Model Based Reinforcement Learning for Closed Loop Dynamic Control of Soft Robotic Manipulators

Thomas George Thuruthel[1*], Egidio Falotico [1], Federico Renda[2], Cecilia Laschi[1]

Abstract— **Dynamic control of soft robotic manipulators is an open problem, yet to be well explored and analyzed. Most of the current applications of soft robotic manipulators utilize static controllers or quasi-dynamic controllers based on kinematic models or linearity in the joint space. However, such approaches are not truly exploiting the rich dynamics of a soft-bodied system. In this paper we present a model based policy learning algorithm for closed-loop predictive control of a soft robotic manipulator. The forward dynamic model is represented using a recurrent neural network. The closed loop policy is derived using trajectory optimization and supervised learning. The approach is verified first on a simulated piecewise constant strain model of a cable driven under-actuated soft manipulator. Further, we experimentally demonstrate on a soft pneumatically actuated manipulator how closed loop control policies can be derived, that can accommodate variable frequency control and unmodeled external loads.**

I. INTRODUCTION

The complexity involved with modeling and control of soft robots is a well known problem which has limited the application of soft robots in real-world scenarios. The widely used method is based on static controllers that rely on the kinematic models of these high dimensional manipulators [1], [2]. However, they rely on the steady state assumption which limits their speed, efficiency and reachability of the manipulator [3], [4]. An alternate approach would be to use dynamic controllers which can plan and exploit the complex dynamics of the system. However, this is not straightforward as the complexity involved with developing an accurate dynamic model is much more than developing a kinematic model.

The earliest usage of a dynamic controller for a continuum robot was executed using a model-free method [5]. The developed closed loop joint space controller was composed of a model-free feedback component based on a continuous asymptotic tracking control strategy for uncertain nonlinear systems and a feed-forward neural network component for compensating for the actuator dynamics. An extended dynamic model for the same variable length multisection manipulator was described in [6]. One of the first model-based approaches, aimed for curvature space control, relied on a dynamic model represented in Euler-Lagrangian form using lumped dynamic parameters [7], [8]. Based on this an experimental evaluation of a sliding mode controller in the configuration space was proposed in [9], however only on a planar manipulator. Along the same lines, a joint space controller that considers the dynamics of pneumatic actuators was proposed in [10]. The control law is based on a decoupled PD computed torque controller.

Another interesting work for the task space control of a completely soft manipulator was described in [11] using a methodically developed dynamic model and trajectory optimization. However all these model-based approaches still rely on the constant curvature approximation which is, theoretically, valid only in the steady state condition. Nonetheless, even with this approximation, dynamic reaching of steady targets was achieved in [11], although only for a planar manipulator and by including an iterative learning scheme.

An implementation of a model predictive controller (MPC) was described in [12] using decoupled dynamic models. Due to the simplicity of the dynamic model and the distinct design of the arm, a MPC could be run at high control frequencies. In [13], the development of a model free dynamic controller directly from the actuator space to task space was proposed. The forward model of the soft manipulator was learned using a recurrent neural network and trajectory optimization was performed to obtain open loop control policies. For more complex dynamic models it is difficult to develop a MPC that can run at the required control cycle for closed loop control as observed in [12] and [13] . One way to solve this would be to directly learn the closed loop control policies. However, learning closed loop policies directly on the real platform is time intensive and prone to falling into poor local optima. On the other hand, learning directly from a simulated model would amplify the inaccuracies already present.

This paper presents an application of model based reinforcement learning for closed loop dynamic control of soft robotic manipulators. The forward dynamic model is learned using a type of recurrent neural network due to the computational cost of analytical models like in [13]. Using the learned model and a single-shooting trajectory optimization algorithm, open

*Corresponding author: t.thuruthel@santannapisa.it

[1] Thomas George Thuruthel, Egidio Falotico and Cecilia Laschi are with the BioRobotics Institute, Scuola Superiore Sant'Anna, Viale Rinaldo Piaggio 34, 56025 Pontedera (Pisa), Italy {t.thuruthel, e.falotico, cecilia.laschi}@santannapisa.it

[2] Federico Renda is with the Department of Mechanical Engineering and the Center for Autonomous Robotics Systems (KUCARS), Khalifa University of Science and Technology, Abu Dhabi, UAE.

loop control policies are sampled on the real platform. The newly obtained experimental trajectories are used to learn, in a supervised way, a closed loop predictive controller using a multilayer perceptron. We test and validate our controller on a simulated tendon driven soft manipulator and experimentally on a two section under-actuated pneumatically driven soft manipulator. To the best of our knowledge this is the first implementation of a closed loop predictive controller for a soft robotic manipulator. The method is easy to apply, requires a short training time and has no exploration phase. Additionally, the learned policies are highly robust to change in control frequency and addition of unmodeled external loads.

II. RELATED WORKS

Direct policy learning for robot control is an effective method for situations where the dynamic modeling is hard or when the environment is unstructured. Alternatively, this approach could be employed in high dimensional systems where traditional model based controllers are not fast enough [14]. The key limitations of policy learning are the high data requirement (for model-free reinforcement learning) or the model bias that deteriorates the performance of the policy (for model-based reinforcement learning) [15]. Additionally for both methods, there are local minima and exploration issues especially for high dimensional policies. Our focus is on model-based reinforcement learning, since it can generate more sample efficient policy learning.

In [16], an algorithm called probabilistic inference for learning control (PILCO) was used for model based policy search. It takes into account the model uncertainties of the learned dynamic model (provided by non-parametric Gaussian process) for long term planning. Recently there has been a substantial interest in using traditional trajectory optimization methods for generating samples for policy learning [17]. Furthermore it could be combined with the function approximation abilities of neural networks to learn and represent these policies [18], [19]. The state of the art approaches using variations of this idea are concerned with using learned local models [20], compound multi-step controllers [21] and deep representation of the control policies [22].

We employ a similar, but simplified, approach like the guided policy search for learning the closed loop control policies. A policy independent trajectory optimization process generates the required samples for policy learning. The samples are obtained from the real system. Finally an offline policy approximation is learned using a simple feedforward neural network.

III. PRELIMINARIES

In this section we give a brief description about the simulation model and experimental setup we are using for this study and the technique for learning the forward dynamics. The simulation platform is primarily used to numerically quantify the robustness of the controller to external disturbances and change in control frequency.

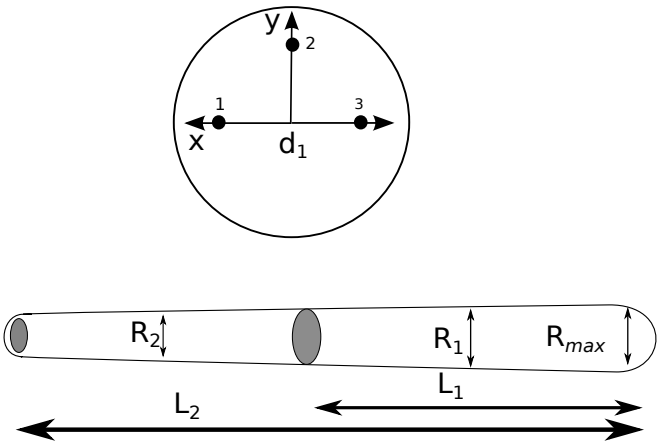

Fig. 1: Schematic of the tendon driven soft manipulator.

A. Piecewise Constant Strain Model

The discrete Cosserat model is a recently developed approach to model continuum, soft manipulator that make use of the intrinsic parametrization of the continuous Cosserat model [23] by discretizing it under the assumption of piecewise-constant deformation along the robot arm [24], [25]. This model has several advantages with respect to the other existing approaches. First of all, while maintaining the computationally efficient piecewise-constant discretization of the more popular constant curvature approach, it allows to model the torsion of the manipulator, as profited in the continuous Cosserat case, which is fundamental to cope with non-negligible external loads. Furthermore, due to its intrinsic parameterization, the discrete Cosserat model shows a uniform geometric structure based on the Special Euclidean group SE(3) that turns out to be a generalization for soft robotics of the fundamental geometric theory of robotics developed since Brocketts [26].

For this study we are using a two section tendon driven soft manipulator. A schematic of the manipulator is shown in Figure 1. The first section is driven by three cables and the second section is passive and the arrangement of the cables is shown in Figure 1. Throughout this paper we will refer to sections without any internal actuation as passive. Some of the important physical parameters used for the simulation are shown in Table I. The geometric parameters in Table I are in reference to Figure 1.

B. Experimental Setup

The controller is also validated experimentally on a two section pneumatically actuated soft manipulator

TABLE I: Parameters of the simulation model.

Parameter	Value
Rmax	15 mm
Rmin	4 mm
L_1	98 mm
L_2	203 mm
Young Modulus	110 $K.Pa$
Shear Viscosity Modulus	300 $Pa.s$
Poisson Ratio	0.5
d_1, d_2, d_3	9 mm
Gravity	$9.81 m/s^2$
Drag Coefficient x	0.01
Drag Coefficient y	2.5
Drag Coefficient z	2.5
Density of Water	1020 kg/m^3
Density of Arm	1080 kg/m^3

[27], [28]. The manipulator is cylindrical with each section having three radially symmetric pneumatic chambers (see Figure 2). The distal section is kept passive. So the manipulator has only three active actuators in the proximal module. The main reason for using pneumatic actuation for the experiments is because we could directly control the forces acting on the manipulator using a simple low-level pressure controller. Tendons on the other hand would require additional sensors for tension measurement and a more complicated low-level controller. In simulations, we are not constrained by this.

The electronic proportional micro regulator Series K8P pressure regulator is used for the low level closed loop control of the chamber pressures. The Vicon tracking system is used to track the five markers attached along the manipulator (Figure 2). These marker positions and velocities are the only state information used for developing the forward model and the control policy. In fact, a good approximation of the dynamic model can be developed from just the distal markers (Refer to the authors previous work [13], [29] for the learning method and the performance of the forward model on the same manipulator). Refer to [30] for details on tracking accuracy of the Vicon system for static and dynamic cases.

C. Forward Dynamic Model

The forward dynamic model which will later be used for trajectory optimization is obtained through machine learning. This is because analytical models are difficult to formulate and develop, or are computationally expensive (as for the case of our simulated model). The learning procedure is based on our previous work on open loop controllers for soft manipulators [13]. A brief introduction is given for clarity of further sections.

The forward dynamic model is represented as a recurrent mapping: $(\tau_i, x_i, x_{i-1}) \rightarrow x_{i+1}$. Here, (x_{i-1}, x_i, x_{i+1}), is the absolute position of manipulator states in the previous, current and next time step. τ_i are the forces applied on the manipulator. It is a discretized representation of the dynamics mapping

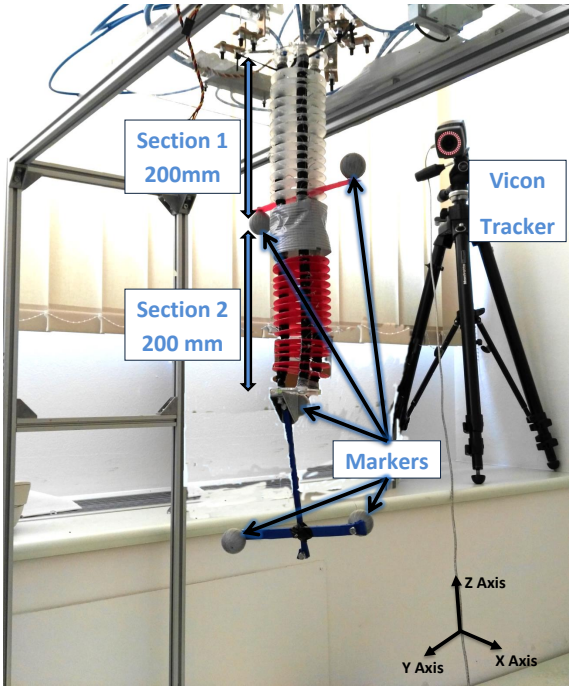

Fig. 2: The pneumatically actuated soft manipulator used for the experiments, Section 1 is pneumatically actuated while Section 2 is completely passive. Three markers are attached to the distal section and two to the proximal one

with fixed time step. This allows us to construct the forward model using a class of dynamic recurrent neural networks called nonlinear autoregressive network with exogenous inputs (NARX). This makes the model more accurate for long term predictions [31].

Sampling is done by motor babbling. For the simulated model, 7000 samples are obtained at 100 Hz. The cable tensions are constrained to a maximum of 3N. The training process is divided into two subparts. Initially the training is performed with the network in open loop mode in order to reduce single step prediction error. The training is accomplished through Bayesian Regularization and an early stopping method is employed. Tan-sigmoid transfer function is used in the input layer and a linear transfer function is used in the output layer. A single layer network composing of 35 neurons was sufficient to learn complete dynamic model of the two-section soft robotic manipulator with 12 degrees of freedom (DoF).

For the real two section soft manipulator, 12000 samples were collected at 50 Hz to learn the forward model. Only information from the last three markers attached to section 2 was sufficient to develop a good approximation of the forward model. The increase in accuracy of the forward model with addition of the base markers was not significant compared to the un-

certainties involved with tracking more markers (Reference [13] provides a detailed analysis on the accuracy of the model with increasing dimension of the state space representation). If the distal section of the manipulator is also actuated, then we would need to also provide the state of the proximal module for learning the forward model. A NARX network with 40 hidden layer units was used to represent the forward model. The training procedure is the same as for the simulated model.

D. Trajectory Optimization

Once we have the forward dynamic model, traditional optimization algorithms can be used to obtain the appropriate control inputs to drive the manipulator to the desired states. In this work, we use a single shooting method for solving the trajectory optimization problem. Given the learned dynamic model, the future states of the system can be predicted with the current control policy:

$$x_{i+1} = f\left(x_i\ , x_{i-1}\ , \tau_i\right) \quad \forall\ i = 0\ ..\ \frac{t_f}{dt} \tag{1}$$

Where, τ_i is the current control input, dt is the step size of the dynamic model (10 ms for the simulations and 20 ms for the real manipulator), t_f is the control horizon and the function f represents the learned forward model. The open loop control policy is given by:

$$\Pi(t) = \tau_i^m \qquad \forall\ \ m = 1\ ..\ M \tag{2}$$
$$i = \left\lfloor \frac{t}{dt} \right\rfloor \qquad \forall\ t = 0\ ..\ t_f$$

Where M is the number of actuators (three for both cases). For dynamic reaching tasks, the objective function tries to reduce the end-effector tracking error at the end of the control horizon. Additionally, a term to minimize the control effort is also added. The input weighting coefficient R is an identity matrix in our case. The optimal policy then can be obtained by optimizing the following nonlinear optimization problem:

$$\Pi(t)^* = \min_{\tau}\left(\left\| x^{task}_{\frac{t_f}{dt}} - x^{des} \right\|^2 + \sum_i \tau_i^T R\ \tau_i \right) \tag{3}$$

$$\text{subject to } 0 \le \tau_i^m \le \tau_{max}^m \quad \forall\ \ m = 1\ ..\ M \text{ and } i = \left\lfloor \frac{t}{dt} \right\rfloor$$

We use the iterative sequential quadratic programming (SQP) algorithm for solving the optimization problem. The obtained optimal control policy, however, works only in open loop and depends on the initial conditions. Therefore, it cannot accommodate modeling errors, external disturbances during execution of the policy or different initial conditions. Moreover, the optimization process is not fast enough to implement a closed loop model predictive controller, even with the computationally efficient learned model. In order to obtain a closed form policies for the dynamic tracking problem we employ a variant of the recently develop technique called guided policy search [24], where trajectory optimization is used to direct policy learning and avoid poor local optima.

IV. Policy Learning

In order to develop a closed loop optimal control policy, we would need the optimal control actions for each reachable state of the manipulator. Methods like guided policy search [17] tries to learn the policies directly and uses trajectory optimization to generate samples for policy learning. The policies can then be represented using any function approximation methods.

For this the optimization problem can be reformulated to obtain multiple trajectories to the same reaching task. This ensures that the manipulator state space is fairly explored and serves as samples for the policy learning. The trajectories provide samples for the appropriate control action for each region in the manipulator state space. To ensure that multiple solutions to the trajectory optimization problem exists, the control horizon has to be kept long enough. Setting the control horizon too long could result in obtaining redundant solutions for the same system state. For our case, we determine this period empirically.

Given the number of trajectories to be generated N and number of targets to be reached P, the objective function can be modified as:

$$\Pi_n{}^p(t)^* =$$
$$\min_{\tau}\left(\left\| x^{tip}_{\frac{t_f}{dt}} - x^{des} \right\|^2 - \alpha \min\left[dist(X^n, X) .* dist(X^n, X) \right] \right)$$
$$X^n \triangleq \left\{ x_1^n, x_2^n,\ \ldots\ x^n_{\frac{t}{dt}-k} \right\},\ X \triangleq \left\{ X^1, X^2, \ldots\ X^{n-1} \right\}$$
$$\forall\ \ n = 1\ ..\ N\ \ \forall\ \ p = 1\ ..\ P \tag{4}$$

The distance function calculates the Euclidean distance from each elements of X^n to the corresponding elements in X. By maximizing the minimum distance (with $-\alpha$), every new trajectory generated is unique and tries to span a larger state space. By increasing value of α, more varied trajectories can be obtained. The constant k specifies a temporal region where the uniqueness of the trajectory is measured .It is used to make sure that the target position is reached at the end of the control horizon. x^{tip} is the end effector position coordinates and x^{des} is the desired end effector position.

The generated open loop policies are executed on the real platform/simulation model to generate samples for training the policy. This would reduce model biases from transferring to the policy. Now with the N trajectories obtained, a simple supervised learning model can be employed to directly learn the appropriate control for each system state.We are using feedforward neural networks to represent the closed loop policies. An

Algorithm 1: Learning closed loop predictive control policies

Generate samples $(\tau, x_{i-1}, x_i, x_{i+1})$ for learning dynamic model

Learn the mapping $(\tau, x_i, x_{i-1}) \rightarrow x_{i+1}$ by a NARX network to generate the forward dynamic model

for $i \leftarrow 0$ **to** P **do**

  Generate N different trajectories to reach the target ;

Learn closed loop policy by learning the mapping: $(x_i, x_{i-1}, x^d es) \rightarrow \tau_i$

Result: policy $\pi(x_i, x_{i-1}, x^{des})$

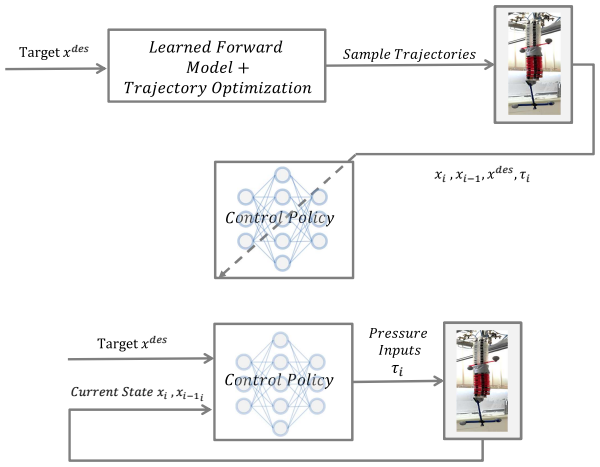

Fig. 3: Block diagram describing the complete procedure for obtaining the closed loop control policy (top). The learned control policy is encoded by a feedforward neural network and provided the appropriate closed loop actions (bottom).

exhaustive search in the state space is not necessary due to the generalizing ability of neural networks. This was validated with the experimental results (See section VI).

The predictive controller is represented using the mapping: $(x_i, x_{i-1}, x^{des}) \rightarrow \tau_i$. This discretized representation of the system state is important to make the control policy robust to the control frequency. It is even possible to reduce the dimensionality of the state space and get a computationally less complex policy but with lower accuracy. For this work, we reduce the dimension of our state space from twenty four (simulation/learned dynamic model) to twelve (for the policy) to generate faster solutions. No changes are made for the experimental case since the forward model itself is low dimensional. Adding multiple targets to the same policy not only provides us a global policy but can also serve as samples among targets. Subsequently we have a dynamic end-to-end policy directly relating the system states to the actuator inputs for a particular target. This is computationally faster since we do not need the optimization step after the offline policy learning. Note that this method does not explicitly take into account the control horizon. Therefore, in the presence of external disturbances it is not possible to predict when the manipulator would reach the desired target positions. It is important to keep the control horizon as short as possible to ensure that the policy mapping is unique.This condition is not valid in the origin, since the same manipulator state and target can have multiple 'correct' actions. However, even a blind averaging of all the possible actions can lead to the desired motions as observed.

The complete procedure for developing the closed loop predictive controller is described in Algorithm 1. The corresponding control architecture is shown in Figure 3 For the two section simulated manipulator, we generate 20 trajectories to reach 65 randomly selected targets from the workspace. The workspace is obtained from the previous motor babbling process. The control horizon is fixed to 1s for each target and the optimization is set to 30 iterations. An example of different trajectories generated for an α value of 0.01 is shown in Figure 4. The policy is represented using a multilayer perceptron with a hidden layer size of 30 units.

For the real manipulator, single trajectories to reach 200 randomly selected targets were used.Since the forward model is less accurate than the simulated case, addition of the unique trajectory generation part led to significant accuracy depreciation. Therefore we opted to span the state space using varied targets instead. The control horizon is fixed to 4s for each target and the optimization is set to 30 iterations.The targets are picked randomly from the dynamic workspace obtained for the forward model. The multilayer perceptron had a hidden layer size of 40 units. The input layer has a tan-sigmoid transfer function and the output layer has a linear transfer function. Training is again done with Bayesian Regularization and early stopping method is employed. The generation of a new policy iteration takes 10.9±0.18 ms and 12.5±0.44 ms for the simulated and real model respectively. All the computation is performed on an Intel(R) Core(TM) i7-3630QM CPU @ 2.40 GHz and 8 Gb RAM.

V. Simulation Results

Since our objective is to develop a global closed loop predictive controller, we conduct four simulation studies to investigate the validity of the learned control policy. For all the tests the control policy is first executed on the learned dynamic model and the derived control actions are transferred to the simulation model.

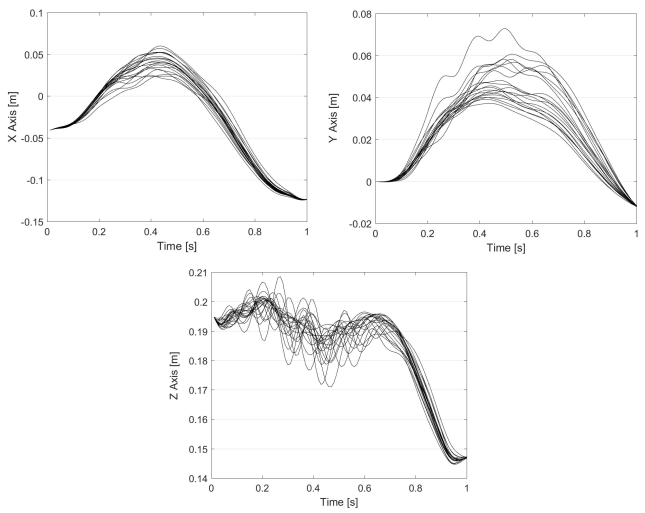

Fig. 4: End effector position for twenty unique trajectories generated by the trajectory optimization algorithm for an example target point.

A. Global Dynamic Reaching

In order to validate the ability of the controller to dynamically reach static targets in the workspace, a tracking task is performed to assess the accuracy of the controller. Fifty random targets, different from the set used for learning the policy, have been selected for testing. The reaching performance of the controller is then evaluated in closed loop using the learned dynamic model. Each tasks last 2 seconds.

After obtaining the control actions for a single task, the same control inputs are provided to the simulation. Note that the policies are derived from the learned model and therefore, we can expect some variations from the simulated model. The error in the reaching task at the expected reaching time (1 second) is shown in Table II for both the learned and simulated model. Since the controller is run without any external disturbances and starting from the home state, the reaching time is consistent. In the presence of external disturbances or different initial states, the reaching time cannot be estimated before hand. It must also brought to the attention of the reader that the target points are not reachable statically and hence the manipulator cannot stop once it has reached its desired location. Therefore the control policy will continue to drive the motion of the manipulator towards the target repeatedly and not necessarily in the same trajectory.

TABLE II: Global tracking performance.

Model	Tracking Error[m]
NARX	0.012±0.010
Simulation	0.013±0.010

B. Reaching with external disturbances

The need for a closed loop policy is more important in the presence of heavy external disturbances which would be considerable in the environments where soft robots are meant to be deployed. To demonstrate that our proposed policy learning approach is robust to heavy external disturbances, we formulate a reaching task with varying initial external disturbances. Fifty random targets are selected at random for the reaching task and varying simulated disturbances are added during the initial $0.5s$ of the task. A folded normal distribution noise is added to the inputs with the variance of the normal distribution increasing from 0 to $9N^2$, amounting to a total of 10 trials. Note that the learned dynamic model is obtained by motor babbling with actuators inputs constrained to $3N$. Since the disturbance is for a short period, this error does not accumulate to adversely affect the controller. For each trial, the controller is run for $2s$ and the time to reach the closest position near the target is recorded along with the error.

The trajectory taken by the end-effector for a single target is shown in Figure 6. Due to the generalization abilities of the neural network and the global coverage of the control policy the controller is highly robust to disturbances and can even generate policies that are unique from the initial samples generated in the policy learning phase. The reaching time is also not adversely affected, considering the fact that during the initial $0.5s$, the underlying control strategy is affected. The effect of varying noise on the performance of the controller can be seen in Figure 5. It can be observed that the simulated model has similar accuracy compared to the learned model, even though the control policy is derived using the learned model.

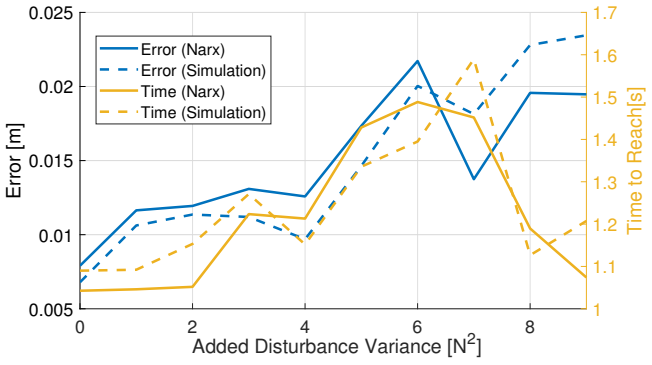

Fig. 5: Reaching error versus external noise variance. Note that the variance value is of the normal distribution before taking its absolute value.

C. Multi-Point Reaching

Although, the trajectory optimization procedure for generating samples for learning the policy is always initiated from the starting point, the controller is not

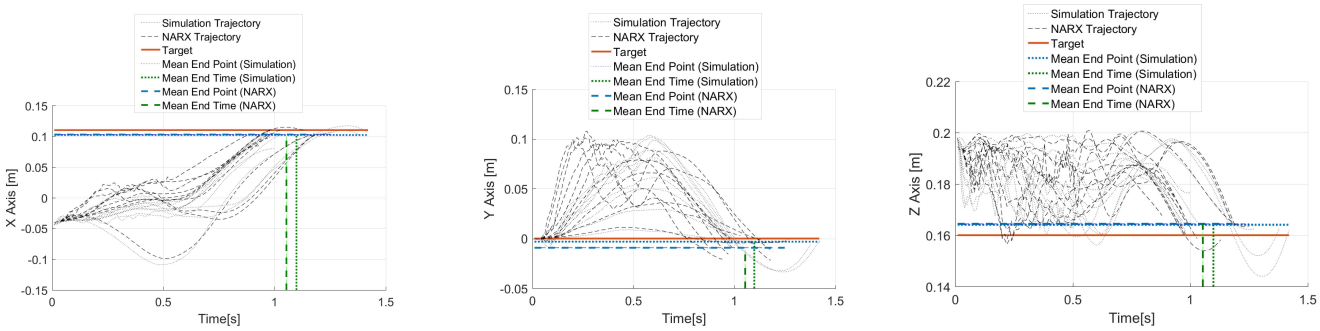

Fig. 6: End- effector trajectory for varying external disturbances during the reaching task. External disturbances are added only for the initial 0.5*s*.

specifically tuned to this initial condition. Since the control policy is dependent only on the current state of the manipulator and not on the previous states, it is possible to reach any desired stationary target in the workspace from any given manipulator state as long as the controller has sufficient time. This is of course based on the assumption that we have obtained adequately spaced samples during the trajectory optimization step.

To demonstrate this we setup a simulation where the manipulator is asked to track two target points successively. The two targets are selected randomly and 50 trials have been conducted using new targets each trial. The first target is provided as input to the policy for only 1*s* and the second target is given for the next 2*s*. The tracking performance is summarized in Table III and an example trial is shown in Figure 7. The tracking error for the first target is slightly lower than the first reaching task since we are now evaluating the lowest distance from target and the average reaching time. The second target needs more time to reach, with a small increase in error. This performance is consistent with the reaching-with-disturbance task.

TABLE III: Tracking performance for multi-point reaching task.

Model	**Performance**			
	Target 1		**Target 2**	
	Error[m]	***Time[s]***	***Error[m]***	***Time[s]***
Narx	0.011±0.011	0.98±0.07	0.012±0.015	2.46±0.29
Sim.	0.011±0.011	0.96±0.07	0.014±0.014	2.39±0.20

D. Variable control frequency

An interesting property of the learned closed loop policy was the observed robustness to the control frequency. Although the control policy is learned from samples collected at 100Hz, the same policy can be run even at lower frequency. This was observed even for the experimental results (See section VI). This means that underlying control policy must be requiring only sparse state space information. This is aided by the fact that soft systems are low bandwidth systems with slow

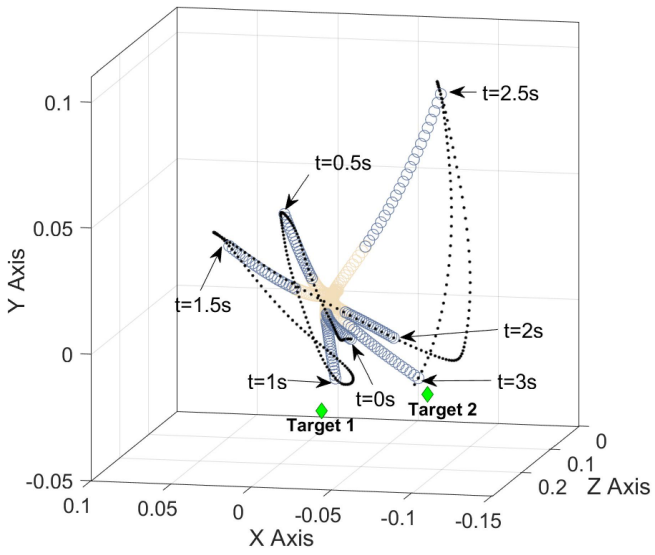

Fig. 7: Manipulator configurations in the multi-point reaching task for an example trial. The end-effector trajectory is shown in black.

system response. Later it will also be show that this can be extended to case where additional load is attached to the end-effector (Section VI-C).

To see how the change in control frequency affects the learned policy we do some preliminary tests on the learned forward model. The forward model is always run at 100Hz, while the controller frequency is varied from 100Hz-5Hz. This means that the inputs to the policy are also delayed by the same amount as the control frequency. The change in tracking error and reaching time with the control frequency is shown in Table IV, for fifty random targets. There is no added noise to the system and therefore, we can see the original policy reaching the desired targets at the expected time. The distance from the target with time for an example case is shown in Figure 8. The corresponding input signals are shown in Figure 9.At lower frequencies it can be seen that the control policy approximates to a bang-bang controller. The simulation results indicate that the

varying the control frequency does not adversely affect the performance of the controller. Although the tracking error and reaching time is not significantly affected by changing the control frequency, the obtained control inputs loose their smoothness, which would have more significance in the real world.

TABLE IV: Controller performance with changing control frequency

Frequency	Performance	
	Tracking Error[m]	*Reaching Time[s]*
100 Hz	0.0085±0.007	0.995±0.094
50 Hz	0.0080±0.005	1.040±0.117
33.3 Hz	0.0097±0.008	1.135±0.130
20 Hz	0.0134±0.010	1.387±0.270
10 Hz	0.0254±0.017	1.559±0.321
5 Hz	0.0226±0.011	1.85±0.252

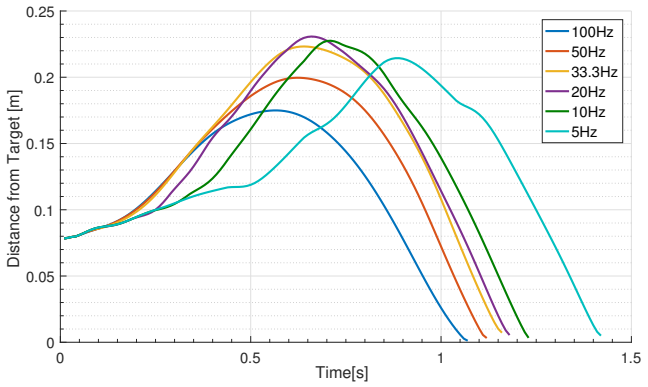

Fig. 8: The distance of the end-effector from the target with varying control frequency.

VI. Experimental Results

The experiments are conducted on the two section pneumatic manipulator with the passive distal section. The closed loop policy is learned from samples collected at 50 Hz as described in section IV. However due to the overhead computational cost for acquiring sensors values and the communication between arduino and the MATLAB environment the highest control frequency we would achieve was at 20 Hz. The dynamic workspace of the soft manipulator obtained by motor babbling is shown in Figure 10 along with the static workspace boundaries. The static workspace is obtained by actuating each chambers to the maximum pressure and letting the manipulator settle. The static workspace becomes almost negligible with the addition of load to the end effector (check the supplementary video attachment). We show in section VI-C, how the learned policy is robust to even such drastic changes. All the experimental plots are shown only in the XY plane. This is because we observed that the motion of the end effector largely progresses along the surface of the ellipsoid. Therefore the manipulator can only reach a particular position in XY plane within a small range in the Z axis coordinate. All the quantitative results are given in three dimensional space.

A. Global Dynamic Reaching

The reaching performance of the controller is summarized in table V. An example case of how the trajectories evolve for two different targets are shown in figure 11. Note that for targets that are along the direction of actuation, a more linear reaching behavior is observed while targets that are not along the direction of actuation adopts a circular trajectory to reach. These trajectories are dictated by the design and the actuation of the manipulator.

The evaluation period is defined as the time period after the initialization of the control policy in which the reaching error is evaluated. It is clear that with a larger window of the evaluation period, the tracking error is lower but with higher variances in the reaching time. Without the presence of external disturbances, sensory losses and the appropriate control frequency the expected time of reaching is 4 seconds for all points in the workspace. Due to the reduction in control frequency (20Hz) with respect from the prescribed value (50Hz), there is a shift in the average reaching time. This behavior is similar to what we observed for the simulation case. However, the reaching time has more uncertainties than the simulation case. This could also indicate that there are significant stochastic factors in the dynamics of the system. A simple way to confirm this is to observe the behavior of the controller for reaching the same target from the same initial configuration as shown in Figure 12. Indeed, the variability in motion is very high even from the starting of the controller. A possible explanation could be highly nonlinear and stochastic friction effects that we incur due to the external braided structure. Hysteresis effects due to the soft chambers could be another factor. This is also reflected by the variability in the home (starting) position (Table VI). In the simulated model the trajectories would be exactly matching, since there is no source of variability.

TABLE V: Tracking performance.

	Performance	
Evaluation Period	*Tracking Error[m]*	*Reaching Time[s]*
3-6 seconds	0.017±0.014	4.3±0.77
3-8 seconds	0.009±0.008	5.1±1.46

TABLE VI: Variability in the home position.

	Range[mm]	Std.[mm]
X	6.8	1.1
Y	8.3	1.7
Z	2.2	0.3

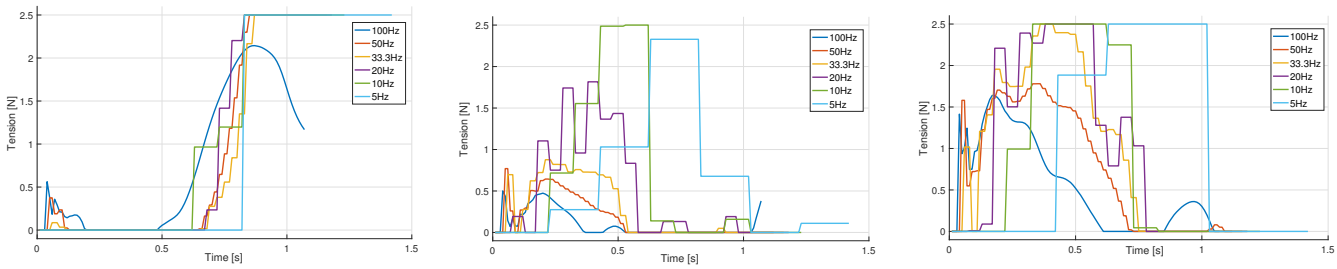

Fig. 9: Input forces from the three tendon actuators with varying control frequency for the simulation.

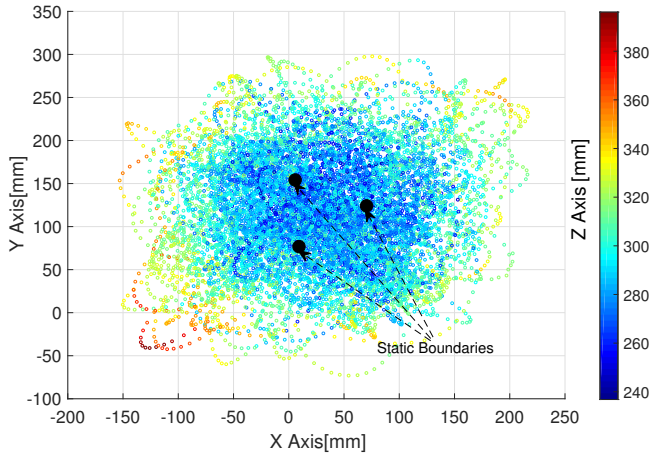

Fig. 10: The dynamic workspace of the manipulator compared to the static boundaries.

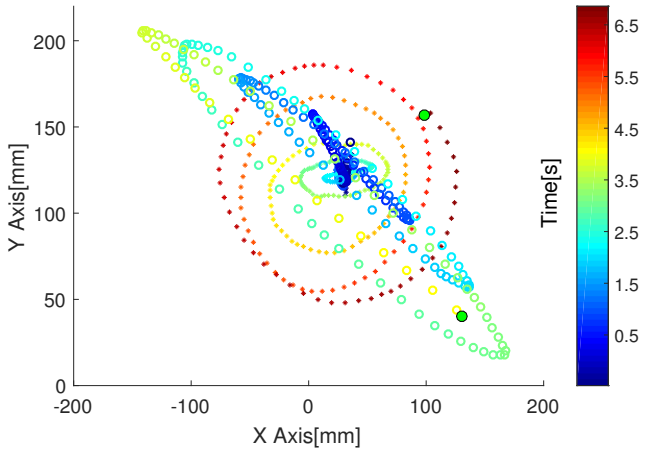

Fig. 11: The trajectory of the end-effector generated to reach two example targets using the proposed controller.

B. Low Frequency reaching

The derived closed loop policy also exhibited robustness to the control frequency for the experimental tests. For this we add fixed delays to the control loop to reduce the control frequency. The controller performance for a 10 Hz controller is shown in table VII. Similar to the simulation results we observe an increase in the tracking error with a shifted expected reaching time. There are also instances where the targets are not reachable at lower control frequencies (Figure 13).

TABLE VII: Tracking performance with reduced control frequency .

	Performance	
Evaluation Period	*Tracking Error[m]*	*Reaching Time[s]*
3-8 seconds	0.026±0.032	5.9±1.28

C. Reaching with load

Another advantage we obtain with our proposed direct policy learning method is the ability to accommodate changes in manipulator dynamics itself. We demonstrate this by attaching a 105 grams load to the tip of the manipulator. Even without any adaptation phase the initial closed loop policy is able to perform the reaching task. The tracking error and reaching time

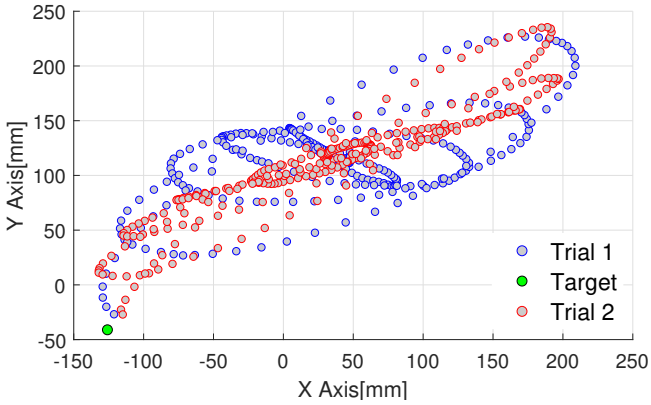

Fig. 12: Variability of the trajectories in reaching the same target without any external disturbances.

is given in table VIII. The reaching time is significantly increased as expected. This is because the controller needs several 'energy pumping' phases to provide enough kinetic energy to the system (See Figure 14 for an example). It is also noteworthy that the added mass is not symmetric. This ensures a disproportionate modification of the manipulator dynamics. Therefore the ensuing reaching motion has a skewness associated with its motion.

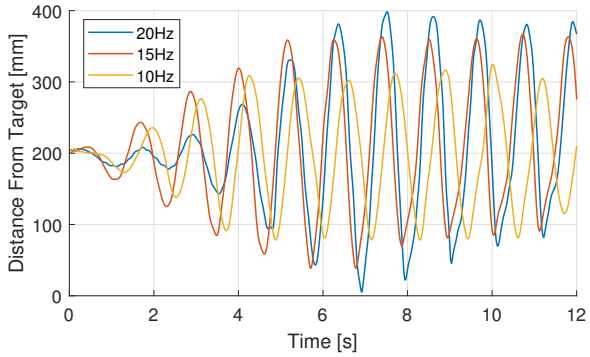

Fig. 13: Reaching error evolution with varying frequency for a target at the dynamic workspace boundary. For this case, timing becomes crucial and hence at low frequencies the target cannot be reached.

Although soft robots have intrinsic compliance that cannot be emulated through control approaches [32], it does not necessarily make them safe. Given sufficient time, our experiments show that soft robotic manipulators can build up significant momentum by storing energy in their complaint elements. Due to the absence of powerful internal actuators, sudden changes in the direction of motion is not achievable. Gravitational forces play the major role for changing the momentum of the load in our case. The velocity of the end-effector during the reaching task with the added load is shown in Figure 15 for reference.

TABLE VIII: Tracking performance with added load.

	Performance	
Evaluation Period	*Tracking Error[m]*	*Reaching Time[s]*
10-20 seconds	0.022±0.022	15.5±3

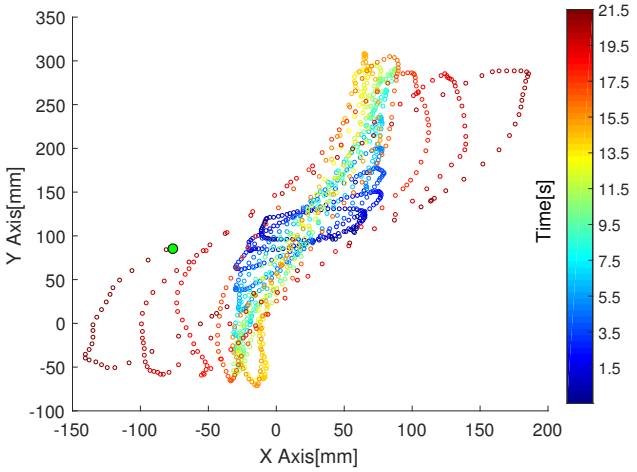

Fig. 14: The trajectory of the end-effector with added load. Note the increase in reaching time and skewness in the trajectory.

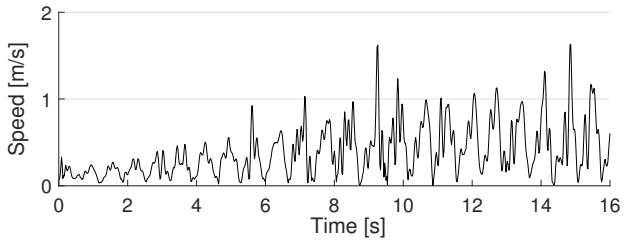

Fig. 15: Velocity of the end-effector for an example case with the added load.

VII. Conclusion and Discussions

This paper presents a direct policy method for closed loop dynamic control of a soft robotic manipulator. This purely data driven approach consists of a three stages; learning the forward dynamic model, generating trajectories as samples for the policy and the final policy learning phase. This way it is possible to directly learn closed loop control policies without the need of an analytical model while being data efficient. For the experiments, the forward model required a sampling period of 240 seconds while the closed loop policy required an additional 8000 seconds. Hence the approach only requires real-world data for approximately 2 hours to develop a closed loop controller from scratch. Moreover, due to the representation of the policy architecture, the derived controller can accommodate changes in control frequency, sensory noise and dynamic changes. For the simulation and experimental case, reasonable accuracy could be maintained up to a five fold decrease in control frequency. This is because the underlying policy is represented like a MPC framework making it more robust to unmodeled factors.

The dynamically reachable workspace is strikingly dissimilar from the static/kinematic workspace. This has its own utility and disadvantages. For one, it allows us to expand the reachable workspace even if external loads are added (see multimedia material), however, it becomes impossible for the manipulator to stop at the desired targets. An interesting solution would be to use variable stiffness mechanisms [33] to freeze the manipulator once the target is reached.The experimental results showed higher variability in the trajectories and reaching time. This could be attributed to stochastic factors affecting the dynamics of the system like friction and hysteresis. This could be improved by better design methodologies and material selection. The draw back of our data driven approach is that it does not impart any insights into the relation between manipulator deign and dynamics. So it becomes difficult to identify the sources of modeling error or in developing optimal design strategies [30].

The feedback control strategy we employ and demonstrated for dynamic reaching tasks is suited

for dynamically grabbing and placing static objects of unknown masses. In such cases scenarios timing is not as important as accuracy, robustness and conformance to the environment. Other tasks would have to adopt different control strategies. For trajectory following, purely feedforward strategies has proven to provide stable motion [29]. Feedforward strategies are more desirable for energy efficient motion without affecting natural dynamics of the system . In fact it was shown that higher feedback gains leads to higher perceived stiffness [34]. Feedback controllers become more important in presence of external disturbances or when the manipulator interacts with the environment [35].

VIII. Acknowledgments

The authors would like to acknowledge the support by the European Commission through the I-SUPPORT project (HORIZON 2020 PHC-19, #643666) and the European Unions Horizon 2020 Research and Innovation Program under Grant Agreement No. 785907 (HBP SGA2).

References

[1] 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. [Online]. Available: http://robotics.sciencemag.org/content/1/1/eaah3690

[2] 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.

[3] T. George Thuruthel, E. Falotico, M. Manti, A. Pratesi, M. Cianchetti, and C. Laschi, "Learning closed loop kinematic controllers for continuum manipulators in unstructured environments," *Soft robotics*, vol. 4, no. 3, pp. 285–296, 2017.

[4] A. D. Marchese and D. Rus, "Design, kinematics, and control of a soft spatial fluidic elastomer manipulator," *The International Journal of Robotics Research*, vol. 35, no. 7, pp. 840–869, 2016.

[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] 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.

[7] A. D. Kapadia, I. D. Walker, D. M. Dawson, and E. Tatlicioglu, "A model-based sliding mode controller for extensible continuum robots," in *Proceedings of the 9th WSEAS international conference on Signal processing, robotics and automation*. World Scientific and Engineering Academy and Society (WSEAS), 2010, pp. 113–120.

[8] A. Kapadia and I. D. Walker, "Task-space control of extensible continuum manipulators," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*. IEEE, 2011, pp. 1087–1092.

[9] A. D. Kapadia, K. E. Fry, and I. D. Walker, "Empirical investigation of closed-loop control of extensible continuum manipulators," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. IEEE, 2014, pp. 329–335.

[10] 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, 2017.

[11] A. D. Marchese, R. Tedrake, and D. Rus, "Dynamics and trajectory optimization for a soft spatial fluidic elastomer manipulator," *The International Journal of Robotics Research*, vol. 35, no. 8, pp. 1000–1019, 2016.

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

[13] T. G. Thuruthel, E. Falotico, F. Renda, and C. Laschi, "Learning dynamic models for open loop predictive control of soft robotic manipulators," *Bioinspiration & Biomimetics*, 2017.

[14] J. Peters and S. Schaal, "Reinforcement learning of motor skills with policy gradients," *Neural networks*, vol. 21, no. 4, pp. 682–697, 2008.

[15] C. G. Atkeson and J. C. Santamaria, "A comparison of direct and model-based reinforcement learning," in *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, vol. 4. IEEE, 1997, pp. 3557–3564.

[16] M. P. Deisenroth, C. E. Rasmussen, and D. Fox, "Learning to control a low-cost manipulator using data-efficient reinforcement learning," 2011.

[17] S. Levine and V. Koltun, "Guided policy search," in *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, 2013, pp. 1–9.

[18] S. Levine and P. Abbeel, "Learning neural network policies with guided policy search under unknown dynamics," in *Advances in Neural Information Processing Systems*, 2014, pp. 1071–1079.

[19] I. Mordatch and E. Todorov, "Combining the benefits of function approximation and trajectory optimization." in *Robotics: Science and Systems*, 2014.

[20] V. Kumar, E. Todorov, and S. Levine, "Optimal control with learned local models: Application to dexterous manipulation," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 2016, pp. 378–383.

[21] W. Han, S. Levine, and P. Abbeel, "Learning compound multi-step controllers under unknown dynamics," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 6435–6442.

[22] T. Zhang, G. Kahn, S. Levine, and P. Abbeel, "Learning deep control policies for autonomous aerial vehicles with mpc-guided policy search," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 2016, pp. 528–535.

[23] F. Renda, M. Giorelli, M. Calisti, M. Cianchetti, and C. Laschi, "Dynamic model of a multibending soft robot arm driven by cables," *IEEE Transactions on Robotics*, vol. 30, no. 5, pp. 1109–1122, 2014.

[24] F. Renda, M. Cianchetti, H. Abidi, J. Dias, and L. Seneviratne, "Screw-based modeling of soft manipulators with tendon and fluidic actuation," *Journal of Mechanisms and Robotics*, vol. 9, no. 4, p. 041012, 2017.

[25] F. Renda, F. Boyer, J. Dias, and L. Seneviratne, "Discrete cosserat approach for multi-section soft manipulators dynamics," *IEEE Transactions on Robotics, (in press)*.

[26] R. W. Brockett, "Robotic manipulators and the product of exponentials formula," in *Mathematical theory of networks and systems*. Springer, 1984, pp. 120–129.

[27] 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. [Online]. Available: https://doi.org/10.1177/1729881416687132

[28] 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.

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

[30] P. Merriaux, Y. Dupuis, R. Boutteau, P. Vasseur, and X. Savatier, "A study of vicon system positioning performance," *Sensors*, vol. 17, no. 7, p. 1591, 2017.

[31] J. M. P. Menezes Jr and G. A. Barreto, "Long-term time series prediction with the narx network: An empirical evaluation," *Neurocomputing*, vol. 71, no. 16-18, pp. 3335–3343, 2008.

[32] A. Bicchi and G. Tonietti, "Design, realization and control of soft robot arms for intrinsically safe interaction with humans,"

in *IARP/RAS Workshop on Technical Challenges for Dependable Robots in Human Environments*, 2002, pp. 79–87.

[33] M. Manti, V. Cacucciolo, and M. Cianchetti, "Stiffening in soft robotics: A review of the state of the art," *IEEE Robotics & Automation Magazine*, vol. 23, no. 3, pp. 93–106, 2016.

[34] C. Della Santina, M. Bianchi, G. Grioli, F. Angelini, M. Catalano, M. Garabini, and A. Bicchi, "Controlling soft robots: balancing feedback and feedforward elements," *IEEE Robotics & Automation Magazine*, vol. 24, no. 3, pp. 75–83, 2017.

[35] C. Della Santina, R. K. Katzschmann, A. Bicchi, and D. Rus, "Dynamic control of soft robots interacting with the environment," 2018.