking place in the motor-level passes through the estimated conditioning points. The last argument implies that the trajectory extracted by the ProMP model is invariant during the execution, under normal circumstances.

Re-planning in the ProMP-level facilitates the exploitation of the new estimation of the anchor points, as they extracted by the updated CACLA-MTL model, performing necessary changes online rather than after the end of the execution. In the present work, claiming that the trajectory re-plans its route, means that the power of the dominant instance of blended primitives is gradually decreased over the course of the execution, and is replaced by another instance that involves the updated estimation of the succeeding actuation. Fig. 3 depicts an illustrative example of transitions between instances of blended primitives.

At this point it should be clarified that as implied previously, the CACLA-MTL controller updates the actuation's estimation model right after the sequence of actions passes through a conditioning point in the actuation-level, indicating that the controller is updated as many times as the number of way-points during the course of a single execution. As a



(a) Activation Functions $b_i(t)$ in time (s)

(b) Replanning in time (s)

Fig. 3: (a) Stars (*) indicate way-points, and circles (o) the time-interval where $b_i(t)$ remains fixed. (b) The activation of Red-Green-Blue trajectories is indicated by the corresponding $b_i(t)$ in (a), while the Cyan is the resulted trajectory generated by replanning.

consequence, the number of activation functions as well as the number of instances of blended primitives that operate in parallel, coincide with the number of conditioning points. In the rest of this section it is assumed that N groups of primitives u_i are introduced, which are of identical structure, meaning that each one contains a copy of all primitives $u_{i,j}$, trained using the process described in Section III-B.

The intuition behind the process of re-planning in the ProMP-level is quite similar to the one that is followed when applying the blending property in the ProMP theory as described in [4], [5], that it could be seen as a blending of blended primitives. Indeed, a time-varying activation function $b(t)$ ensures stable and smooth transition between the trajectories as it changes gradually from 0 to 1 and vice-versa in order to activate or disable, respectively, a sequence of actions. Concretely, it only differs in the aspect that the first is computed as the product of the exponentiation of every single primitive $u_{i,j}$ to the power of the corresponding activation function $a_j(t)$, while the latter is the sum of the product between every instance of the blended primitives u_i and the corresponding activation function $b_i(t)$, as indicated in the following equation:

$$v(t) = \sum_{i=1}^{N} b_i(t) \left[\prod_{j=1}^{M} u_{i,j}(t)^{a_j(t)} \right] \tag{11}$$

where the actuation at time t is computed. Another small but not minor difference between the two procedures is that there is a time-interval around the critical area of each conditioning point where the activation function $b(t)$ remains fixed in order to facilitate the evaluation of the controller's performance. The example in Fig. 3 depicts the evolution of activation function $b(t)$ as well as the replanned trajectory of $v(t)$.

E. Complete Methodology

The ProMP training process is followed by the execution of the proposed methodology, presented step-by-step in Alg. 2, resulting in the generation of trajectories by the robot's EE. Few elements of Alg. 2 are given below: At the beginning, the user should define the set P_{cond} of conditioning points in the task-space and the corresponding timestamps T_{cond} where the first takes place. Afterwards,

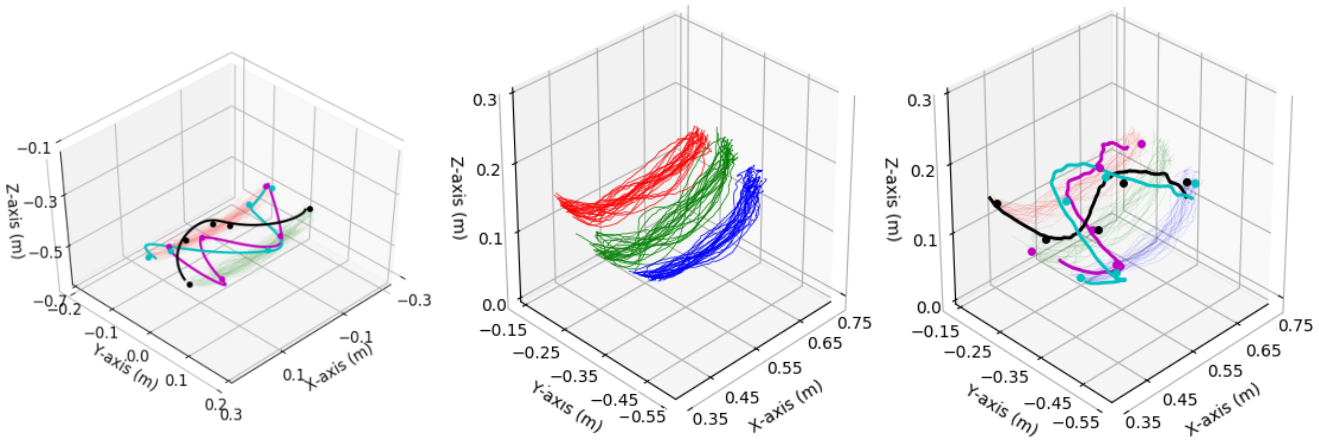
Algorithm 2: Methodology

Set P_{cond} and corresponding T_{cond}
N : number of conditioning points
M : number of primitives
n : ID of current conditioning point
n_+ : ID of subsequent conditioning point
u_i : group of primitives i
$u_{i,j}$: primitive j of group i
Q = **ClassifyToPrimitives**(P_{cond})
$a(t)$ = **ActivationFunA**(T_{cond})
$b(t)$ = **ActivationFunB**(T_{cond})
Set $n = 1$
$a_{cond}^{(n)}$ = **CACLA-MTL-init**$(n, N, P_{cond}[n])$
$u_{n,Q[n]}$ = **UpdatePrim**$\left(a_{cond}^{(n)}\right)$
while *Not Converged* **do**
  $t_{start} = getCurrentTime()$
  $t = 0$
  while *Not the End of Trajectory* **do**
    Compute & execute actuation $v(t)$ (Eq. 11)
    if $t == T_{cond}[n]$ **then**
      $n_+ = n \pmod N + 1$
      $a_{cond}^{(n_+)} =$ **CACLA-MTL**$(n, n_+, v(t), P_{cond}[n_+])$
      $u_{n_+,Q[n_+]}$ = **UpdatePrim**$\left(a_{cond}^{(n_+)}\right)$
      $n = n_+$
    end
    $wait()$
    $t = getCurrentTime() - t_{start}$
  end
end

each element $P_{cond}[n]$ is classified to the primitive Q_n whose parameters - weights' mean and standard deviation - differ the least, using Mahalanobis distance as heuristic. Regarding the next steps, note that the activation functions are formed depending only on T_{cond}, while they remain invariant as long as P_{cond} and T_{cond} are too. Subsequently, the algorithm runs constantly until a convergence criterion is satisfied regarding the estimation of conditioning points. The estimation update $a_{cond}^{(n_+)}$ of conditioning point n_+ is performed after the actuation passes through the precedent point $a_{cond}^{(n)}$ when its evaluation takes place. Later when conditioning point $a_{cond}^{(n_+)}$ is available, reformation of $u_{n_+,Q[n_+]}$'s weights is performed in order to re-plan the route of the actuation's sequence and pass through the updated estimation.

IV. Experimental Evaluation

In this work, more than one algorithms are designed and proposed in order to fulfil the requirements that have been initially set. Therefore, the evaluation should focus on their capability to perform as an entity in a collaborative way rather than on the efficiency of each algorithm individually. A set of preliminary experimental results is presented



(a) Motion Execution in Simulation (b) Real Robot Demonstrations (c) Motion Execution in Real Robot

Fig. 4: Experimental results presented both for simulated and real robot. (a) The execution of three different trajectories defined by their conditioning points in time are shown for simulation environment with magenta, black and cyan, respectively. (b) The generated demonstrations on the real robot are grouped in different ProMPs marked with RGB colors. These demonstrations cover the whole subspace of interest in which the robot will operate. (c) The execution of three different trajectories, defined by their conditioning points in time, are shown for the real robot with magenta, black and cyan, respectively. The proposed approach is able to generalize in novel motions with the subspace of interest.

TABLE I: Mean error values (in cm) for 10 consecutive executions of experimental trajectories described with their Dominant ProMP (Fig. 4c) with real Soft-arm. The errors are measured in each conditioning point (CP) in the respective time

Trajectory ID	Dominant ProMP	**CP1**	**CP2**	**CP3**	**CP4**	**CP5**
1 (Magenta)	11232	3.1	1.9	2.5	3.3	4.3
2 (Black)	32221	2.9	3.2	2.3	1.2	3.9
3 (Cyan)	31133	2.0	2.4	2.6	1.0	4.0

demonstrating the capabilities of the proposed approach and the novelty of this work. The proposed dynamic control scheme is trained, while its performance is evaluated on both the simulation and the real robot, which are presented in Section III-A. Before proceeding to the execution of the proposed methodology, demonstrations are extracted and grouped into classes following the process described in Section III-B, capturing not only the generated paths in the task-space illustrated in Fig. 4b, but also the sequence of actions resulting in each path.

As a consequence, a dataset D_{fwd} of EE's points with the corresponding actuation is formed that is exploited as follows: a Radial Basis Function Neural Network (RBF-NN) [28] is trained using the extracted information in order to form a model that approximates the forward kinematics of the robot. Such a model facilitates the learning process since it gives the possibility to train the proposed controller offline, providing a good initial estimate of the model before the online execution. This technique is recommended when the convergence is proved to be time-consuming due to the absence of a fixed mathematical model that approximates the robot's behavior. Note that, the same process is followed for the simulation even though the latter is executed in the software level. The evaluation of the RBF-NN on D_{fwd}'s observations (error in simulation: [3.7$\pm$1.6]mm, error in real softarm: [6.3$\pm$3.4]mm) ensures that it approximates adequately the forward kinematics.

The training of the CACLA-MTL is then performed offline for both configurations - simulation and real robot - using the RBF-NN model. It should be noted that at the beginning this process takes place independently of the ProMP, since the controller is randomly initialized and potentially results in actuations that are out of the predefined subspace of demonstrations. Therefore, even though all algorithms should be evaluated as an entity, CACLA-MTL's performance is clearly independent of the ProMP. At every epoch, N points are selected randomly in the task-space considered as targets constituting the set P_{cond} of conditioning points, and CACLA-MTL is executed continuously until a criterion is satisfied; all conditioning points of P_{cond} simultaneously result in error that is under a predetermined threshold, or the maximum number of iterations is exceeded. Considering the distance between the RBFs forming the linear FA (30mm in critic, and 15mm in actor), the evaluation of CACLA-MTL on D_{fwd}'s observations (error in simulation: [17.3$\pm$9.4]mm, error in real softarm: [22.2$\pm$12.8]mm) proves that the controller sufficiently models the inverse kinematics.

The training of each MP as described in [5] using the corresponding set of demonstrations is followed by the assessment of the complete methodology. At this step, the learned models are used by the robot in order to perform some desired tasks, and are evaluated in terms of repeatability and skill transfer. As proposed in Alg. 2 the user defines a set of conditioning points P_{cond} in the task-space along with the corresponding timestamps T_{cond} and the algorithm is executed, resulting in the generation of the trajectory only once. As implied in Alg. 1 during the first execution the component of Gaussian exploration is removed for all $P_{cond}[n]$ and the pure estimation of the learned model is used without performing online correction. Figures 4a and 4c depict three trajectories in the task-space produced in simula-

tion and the real robot, along with the corresponding desired conditioning points. From Table I we see that the mean error in each conditioning point for consecutive execution (10 times) of three different trajectories is small, demonstrating the robustness of our algorithm. These results show that the dynamic control scheme proposed in this paper is capable of producing and generalizing in complex movements out of simple demonstrations on a soft-robotic arm.

A more illustrative presentation of the real robot's performance is given in the video accompanying the manuscript.

V. Conclusion and Discussion

In the frames of this work, apart from the application of the existing ProMP theory on a soft-robotic arm, two new methods are proposed: (i) a variation of CACLA for multi-task learning named CACLA-MTK and (ii) a technique for replanning in the ProMP-level. Both implementations are used as auxiliary algorithms to the ProMP in order to facilitate the trajectory generation in the task-space, providing online correction and adaptation capabilities. The results show that the proposed architecture satisfies the requirements that have been set from the beginning regarding the ability of the algorithm to qualitatively reproduce a trajectory defined by sparse way-points. In future work, we plan to improve the RL controller, accelerating the convergence in order to learn faster when using the real robot. One of the main drawbacks of the methodology is that the requested trajectories are limited to be similar with the derived demonstrations, e.g. in terms of trajectory direction/flow, which is planned to be improved. 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.

Acknowledgment

This research has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no. 101017054 (project: SoftGrip). This work has also been partially supported by the French Centre National de la Recherche Scientifique (CNRS) PICS international scheme no. 279521.

References

[1] W. H. Organization, *World report on ageing and health*. World Health Organization, 2015.

[2] J. C. Millán-Calenti, J. Tubío, S. Pita-Fernández, I. González-Abraldes, T. Lorenzo, T. Fernández-Arruty, and A. Maseda, "Prevalence of functional disability in activities of daily living (adl), instrumental activities of daily living (iadl) and associated factors, as predictors of morbidity and mortality," *Archives of gerontology and geriatrics*, vol. 50, no. 3, pp. 306–310, 2010.

[3] S. Schaal, "Dynamic movement primitives-a framework for motor control in humans and humanoid robotics," in *Adaptive motion of animals and machines*. Springer, 2006, pp. 261–280.

[4] A. Paraschos, C. Daniel, J. Peters, G. Neumann, *et al.*, "Probabilistic movement primitives," *Advances in neural information processing systems*, 2013.

[5] A. Paraschos, C. Daniel, J. Peters, and G. Neumann, "Using probabilistic movement primitives in robotics," *Autonomous Robots*, vol. 42, no. 3, pp. 529–551, 2018.

[6] C. Laschi, B. Mazzolai, and M. Cianchetti, "Soft robotics: Technologies and systems pushing the boundaries of robot abilities," *Science Robotics*, vol. 1, no. 1, 2016.

[7] F. Renda and C. Laschi, "A general mechanical model for tendon-driven continuum manipulators," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 3813–3818.

[8] A. Grzesiak, R. Becker, and A. Verl, "The bionic handling assistant: a success story of additive manufacturing," *Assembly Automation*, 2011.

[9] 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.

[10] 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.

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

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

[13] 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.

[14] 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.

[15] 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

[16] 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.

[17] P. Oikonomou, M. Khamassi, and C. S. Tzafestas, "Periodic movement learning in a soft-robotic arm," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 4586–4592.

[18] 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.

[19] S. Gomez-Gonzalez, G. Neumann, B. Schölkopf, and J. Peters, "Adaptation and robust learning of probabilistic movement primitives," *IEEE Transactions on Robotics*, vol. 36, no. 2, pp. 366–379, 2020.

[20] D. Nguyen-Tuong and J. Peters, "Model learning for robot control: a survey," *Cognitive processing*, vol. 12, no. 4, pp. 319–340, 2011.

[21] O. Sigaud, C. Salaün, and V. Padois, "On-line regression algorithms for learning mechanical models of robots: a survey," *Robotics and Autonomous Systems*, vol. 59, no. 12, pp. 1115–1129, 2011.

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

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] H. Van Hasselt, "Reinforcement learning in continuous state and action spaces," in *Reinforcement learning*. Springer, 2012, pp. 207–251.

[28] S. Haykin, *Neural networks and learning machines, 3/E*. Pearson Education India, 2010.