PAPER

Learning dynamic models for open loop predictive control of soft robotic manipulators

To cite this article: Thomas George Thuruthel *et al* 2017 *Bioinspir. Biomim.* **12** 066003

Bioinspiration & Biomimetics

PAPER

Learning dynamic models for open loop predictive control of soft robotic manipulators

RECEIVED
28 February 2017

REVISED
20 July 2017

ACCEPTED FOR PUBLICATION
2 August 2017

PUBLISHED
6 October 2017

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

[1] The Biorobotics Institute, Scuola Superiore Sant'Anna, Pisa, Italy
[2] Department of Mechanical Engineering, Khalifa University of Science and Technology, Abu Dhabi, United Arab Emirates

E-mail: t.thuruthel@santannapisa.it, e.falotico@santannapisa.it, cecilia.laschi@santannapisa.it and federico.renda@kustar.ac.ae

Keywords: soft robot, control, dynamics, trajectory optimization, recurrent neural network, machine learning

Supplementary material for this article is available online

Abstract

The soft capabilities of biological appendages like the arms of *Octopus vulgaris* and elephants' trunks have inspired roboticists to develop their robotic equivalents. Although there have been considerable efforts to replicate their morphology and behavior patterns, we are still lagging behind in replicating the dexterity and efficiency of these biological systems. This is mostly due to the lack of development and application of dynamic controllers on these robots which could exploit the morphological properties that a soft-bodied manipulator possesses. The complexity of these high-dimensional nonlinear systems has deterred the application of traditional model-based approaches. This paper provides a machine learning-based approach for the development of dynamic models for a soft robotic manipulator and a trajectory optimization method for predictive control of the manipulator in task space. To the best of our knowledge this is the first demonstration of a learned dynamic model and a derived task space controller for a soft robotic manipulator. The validation of the controller is carried out on an octopus-inspired soft manipulator simulation derived from a piecewise constant strain approximation and then experimentally on a pneumatically actuated soft manipulator. The results indicate that such an approach is promising for developing fast and accurate dynamic models for soft robotic manipulators while being applicable on a wide range of soft manipulators.

1. Introduction

The development of dynamic controllers for soft robotics applications is a relatively unexplored area, even with the rising progress and applications of soft robotics technologies. This is mainly because of the complexity involved in developing dynamic models and their resultant controllers. As a result current controllers developed for soft robotic manipulators are largely static controllers which do not completely exploit the capabilities of the manipulator. The majority of dynamic controllers developed for soft/continuum manipulators are model-based approaches which rely on the development of analytical kinematic models and their corresponding dynamic model. Considering the fact that accurate kinematic models are difficult to develop themselves, dynamic models based on these kinematic models are even more complex and error prone (refer to [38] for an example on the inadequacies of traditional kinematic models for developing dynamic models). Even if exact kinematic and dynamic models are available, controllers based on these would require high dimensional sensory feedback [1]. Moreover, there is the problem of parameter estimation which would require proper excitation of the dynamic system and highly robust and well thought out optimization techniques. Although the same applies to learned models, the parameter estimation process is not constrained by predefined models and assumptions.

Due to the complexity involved with the development of dynamic models for continuum/soft manipulators, the earliest dynamic controller was based on a model-free method [2]. The objective of the work was only for closed loop control in the joint space. The control architecture was composed of a model-free feedback component based on a continuous asymptotic tracking control strategy for uncertain nonlinear systems [3] and a feedforward component made using neural networks. The feedforward component was used to compensate for the dynamic uncertainties, thereby reducing the uncertainty bound that

improves the performance of the feedback controller. One of the first model-based approach for dynamics controllers used constant curvature (CC) kinematics with the dynamic model in the configuration space being represented in the Euler–Lagrangian form using lumped dynamic parameters [4, 5]. Building on this work, an experimental evaluation of a sliding mode controller for closed loop configuration space control was proposed in [6], but only on a planar manipulator. Comparisons with a simple feedback linearization-based proportional–derivative (PD) controller in the configuration space were also conducted and it was observed that the sliding mode controller performed better in terms of accuracy and speed, indicating that model uncertainties were significant. The main disadvantage of developing a controller in the joint space or configuration space is an additional error component that arises from the joint/configuration space to task space transformation. Therefore these methods cannot guarantee convergence of the task space error.

The first dynamic controller for a completely soft planar manipulator was proposed in [7] using a trajectory optimization scheme for developing open loop task space controllers. Using the CC model, a methodically developed dynamic model in the configuration space was derived and estimated. A direct collocation approach was employed to simultaneously identify the optimal generalized torques and corresponding manipulator state with the systems kinematics, dynamics, boundary conditions and tracking objective as constraints. However, the dependence of the controller on the analytical model and parameter estimation technique is quite evident as an additional iterative learning scheme was required to re-identify the system parameters in between trials.

Similar to [4], a joint space controller that considers the dynamics of the pneumatic actuators was proposed in [8, 9]. The control law is based on a decoupled PD-computed torque controller. Again, the kinematic model is still based on the CC approximation. Although a promising method, the dynamic model is highly simplified with lumped parameters and ignores rotational energies. The generation of feasible actuator trajectories that can track a desired end effector position is also not so trivial. In addition, the performance of a closed loop controller in the joint space is questionable at higher frequencies due to the slow response of soft systems. Other interesting work using model predictive controllers was described in [10]. Due to the unique manipulator design the kinematics and the dynamics of that system could be represented just like a rigid robot. The advantage of solving the control problem as an optimization problem is that it alleviates the need for a high-level path planner. Modeling and control design for a similar design using data for system identification was presented in [24].

Most of the literature that we have presented deals with joint space control—so these approaches deal with dynamic control of joint configuration (usually cable lengths). All these methods then assume a fixed relation between the joint space and the task space (typically using CC approximation) without any dynamic coupling effects. This paper proposes a methodology for learning forward dynamic models of a soft manipulator and then using this learned model for developing open loop predictive control policies. The main purpose of this research is to show that machine learning-based approaches provide numerous advantages over traditional model-based approaches specifically for soft robotic manipulators. Using simulations of a cable-driven under-actuated conical soft manipulator based on the piecewise constant strain (PCS) approximation, we try to demonstrate that learning-based controllers are easier to develop, apply and transfer while being accurate at the same time. To corroborate this, we provide experimental results on a single-section robot, to showcase the generality, accuracy and the simplicity of the approach.

1.1. Similar works on rigid robots

Contrary to the case of soft robots, learning forward or inverse dynamics of rigid robots have been fairly well studied. Learning inverse dynamics is the most common among the two as it would need only single step-ahead predictions. Local Gaussian process regression and locally weighted projection regression for real-time online model learning [11] and local online support vector regression [12] are among some of the many approaches investigated. Inverse dynamics-based controllers would not need long step-ahead prediction accuracy; however, there is an additional burden in terms of specifying the desired state trajectories. Considering the response delay of soft systems, and taking a cue from biological systems [13, 14], predictive controllers using forward models should fare better.

Learning forward models for rigid robots have also been keenly investigated. Most of the algorithms are concerned with the computational time, online learning abilities and high-dimensional systems. Some of the most popular methods use learning approaches like locally weighted projection regression [15], locally weighted Bayesian regression [16], Gaussian processes [17] or the recently proposed enhanced version of the principal-components echo state network [18]. Although there have been good experimental validations using these methods, they still have the issue of error accumulation and instability. Nonetheless, the purpose of this paper is not to compare these methods with the proposed approach. The application of learned dynamic models for control is also not a new field. Learning dynamics using Gaussian process regression for model predictive control was done in [19], where Gaussian processes also provide additional information about the uncertainty in prediction. Another approach where learned inverse dynamics models were used for computed torque control was discussed in [20]. However, none of these techniques

has been applied for control of a continuum or soft manipulator. For a comprehensive review on different learning methods of forward and inverse models refer to [21, 22].

1.2. Contributions

In this paper we demonstrate for the first time that the complete dynamics of a soft robot can be learned using a class of recurrent neural networks without any assumption about the form of the kinematic or dynamic model. We attempt to demonstrate that such an approach is very well suited for soft robots and much easier to develop than analytical models, with the ability to provide accurate, stable and scalable models. To the best of our knowledge our formulation of the dynamics of the system is unique and provides highly robust and stable dynamic models in combination with a recurrent neural network architecture. This allows the development of arbitrarily complex models of the manipulator from the dimension of the sensory elements while dispelling the need to develop complex analytical models and parameter estimation experiments. Finally, using the learned dynamic model we show that open loop predictive control policies can be successfully implemented even with long control horizons, both in simulation and on a real robot. This is also the first demonstration of the use of a learned dynamic controller in task space for a soft or continuum robotic manipulator and also the first implementation of a direct input space to task space control approach on a 3D manipulator. Through the experiments, we are able to show that it is possible to directly learn the forward dynamic mapping from the input commands to the task space variables. The next section describes the analytical model we use for validating the proposed controller, followed by a detailed description of our approach. The simulation results on the analytical model are presented in section 5. The experimental results on a pneumatically actuated soft manipulator are provided in section 6. A brief conclusion enumerating the advantages and disadvantages of the controller with possible improvements and applications is given in the last section.

2. Simulation setup

2.1. Model description

We use a recently developed analytical model for our simulation studies. This model will be used for preliminary validation of the proposed controller. The simulation setup is built on an advanced dynamic model for soft robotics recently developed based on a discrete Cosserat formulation of soft robot arm dynamics. In [23], the continuous Cosserat model for soft robotics previously presented in [1, 40] was discretized by assuming a PCS condition, which allows an analytical integration of the continuous kinematics and dynamic equations. This led to a discrete formulation of soft robot dynamics [37] that turns out to be the geometrically equivalent generalization of the traditional rigid robotics dynamics [39]. From a simulation point of view, the main features of the model are as follows:

- full multi-section dynamics capable of predicting the motion of complex soft robot arms;
- PCS assumption allowing for a full six degree of freedom (DoF) deformation of each section;
- support of any kind of external load, including the interaction with a dense medium like water (which is used for this work).

The development of the model is summarized next, while for a more detailed description we refer to [23, 37, 39]. In the Cosserat theory, a soft body is considered as an infinite stacking of infinitesimal micro-solids, whose configuration space is defined as a curve $\boldsymbol{g}(\cdot)\colon X \mapsto \boldsymbol{g}(X) \in SE(3)$ of homogeneous transformation parameterized by the material abscissa $X \in [0, L]$. Then, the strain state of the soft arm is defined by the vector field along the curve $\boldsymbol{g}(\cdot)$ given by $X \mapsto \hat{\boldsymbol{\xi}}(X) = \boldsymbol{g}^{-1}\partial \boldsymbol{g}/\partial X = \boldsymbol{g}^{-1}\boldsymbol{g}' \in \mathfrak{se}(3)$, where the hat is the isomorphism between the twist vector representation and the matrix representation of the Lie algebra $\mathfrak{se}(3)$. The time evolution of the configuration curve $\boldsymbol{g}(\cdot)$ is represented by the twist vector field $X \mapsto \boldsymbol{\eta}(X) \in \mathbb{R}^6$ defined by $\hat{\boldsymbol{\eta}}(X) = \boldsymbol{g}^{-1}\partial \boldsymbol{g}/\partial t = \boldsymbol{g}^{-1}\dot{\boldsymbol{g}}$.

With these definitions at hand, we can obtain the kinematic equations relating the strains of the robot arm $\boldsymbol{\xi}$ with the position $\boldsymbol{g}$, velocity $\boldsymbol{\eta}$ and acceleration $\dot{\boldsymbol{\eta}}$ for each infinitesimal micro-solid constituting the robot. The continuous kinematic equations are:

$$\boldsymbol{g}' = \boldsymbol{g}\hat{\boldsymbol{\xi}} \tag{1}$$

$$\boldsymbol{\eta}' = \dot{\boldsymbol{\xi}} - \mathrm{ad}_{\boldsymbol{\xi}}\boldsymbol{\eta} \tag{2}$$

$$\dot{\boldsymbol{\eta}}' = \ddot{\boldsymbol{\xi}} - \mathrm{ad}_{\dot{\boldsymbol{\xi}}}\boldsymbol{\eta} - \mathrm{ad}_{\boldsymbol{\xi}}\dot{\boldsymbol{\eta}} \tag{3}$$

where *ad* is the adjoint map, i.e. the adjoint representation of the vector field commutator in $\mathfrak{se}(3)$. In addition, for later use, the coadjoint map ad^* is defined as $\mathrm{ad}^* = -\mathrm{ad}^{\mathrm{T}}$.

The Cosserat beam dynamics can be directly derived from the extension to continuum media of a variational calculus originally introduced by Poincaré [25]. Applying this variational calculus to a Lagrangian density $(\mathfrak{T}(\boldsymbol{\eta}) - \mathfrak{U}(\boldsymbol{\xi}))$ leads to the strong form of a Cosserat beam with respect to the micro-solid frames:

$$\boldsymbol{\mathcal{M}}\dot{\boldsymbol{\eta}} + \mathrm{ad}^*_{\boldsymbol{\eta}}(\boldsymbol{\mathcal{M}}\boldsymbol{\eta}) = \boldsymbol{\mathcal{F}}'_{\mathrm{i}} + \mathrm{ad}^*_{\boldsymbol{\xi}}\boldsymbol{\mathcal{F}}_{\mathrm{i}} + \overline{\boldsymbol{\mathcal{F}}}_{\mathrm{a}} + \overline{\boldsymbol{\mathcal{F}}}_{\mathrm{e}} \tag{4}$$

where $\boldsymbol{\mathcal{F}}_{\mathrm{i}}(X) = \partial\mathfrak{U}/\partial X$ is the wrench of internal forces, $\overline{\boldsymbol{\mathcal{F}}}_{\mathrm{a}}(X, t)$ is the distributed actuation loads, $\overline{\boldsymbol{\mathcal{F}}}_{\mathrm{e}}(X)$ is the external wrench of distributed applied forces and $\boldsymbol{\mathcal{M}}(X)$ is the screw inertia matrix. A linear visco-elastic constitutive model has been chosen for the internal wrench: $\boldsymbol{\mathcal{F}}_{\mathrm{i}}(X) = \boldsymbol{\Sigma}(\boldsymbol{\xi} - \boldsymbol{\xi}^0) + \Upsilon\dot{\boldsymbol{\xi}}$, where $\boldsymbol{\Sigma}$ and Υ are constant screw stiffness and viscosity

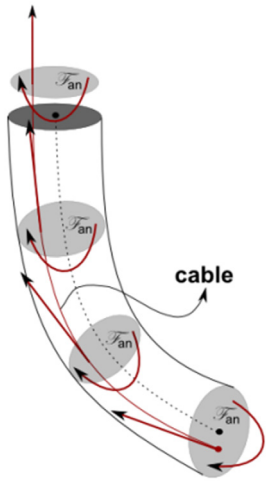

Figure 1. Depiction of the tendon actuation for one section.

matrices and $\boldsymbol{\xi}^0$ is the strain filed of the reference configuration. Regarding the distributed actuation load, in the case of cable-driven actuation, we have: $\overline{\mathcal{F}}_{\mathrm{a}}(X,t) = -\mathcal{F}'_{\mathrm{a}} - \mathrm{ad}^*_{\boldsymbol{\xi}}\mathcal{F}_{\mathrm{a}}$, where $\mathcal{F}_{\mathrm{a}}$ is the cable wrench acting on the micro-solid given by the cable tension and the cable path, as shown in figure 1. As for the external loads, we have considered the general case of underwater operation, i.e. distributed loads due to gravity and buoyancy, drag and added mass:

$$\overline{\mathcal{F}}_{\mathrm{e}}(X) = \left(1 - \frac{\rho_{\mathrm{w}}}{\rho}\right)\mathcal{G}\mathrm{Ad}^{-1}_{g,g(X)}\mathcal{G} - \mathcal{D}\,|\boldsymbol{\eta}|_v\,\boldsymbol{\eta} \quad (5)$$

$$\mathcal{M}_{\mathrm{a}} = \mathcal{M} + \mathcal{A} \quad (6)$$

where $|\cdot|_v$ takes the norm of the translational part of the operand, ρ_{w} is the density of water, $\mathcal{G}$ is the gravity twist, is the transformation between the spatial frame and the base frame of the soft manipulator, $\mathcal{D}(X)$ is the screw matrix of the drag fluid dynamics coefficient and $\mathcal{A}(X)$ is the screw matrix of the added mass fluid dynamics coefficient. Note here that replacing $\mathcal{M}$ by $\mathcal{M}_{\mathrm{a}}$ in (4) allows us to model the inertial hydrodynamics forces exerted along the arm. Finally, we have introduced the Adjoint representation (Ad) of Lie group $SE(3)$, while the coAdjoint map is defined by $\mathrm{Ad}^* = \mathrm{Ad}^{-\mathrm{T}}$.

The discretization of the continuous model outlined above is achieved through the PCS assumption [23]. At any instant t, considering the strain field $\boldsymbol{\xi}$ constant along each of the N sections of the soft arm, indicated by $[0,\ L_1), (L_1, L_2) \cdots (L_{N-1}, L_N]$, we can replace the continuous field with a finite set of N twist vectors $\boldsymbol{\xi}_n$ ($n \in \{1, 2, \cdots, N\}$), which play the role of the joint variables of traditional rigid robotics. Under this assumption, the continuous kinematics equations (1)–(3) become linear matrix differential equations, which can be analytically solved at any section n using the variation of parameters method with the appropriate initial value [26]. Applying this integration and rearranging the terms, we obtain the discrete kinematics equation [39]:

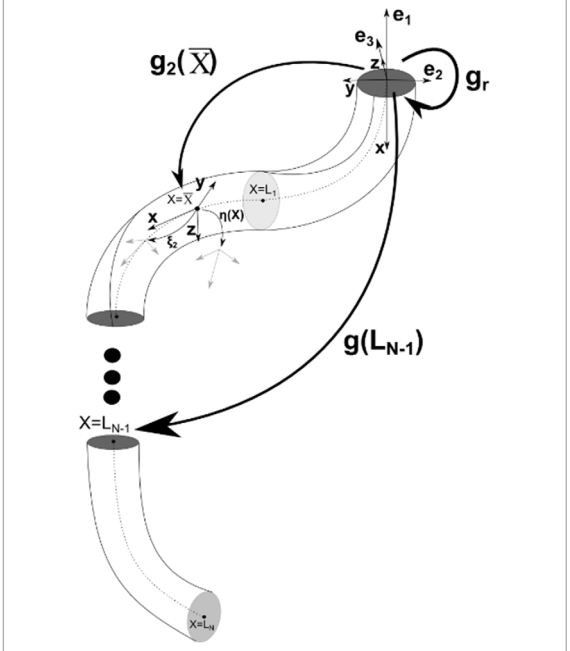

Figure 2. Schematic of the kinematics of the piece-wise constant strain model.

$$\boldsymbol{g}(X) = \boldsymbol{g}(L_{n-1})\mathrm{e}^{(X-L_{n-1})\hat{\boldsymbol{\xi}}_n} = \boldsymbol{g}(L_{n-1})\boldsymbol{g}_n(X) \quad (7)$$

$$\boldsymbol{\eta}(X) = \mathrm{Ad}^{-1}_{\boldsymbol{g}_n(X)}(\boldsymbol{\eta}(L_{n-1}) + \mathrm{AD}_{\boldsymbol{g}_n}(X)\dot{\boldsymbol{\xi}}_n) \quad (8)$$

$$\dot{\boldsymbol{\eta}}(X) = \mathrm{Ad}^{-1}_{\boldsymbol{g}_n(X)}(\dot{\boldsymbol{\eta}}(L_{n-1}) - \mathrm{ad}_{\mathrm{AD}_{\boldsymbol{g}_n}\dot{\boldsymbol{\xi}}_n}\boldsymbol{\eta}(L_{n-1}) + \mathrm{AD}_{\boldsymbol{g}_n}(X)\ddot{\boldsymbol{\xi}}_n) \quad (9)$$

where we have used the following results on the Lie Group SE(3) [36]:

$$\boldsymbol{g}_n(X) = \mathrm{e}^{x\hat{\boldsymbol{\xi}}_n} = \boldsymbol{I}_4 + x\hat{\boldsymbol{\xi}}_n + 1/\theta_n^2(1-\cos(x\theta_n))\hat{\boldsymbol{\xi}}_n^2 + 1/\theta_n^3(x\theta_n - \sin(x\theta_n))\hat{\boldsymbol{\xi}}_n^3 \quad (10)$$

$$\begin{aligned}\mathrm{Ad}_{\boldsymbol{g}_n(X)} &= \mathrm{e}^{x\mathrm{ad}_{\boldsymbol{\xi}_n}} \\ &= \boldsymbol{I}_6 + 1/2\theta_n(3\sin(x\theta_n) - x\theta_n\cos(x\theta_n))\mathrm{ad}_{\boldsymbol{\xi}_n} \\ &+ 1/2\theta_n^2(4 - 4\cos(x\theta_n) - x\theta_n\sin(x\theta_n))\mathrm{ad}^2_{\boldsymbol{\xi}_n} \\ &+ 1/2\theta_n^3(\sin(x\theta_n) - x\theta_n\cos(x\theta_n))\mathrm{ad}^3_{\boldsymbol{\xi}_n} \\ &+ 1/2\theta_n^4(2 - 2\cos(x\theta_n) - x\theta_n\sin(x\theta_n))\mathrm{ad}^4_{\boldsymbol{\xi}_n}\end{aligned} \quad (11)$$

$$\begin{aligned}\mathrm{AD}_{\boldsymbol{g}_n}(X) &= \int_{L_{n-1}}^{\Lambda}\mathrm{Ad}_{\boldsymbol{g}_n(s)}\mathrm{d}s \\ &= x\boldsymbol{I}_6 + 1/2\theta_n^2(4 - 4\cos(x\theta_n) - x\theta_n\sin(x\theta_n))\mathrm{ad}_{\boldsymbol{\xi}_n} \\ &+ 1/2\theta_n^3(4x\theta_n - 5\sin(x\theta_n) + x\theta_n\cos(x\theta_n))\mathrm{ad}^2_{\boldsymbol{\xi}_n} \\ &+ 1/2\theta_n^4(2 - 2\cos(x\theta_n) - x\theta_n\sin(x\theta_n))\mathrm{ad}^3_{\boldsymbol{\xi}_n} \\ &+ 1/2\theta_n^5(2x\theta_n - 3\sin(x\theta_n) + x\theta_n\cos(x\theta_n))\mathrm{ad}^4_{\boldsymbol{\xi}_n}\end{aligned} \quad (12)$$

with θ_n the norm of the rotational part of the constant strain $\boldsymbol{\xi}_n$ and $x = X - L_{n-1}$. A schematic of the piece-wise constant strain kinematics is shown in figure 2.

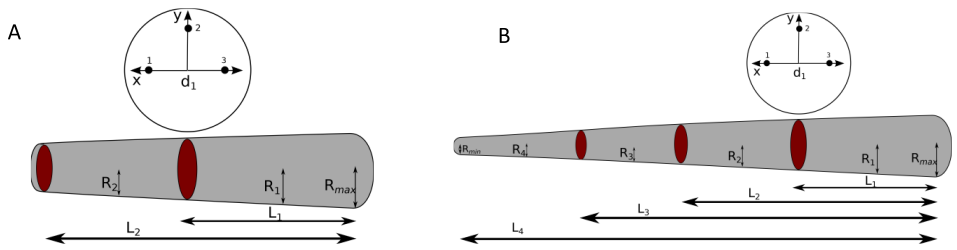

Figure 3. Schematic of the soft manipulator used in the simulation. (a) Two section manipulator (b) four section manipulator. The red circles represent the cross sections to which the cables are anchored.

In order to develop the discrete Cosserat dynamic model for soft robots a relation between the kinematics quantities $\boldsymbol{\eta}, \dot{\boldsymbol{\eta}}$ and a joint vector for soft robotics needs to be established [37]. To do this, we back track to the base the velocity term $\boldsymbol{\eta}(L_{n-1})$ on the right-hand side of (8), which becomes:

$$\begin{aligned}\boldsymbol{\eta}(X) &= \sum_{i=1}^{n}\left(\prod_{j=n}^{i}\mathrm{Ad}^{-1}_{\boldsymbol{g}_j(\min(L_j,X))}\right)\mathrm{AD}_{\boldsymbol{g}_i}(\min(L_i,X))\dot{\boldsymbol{\xi}}_i \\ &= \sum_{i=1}^{n}\boldsymbol{S}_i(X)\dot{\boldsymbol{\xi}}_i = \boldsymbol{J}(X)\dot{\boldsymbol{q}}\end{aligned} \tag{13}$$

where we have defined the soft robot geometric Jacobian $\boldsymbol{J}(X) = [\boldsymbol{S}_1(X)\boldsymbol{S}_2(X)\cdots\boldsymbol{S}_N(X)] \in \mathbb{R}^6 \otimes \mathbb{R}^{6N}$ and the soft robots joint vector $\boldsymbol{q} = [\boldsymbol{\xi}_1^{\mathrm{T}}, \boldsymbol{\xi}_2^{\mathrm{T}}, \cdots, \boldsymbol{\xi}_N^{\mathrm{T}}]^{\mathrm{T}} \in \mathbb{R}^{6N}$. Similarly, by taking the time derivative of (13), equation (9) can be written as

$$\dot{\boldsymbol{\eta}}(X) = \boldsymbol{J}(X)\ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}(X)\dot{\boldsymbol{q}}. \tag{14}$$

At this point, we are able to obtain the discrete Cosserat dynamics by projecting the continuous dynamics (4) into the joint space with the Jacobian transpose $\boldsymbol{J}^{\mathrm{T}}$ and integrating over the different pieces of the soft arm. By doing that and substituting the discrete model of velocity (13) and acceleration (14), we obtain (as shown in [37])

$$\begin{aligned}&\boldsymbol{\mathcal{M}}(\boldsymbol{q})\ddot{\boldsymbol{q}} + (\boldsymbol{\mathcal{C}}_1(\vec{\xi}, \dot{\vec{\xi}}) + \boldsymbol{\mathcal{C}}_2(\boldsymbol{q}, \dot{\boldsymbol{q}}))\dot{\boldsymbol{q}} \\ &\quad = \boldsymbol{\tau}(\boldsymbol{q}) + N(\boldsymbol{q})\mathrm{Ad}^{-1}_{\boldsymbol{g}_r}\boldsymbol{\mathcal{G}} - \boldsymbol{\mathcal{D}}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}}\end{aligned} \tag{15}$$

where we recognize the structure of the Lagrangian model of rigid serial manipulators and we have introduced the soft robotics generalized actuation vector: $\boldsymbol{\tau} = [\boldsymbol{\tau}_1^{\mathrm{T}}, \boldsymbol{\tau}_2^{\mathrm{T}}, \cdots, \boldsymbol{\tau}_N^{\mathrm{T}}]^{\mathrm{T}} \in \mathbb{R}^{6N}$. The different terms of (15) are specified below, block-element-wise [37]:

$$\boldsymbol{\mathcal{M}}_{(n,m)} = \sum_{i=\max(n,m)}^{N}\int_{L_{i-1}}^{L_i}\boldsymbol{\mathcal{S}}_n^{\mathrm{T}}\boldsymbol{\mathcal{M}}_{\mathrm{a}}\boldsymbol{\mathcal{S}}_m \mathrm{d}X \tag{16}$$

$$\boldsymbol{\mathcal{C}}_{1_{(n,m)}} = \sum_{i=\max(n,m)}^{N}\int_{L_{i-1}}^{L_i}\boldsymbol{\mathcal{S}}_n^{\mathrm{T}}\mathrm{ad}^{*}_{\boldsymbol{J}\dot{\boldsymbol{q}}}\boldsymbol{\mathcal{M}}_{\mathrm{a}}\boldsymbol{\mathcal{S}}_m \,\mathrm{d}X \tag{17}$$

$$\boldsymbol{\mathcal{C}}_{2_{(n,m)}} = -\sum_{i=\max(n,m)}^{N}\int_{L_{i-1}}^{L_i}\boldsymbol{\mathcal{S}}_n^{\mathrm{T}}\boldsymbol{\mathcal{M}}_{\mathrm{a}}\mathrm{ad}_{\sum_{j=m+1}^{i}\boldsymbol{\mathcal{S}}_j\dot{\xi}_j}\boldsymbol{\mathcal{S}}_m \mathrm{d}X \tag{18}$$

Table 1. Design parameters of the simulated prototype.

Parameter	Value	Parameter	Value
$R_{\max}$	15 mm	d_1, d_2, d_3	9 mm
$R_{\min}$	4 mm	Gravity acceleration	9.81 m s^{-2}
L_1	98 mm	Drag coefficient x C_x	0.01
L_2	203 mm	Drag coefficient y C_y	2.5
L_3	311 mm	Drag coefficient z C_z	2.5
L_4	418 mm	Added mass coeff. y B_y	1.5
Young modulus E	110 kPa	Added mass coeff. z B_z	1.5
Shear viscosity modulus μ	300 Pa s	ρ_{w}	1.02 kg dm^{-3}
Poisson ratio ν	0.5	ρ	1.08 kg dm^{-3}

$$\boldsymbol{\mathcal{D}}_{(n,m)} = \sum_{i=\max(n,m)}^{N}\int_{L_{i-1}}^{L_i}\boldsymbol{\mathcal{S}}_n^{T}\boldsymbol{\mathcal{D}}\boldsymbol{\mathcal{S}}_m\left|\boldsymbol{J}\dot{\boldsymbol{q}}\right|_{\nu}\mathrm{d}X \tag{19}$$

$$\boldsymbol{\mathcal{N}}_{(n)} = (1-\rho_{\mathrm{w}}/\rho)\sum_{i=n}^{N}\int_{L_{i-1}}^{L_i}\boldsymbol{\mathcal{S}}_n^{\mathrm{T}}\boldsymbol{\mathcal{M}}\mathrm{Ad}^{-1}_{\boldsymbol{g}}\,\mathrm{d}X. \tag{20}$$

Finally, for cable-driven soft arms, the actuation load at each section is given by

$$\boldsymbol{\tau}_n = (L_n - L_{n-1})\left(\sum_{j=n}^{N}\boldsymbol{\mathcal{F}}_{\mathrm{aj}} - \boldsymbol{\mathcal{F}}_{\mathrm{in}}\right) \tag{21}$$

where $\boldsymbol{\mathcal{F}}_{\mathrm{aj}}$ indicates the contribution of the cables attached at L_n (figure 1) and $\boldsymbol{\mathcal{F}}_{\mathrm{in}}$ is the constant internal load of the section n.

Simulation results and experimental comparisons against real soft robotics prototypes for the piece-wise constant strain model can be found in [23, 37, 39].

2.2. Soft arm design description

The simulated prototype is shown in figure 3 and described in [1], to which we refer for more exhaustive details. In short, the prototype is composed of a single conical piece of silicone, with a base radius $R_{\max}$ and a tip radius $R_{\min}$, actuated by 12 cables embedded inside the robot body. The cables run parallel to the midline at a distance d_j ($j \in \{1, 2, \cdots, 12\}$) and are anchored four at a time at three different lengths along the robot arm (L_1, L_2, L_3) and with a relative angle of 90° (figure 3). The soft arm operates underwater. For this paper, we use two configurations of the manipulator for

our controller (figure 3). The first case has only two sections, actuated only in the proximal section by three cables, and the second case has all four sections and is actuated in the same way.

In order to exploit the dynamics equations developed above, the soft manipulator has been modeled as a stack of four cylindrical constant-strain sections defined by L_1, L_2, L_3, L_4, with a radius equal to the mean of the prototype radius for each section (R_1, R_2, R_3 and R_4 in figure 3). The value of the parameters used in the simulation is reported in table 1.

3. Learning the forward dynamics

In this section we describe how our learned dynamic model is formulated and developed. This learned model will be verified on the simulated model described above and on a real soft robotic platform for validation. Assume that the infinite dimensional configuration space can be approximated using an n-dimensional vector space. The kinematics of the manipulator can now be represented as

$$\boldsymbol{x} = F(\boldsymbol{q}) \tag{22}$$

where $\boldsymbol{x}$ is the task space variable and $q \in \mathbb{R}^n$ is the configuration space variable. In order to obtain this inverse kinematics, a sufficient condition is that the dimension of the task space variable is also n. Consequently all instances of the configuration space variable q can be replaced by the task space variable $\boldsymbol{x}$. Note that the approximation of the inverse kinematics (consequently the forward dynamics) becomes more accurate with a higher-dimensional task space representation.

Using these assumptions, it is possible to transform the forward dynamics of the manipulator from the usual form given in (23) to a form using only the task space variables, as shown in (24):

$$M(\boldsymbol{q})\ddot{\boldsymbol{q}} + V(\boldsymbol{q})\dot{\boldsymbol{q}} + P(\boldsymbol{q}) = \tau \tag{23}$$

$$\overline{M}(\boldsymbol{x})\ddot{\boldsymbol{x}} + \overline{V}(\boldsymbol{x})\dot{\boldsymbol{x}} + \overline{P}(\boldsymbol{x}) = \tau. \tag{24}$$

Here, $\tau \in \mathbb{R}^m$ are the control inputs. M, V and P represent the inertia matrix, centripetal–Coriolis forces and potential energy stored due to gravity/deformation, respectively. $\overline{M}$, $\overline{V}$ and $\overline{P}$ are the corresponding matrices obtained after the transformation. This implies that, under these assumptions, it is always possible to learn a direct mapping between the states of the task space variables and the control inputs: $(\boldsymbol{\tau}, \boldsymbol{x}, \dot{\boldsymbol{x}}) \rightarrow \ddot{\boldsymbol{x}}$.

Consequently, by varying the dimension of the task space (the number of sensory inputs), the user can arbitrarily increase or decrease the complexity and accuracy of the dynamic/kinematic model (refer to section 3.1.2 for a demonstration). This is a huge advantage that machine learning provides for learning the dynamics of a soft manipulator. On the contrary, for model-based approaches, the analytical dynamic model determines the sensory requirements (not just the dimensionality but also the type). For instance, models that are based on the CC approximation rely on cable potentiometers for measuring the length of modules (at least in the case of pneumatically actuated manipulators). For rigid robots, the dimension of the joint space (equivalent to the configuration space) is finite and therefore the number of sensors required is fixed.

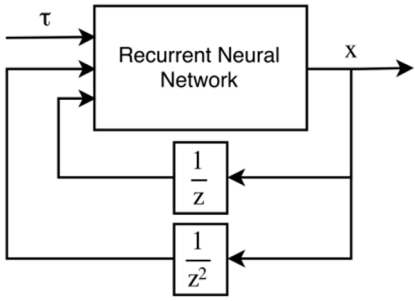

Figure 4. The architecture of the dynamic model using the NARX network.

Taking a cue from our previous works on learned controllers [27, 28], we modify the mapping in terms of absolute values, thereby obtaining the new mapping $(\boldsymbol{\tau}^{\mathrm{c}}, \boldsymbol{x}^{\mathrm{p}}, \boldsymbol{x}^{\mathrm{c}}) \rightarrow \boldsymbol{x}^{\mathrm{n}}$, where the superscript p represents the previous value of the variable, c represents the current value and n represents the next value. The new mapping is an approximation of the continuous dynamic model using a finite difference approximation. This would restrict the learned dynamic model to have a fixed step size. But by representing the variables only in absolute terms, we gain three main advantages. Primarily, such a mapping allows us to represent the dynamic model using a recurrent neural network (RNN) (see figure 4). The advantages of a nonlinear autoregressive network with exogenous inputs (NARX) for long-term time series prediction have been widely discussed [29, 30]. In addition, representing the dynamic model using only absolute terms also helps in encoding the boundary conditions in the data which, in turn, helps the stability of the network. Finally, in this way we can avoid taking first-order and second-order derivatives of the position term which would increase the variance of any existing noise. This would further deteriorate the prediction performance.

The structure and learning algorithm of the network is described in the next section.

3.1. Recurrent neural network description and learning

We are using a NARX for developing a multi-step prediction model for the forward dynamics. The advantage of using a recurrent-dynamic network over a recursive feedforward-dynamic network is that such networks are more accurate as the training is done to reduce the cumulative error over the whole continuous training set (feedforward networks try to only reduce the prediction error for each step and are thereby prone to overfitting and instability). The architecture of the dynamic model based on the recurrent network is shown in figure 4. Note that the inputs are normalized inside the network. The network has a single hidden

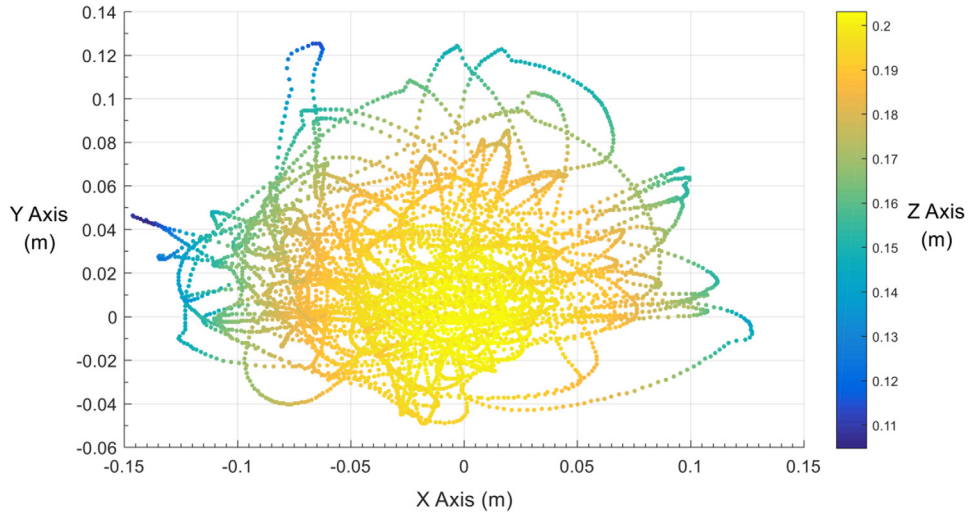

Figure 5. Workspace of the manipulator obtained by the random exploration.

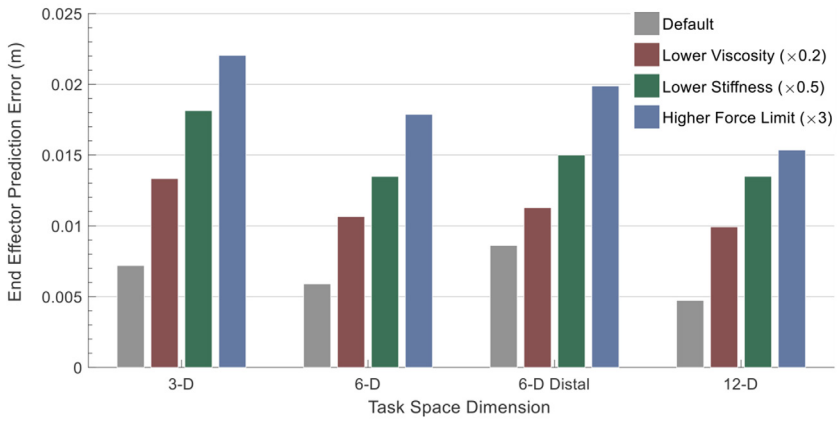

Figure 6. Mean multistep prediction error using the NARX network for different manipulator characteristics.

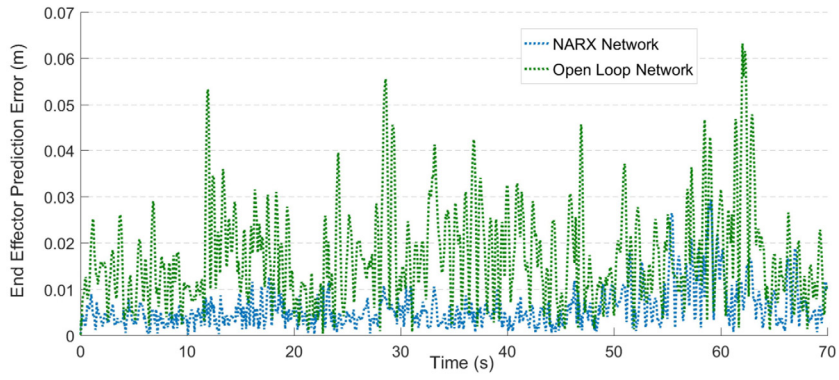

Figure 7. Time evolution of the multistep prediction error for the recurrent network and open loop network.

layer with 35 units. The transfer function in the hidden layer is Tan-sigmoid and a linear transfer function is used in the output layer.

3.1.1. *Sampling and training*

For the purpose of this paper, samples for learning are obtained from the simulated cable-driven two-section soft manipulator described in section 2.2. The second section is unactuated and the first section is actuated by three radially arranged cables. The exploration is done by inputting pseudorandom variable-amplitude square wave sequences, with a 50% probability of the actuator being idle. The exploration signals are decided based on empirical data. The maximum force applicable by the cables is fixed to 3 N. Sampling is done at a fixed frequency of 100 Hz and consists of 7000 samples, amounting to a duration of 70 s. The corresponding explored workspace is shown in

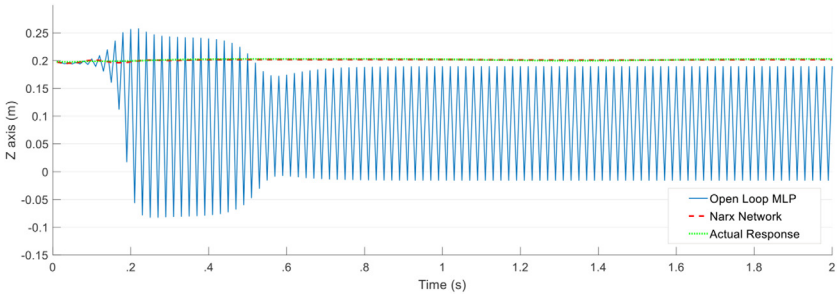

Figure 8. Instability of the open loop network for a zero actuation case. The response of the recurrent network and the simulation is given for reference.

figure 5. Additionally, for more efficient and complete exploration, more goal-directed explorations can be performed after learning the forward dynamics with the initial sample.

The training of the network is done in two steps. Initially, the network is trained in open loop by unfolding the recurrent network and training by Bayesian regularization. However, this trained network is prone to over fitting, and therefore the network is closed and further trained using the same network weights (closing the loop does not change the size of the network). The performance function for the open network is calculated as

$$\mathrm{MSE} = \frac{1}{T}\sum_{t=0}^{T}\|X_t - f(X_{t-1}, U_t)\|^2 \qquad (25)$$

where X is the input vector and U is the exogenous input vector. The function f represents the mapping formed by the neural network. For training the recurrent network Levenberg–Marquardt backpropagation is used and a validation set is used to avoid overfitting. Directly training the closed loop network from randomly initialized weights is not desirable as the training would be highly susceptible to the gradient exploding problem. Also, the training and testing sets are divided into continuous (to keep time correlations intact) blocks in the ratio 70:30 for the first step (open loop network) then the training, testing and validation set are divided in the ratio 70:15:15 for the final step (closed loop network). The performance function is now represented as

$$\mathrm{MSE} = \frac{1}{T}\sum_{t=0}^{T}\|X_t - f(\widehat{X}_{t-1}, U_t)\|^2. \qquad (26)$$

Here, $\widehat{X}_{t-1}$ is the prediction of the NARX network in the previous iteration. Now the learning algorithm is not trying to reduce the single-step error but the whole multi-step prediction error (the performance function is the only difference between the open loop network and the closed loop network). The next section describes how the dimension of the task space variables is decided.

3.1.2. *Deciding on the task space dimension*

As previously mentioned, it is up to the user to decide the dimension of the task variables that determines the underlying dynamic model. Clearly, providing more information about the state of the manipulator leads to a better prediction. The mean prediction error for different numbers of task space parameters is shown in figure 6. The prediction is computed by the recurrent network for all the sample data by a single multi-step ahead simulation (a 70 s simulation). For the three-dimensional (3D) case, only the Cartesian position of the end effector is used for prediction. For the 6D case two scenarios are compared; the first one uses the Cartesian position of the tip of both sections and the second uses only Cartesian positions of the tip (end effector) and the midpoint of the second section. For the 12D case, Cartesian positions of the tip and the mid-point of each section are used. For all the cases, the network size is fixed (35 neurons).

Table 2. Reaching error for 50 random targets.

	Mean error (m)	Standard deviation (m)
Predicted by RNN	0.001	0.001
From simulation	0.007	0.002
Difference between prediction and simulation	0.007	0.002

Furthermore, we also try to investigate how the prediction accuracy is affected by material properties and force limits (figure 6). As expected, varying these parameters increases the chaoticity of the manipulator dynamics and thereby deteriorates the prediction accuracy. For all cases a better accuracy can be obtained by increasing the task space dimension. Changing the environment from water to air makes the manipulator dynamics highly chaotic and therefore not learnable. This is because significantly larger actuation forces are required to compensate for the effects of gravity on the soft body. Numerical instabilities in the analytical model are also a problem with high forces.

The time evolution of prediction error for a single 70 s simulation of the manipulator is shown in figure 7. Slight overfitting of the data can be seen from the apparent increase in error near the training set. However, more importantly, the errors are bounded even for such a long simulation. The corresponding error plot for the open loop network obtained after the first training is also shown in figure 7. Both the plots are obtained for the 12D case. The stability advantages

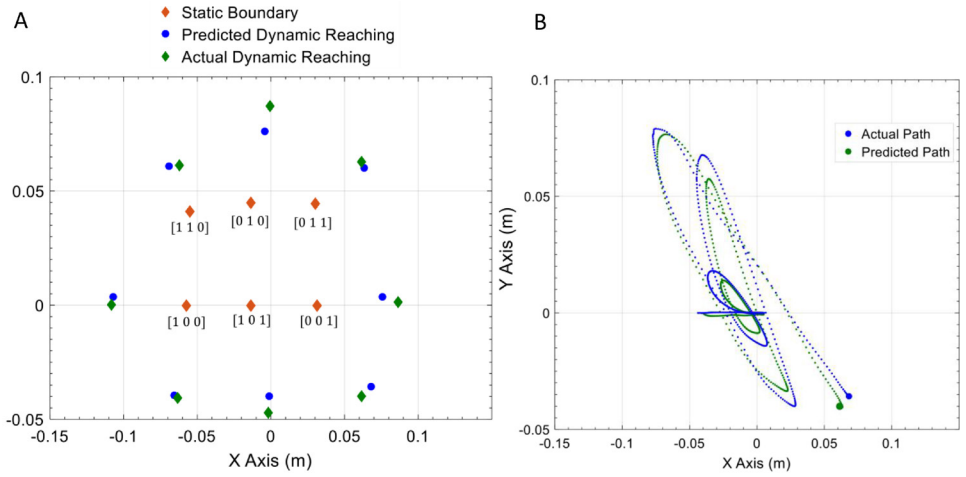

Figure 9. (a) Static reachable boundaries of the manipulator and the reachability of the manipulator with a dynamic controller. (b) Illustration of the complex path the manipulator takes to reach one example target.

of the NARX network over the open loop network obtained from the first learning can be seen in figure 8, where the input forces are all set to zero. The error accumulation problem causes the open loop network to become highly unstable even for this simple case.

Although the learned dynamic model may not be as accurate as a detailed analytical formulation, the recurrent neural network runs much faster. A 2 s simulation of the forward model takes 63 ms using the recurrent neural network, whereas the same simulation using the PCS model takes 523 s. Both the models are evaluated on an Intel(R) Core(TM) i7-3630QM CPU @ 2.40 GHz with 8 Gb RAM

4. Trajectory optimization

Once we have obtained the learned dynamic model of the manipulator, trajectory optimization can be performed for developing an open loop predictive controller. For this purpose, we employ a single shooting technique to obtaining the optimal control policies.

Let the fixed control horizon be t_f. It is discretized by a fixed step size of dt (10 ms in our case). Given the control policy, the trajectory of the dynamic system can be simulated using the recurrent neural network

$$x_{i+1} = f(x_i, x_{i-1}, \tau_i) \quad \forall\, i = 1.. \frac{t_f}{dt} \tag{27}$$

where x_i, x_{i-1} and x_{i+1} represent the current, previous and next state positions of the manipulator, respectively. τ_i is the forces applied on all the cables at each time step and f represents the learned mapping. To simplify the optimization problem and for computational reasons we reduce the number of variables by expanding the control frequency to $1/t_s$ (t_s is 50 ms in our case). The control inputs for each time step dt can now be written as:

Table 3. Reaching error for the targets with limited actuation forces.

	Mean error (m)	Standard deviation (m)
Predicted by RNN	0.033	0.021
From simulation	0.026	0.022
Difference between prediction and simulation	0.007	0.004

$$\tau_i^m \equiv \begin{cases} \tau_{i-1}^m, & \mathrm{mod}(i, t_s/dt) \neq 0 \\ \overline{\tau}_k^m, & \mathrm{mod}(i, t_s/dt) = 0 \end{cases} \quad \forall\, m = 1.. M$$
$$i = \left\lfloor \frac{t}{dt} \right\rfloor \quad \forall\, t = 0 \ldots t_f$$
$$k = \left\lfloor \frac{t}{t_s} \right\rfloor \quad \forall\, t = 0 \ldots t_f. \tag{28}$$

Here, M is the number of actuators and t is the current time. The time-dependent control policy is represented by the low-dimensional vector $\overline{\tau}$:

$$\Pi(t) = \overline{\tau}_k^m \quad \forall\, m = 1 \ldots M$$
$$k = \left\lfloor \frac{t}{t_s} \right\rfloor \quad \forall\, t = 0 \ldots t_f. \tag{29}$$

The optimal policy can be estimated by minimizing the objective function given below:

$$\Pi(t)^* = \min_{\tau} \left\| x_{\frac{t_f}{dt}}^{\mathrm{task}} - x^{des} \right\|^2 + \sum_k \tau_k^{\mathrm{T}} R\, \tau_k$$
$$\text{subject to } 0 \leqslant \tau_k^m \leqslant \tau_{\max}^m \quad \forall\, m = 1 \ldots M \text{ and } k = 0 \ldots \frac{t_f}{t_s}. \tag{30}$$

The control objective is formulated to reach a desired position at the end of the control horizon while simultaneously optimizing the control effort. No

Table 4. Performance of the controller with noise-contaminated sample data.

	With noise		Without noise	
	Mean error (m)	Standard deviation (m)	Mean error (m)	Standard deviation (m)
Predicted by RNN	0.03	0.019	0.019	0.02
From simulation	0.043	0.031	0.021	0.019
Difference between prediction and simulation	0.026	0.015	0.006	0.002

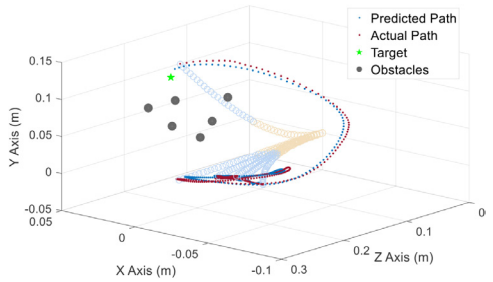

Figure 10. Highly dexterous motion achievable due to the manipulator properties and controller formulation.

constraint on the final velocity is given since it difficult to estimate if a point is statically and dynamically reachable using only the learned model.

To solve this nonlinear optimization problem we use the iterative sequential quadratic programming (SQP) algorithm [31]. Since the dynamic model is represented by neural networks with continuous and smooth transfer functions, the objective function is always twice continuously differentiable. The derivatives and double derivatives are estimated by numerical methods for the objective function. The MATLAB fmincon function is used for optimization. After optimization we obtain an optimal policy that controls the manipulator to the desired position in the commanded time period. Using parallel processing on four cores, a 10-step iteration for a control horizon of 1 s (60 variables) takes 6.2 s on average. This is not fast enough to implement a closed loop model predictive controller, therefore the developed controller is fully open loop with no feedback.

5. Simulation results

First we present the tracking error incurred by the open loop predictive control policy using the described learned dynamic model and trajectory optimization algorithm for a two-section soft manipulator (see section 2). Fifty points are randomly selected from the end effector workspace and the objective function is designed so that the end effector reaches the target at the end of the control horizon (2 s) with minimal control effort. The actuator forces are limited to 2.5 N and the optimization algorithm is run for 20 iterations. The results are summarized in table 2. The error is calculated at the end of the control horizon between the position of the end effector and the target. The optimization on average takes 25 s.

Table 5. Performance of the controller for the four-section manipulator.

	Mean error (m)	Standard deviation (m)
Predicted by RNN	0.005	0.007
From simulation	0.018	0.004
Difference between prediction and simulation	0.016	0.004

The simulations conducted next are done to showcase four important characteristics that we consider important. The first simulation is to demonstrate the need for dynamic controllers when actuator forces are limited or scenarios where energy conservation is vital. The second simulation is to validate the approach for more realistic scenarios, and is simulated by adding artificial noise to the sample data. The third simulation is performed to highlight the high dexterity and manipulability that a soft manipulator can achieve with the help of dynamic controllers. The final simulation is to exhibit the scalability of the proposed approach to higher-dimensional nonlinear soft manipulators.

5.1. Dynamic reaching

The advantage of using a dynamic controller is not only limited to energy and time considerations. Furthermore, such a controller can expand the workspace of manipulators with fixed actuator forces. To exhibit this, and to validate the trajectory optimization approach with the learned model, a dynamic reaching simulation is conducted. The tests are conducted using the two-section soft manipulator with three cables. The distal section is underactuated. The maximum force applicable by the cables is also limited to 1 N. The reachability of the manipulator if it only relied on a static controller is shown in figure 9(a). This is achieved by giving constant forces to each cable (shown in brackets in figure 9) and letting the manipulator stabilize for 10 s. To show the dynamic boundaries, a set of targets are set circumferentially around the home position. The trajectory optimization algorithm is run on the learned dynamic model for 10 iterations. The time period is set at 5 s. The control horizon is longer than required to showcase that the

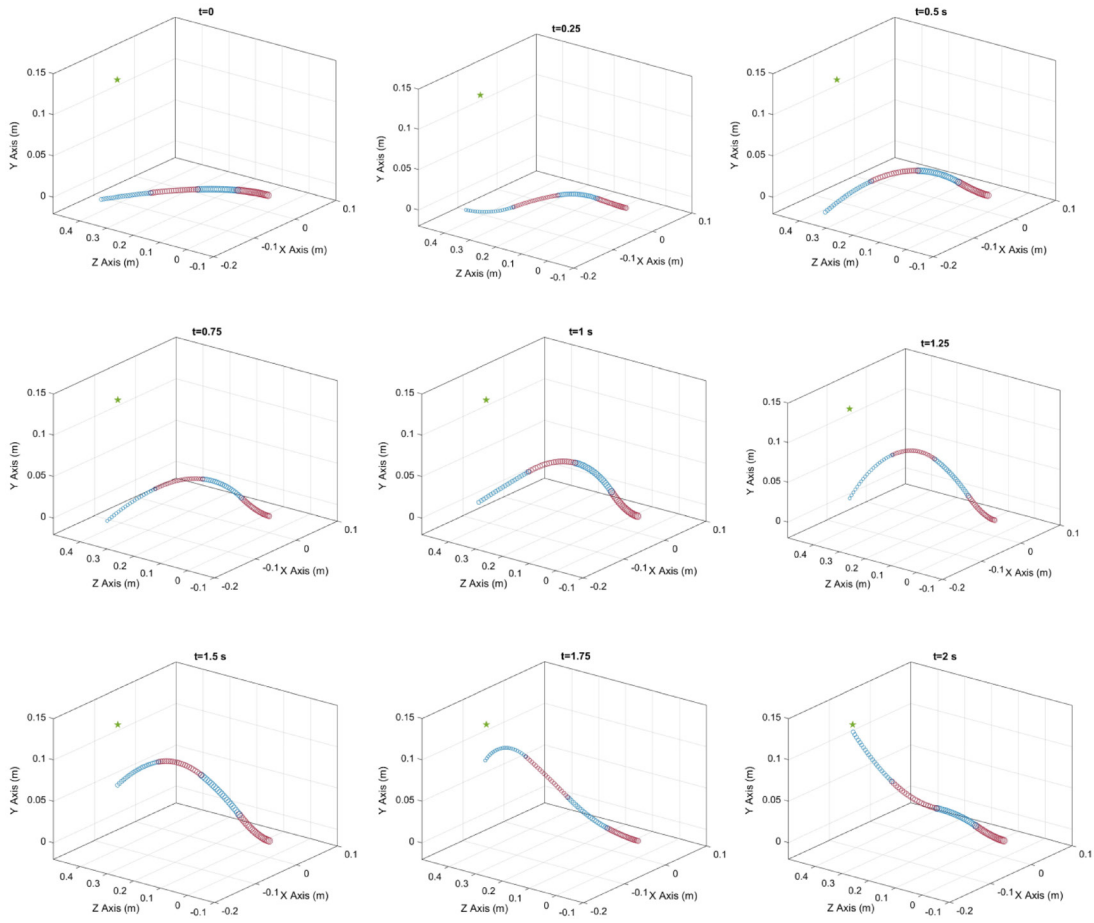

Figure 11. Reaching task executed by the four-section manipulator. Each section is differently colored, with the base and only actuated section fixed at [0 0 0]. The target is shown in green.

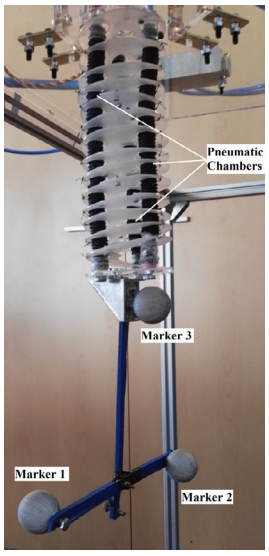

Figure 12. Pneumatically actuated manipulator used for the experiments.

prediction error does not rise exponentially even for such a long prediction horizon. The predicted boundaries of the manipulator are shown in figure 9(a) with the actual position of the end effector obtained using the numerical simulation with the obtained open loop policy. The predicted path generated by the optimization algorithm in conjunction with the learned model for one case is shown in figure 9(b). The corresponding trajectory for the same policy for the numerical simulation is shown in blue.

The average error and standard deviation of the controller in the reaching task are shown in table 3 for the eight reaching targets. The validity of the learned model can be seen from the difference between the prediction and simulation positions.

5.2. Effect of noise

One of the main concerns of modeling using machine learning is its performance under sensory noise, especially for dynamic models. The purpose of this section is to show how the proposed learned model and predictive controller would fare with artificially added noise in the sample data during training. Since the NARX network is trained to minimize the overall prediction error in a large time series prediction, it is to some extent able to weed out the effects of noise. Additionally, neural networks are also good at dealing with noisy data.

A zero mean Gaussian noise of standard deviation 0.005 m is added to the task space positions obtained from the simulation. Since the representation of the network is only in absolute coordinates, we do not need to take the first-order and second-order

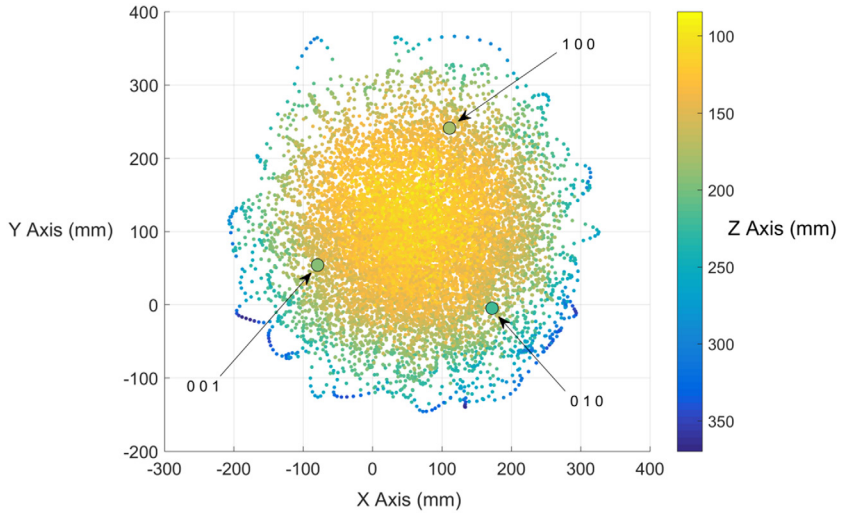

Figure 13. Dynamic workspace of the manipulator obtained by motor babbling. The static boundaries are also denoted for comparison. The static boundaries are obtained by actuating one of three pneumatic chambers to the maximum and letting the manipulator settle.

derivatives of the position to obtain the velocity and acceleration of the manipulator. This makes our network more robust to sensor noises, as stated earlier. We are using the 12 DoF task space model for this scenario. Learning is performed with these data and the performance of the trajectory optimization algorithm is compared with the previously learned model. The control horizon is only 2 s for this case; however, the actuator limits are increased to 2 N and the optimization algorithm is run for 10 iterations. The performance of the controller with the new learned dynamic model is shown in table 4 with the corresponding error of the original model for comparison. The same eight targets as in the previous section are used. Also note that with the shorter control horizon the accuracy of the controller is better and the prediction errors are smaller, as expected.

5.3. Obstacle avoidance

Another advantage of a high-dimensional robot coupled with a dynamic controller is that it can provide highly redundant and fast motion while being inherently safe using only a few actuators. To demonstrate this, we devise a scenario where the manipulator aims to reach a target position while avoiding obstacles in its path. The obstacles are shown in figure 10. The objective function is modified such that the end effector tries to stay as far away as possible from the obstacles while ensuring that the target is reached.

5.4. Scalability

Another aspect of interest is the scalability of this approach to higher-dimensional systems. For this the same approach is tested on a four-section manipulator (see figure 3(b)). The samples collected are for the same 70 s duration with the task space dimension summing to 24 (6 for each section). Three actuators, arranged in the same configuration as the previous case, are the only inputs. After learning the forward dynamics, the same experiments on reaching a static target using the end effector are performed for 10 randomly selected points. The control horizon is for 2 s, the forces are constrained to 2 N and the optimization algorithm is run for 10 iterations. The results are summarized in table 5. The error is calculated at the end of the control horizon between the position of the end effector and the target. Figure 11 shows one example case of the manipulator performing the reaching task (see supplementary materials for the video, available at stacks.iop.org/BB/12/066003/mmedia).

6. Experimental results

For preliminary validation of the control approach developed here we perform tests on a single-section pneumatically actuated manipulator [43]. The manipulator is cylindrical and placed in air, unlike the simulations. The single module consists of three pneumatic chambers arranged in a radially symmetric configuration (see figure 12). In this way the experimental model is different from the simulations in terms of morphology, actuation and environment, and therefore showcases the wide applicability of our proposed method. We use a pressure regulator for the closed loop control of the chamber pressures. The Vicon tracking system is used to track three markers attached to the tip of the manipulator during the tests (figure 12).

We use the electronic proportional micro-regulator Series K8P for the pressure regulation. Since our method is a data-driven approach, and due to the high linearity between the commanded signal and the pressure output ($\leqslant \pm 1\%$ full span), it is not necessary to derive our dynamic model in terms of the actual pressure in the chamber. Instead, our dynamic equation (equation (24)) becomes a function of the commanded signal rather than the actual forces (pressure)

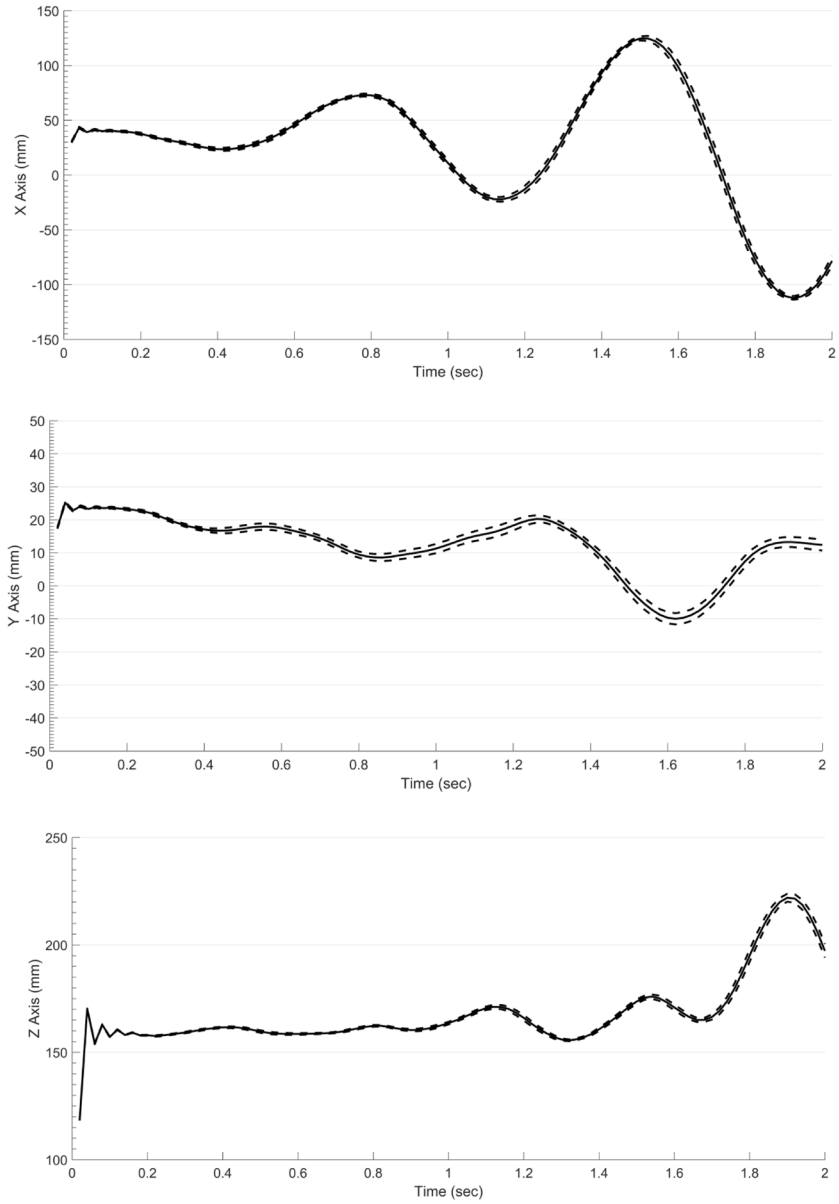

Figure 14. Repeatability of the manipulator dynamics.

Table 6. Prediction performance of the learned dynamic model.

	Mean error (m)	Standard deviation (m)
Using three markers	0.038	0.027
Using two markers	0.055	0.031
Using one marker	0.056	0.031

inside the chamber. This is a huge advantage of learning-based methods over analytical methods, since the learned model will automatically incorporate the actuator dynamics. Therefore, we can learn a direct end-to-end mapping between the commanded signals and manipulator states to the future manipulator states.

The methodology of sampling and learning is the same as performed for the simulations. The sampling rate, however, is reduced to 50 Hz while the sampling period is increased to 240 s. The chamber pressure is limited to 1.1 bar (relative to the atmospheric pressure).

6.1. Workspace and repeatability

The end effector workspace obtained by motor babbling is shown in figure 13 along with the statically reachable boundaries. We define the end effector as the point lying halfway between the bottom two markers (markers 1 and 2). The length of the module from the base to the end effector is 320 mm, while the soft module is 165 mm in the unactuated configuration. This particular configuration of the markers is used for ensuring marker detection, preventing occlusion and avoiding collisions with the support structure during the random exploration process. Due to the ability of the manipulator to extend and the dynamic nature of the motions, the workspace obtained is very large for its size (even considering the extension due to marker support).

The repeatability of the manipulator dynamics is also analyzed by providing a random actuation sequence for the module and then repeating the

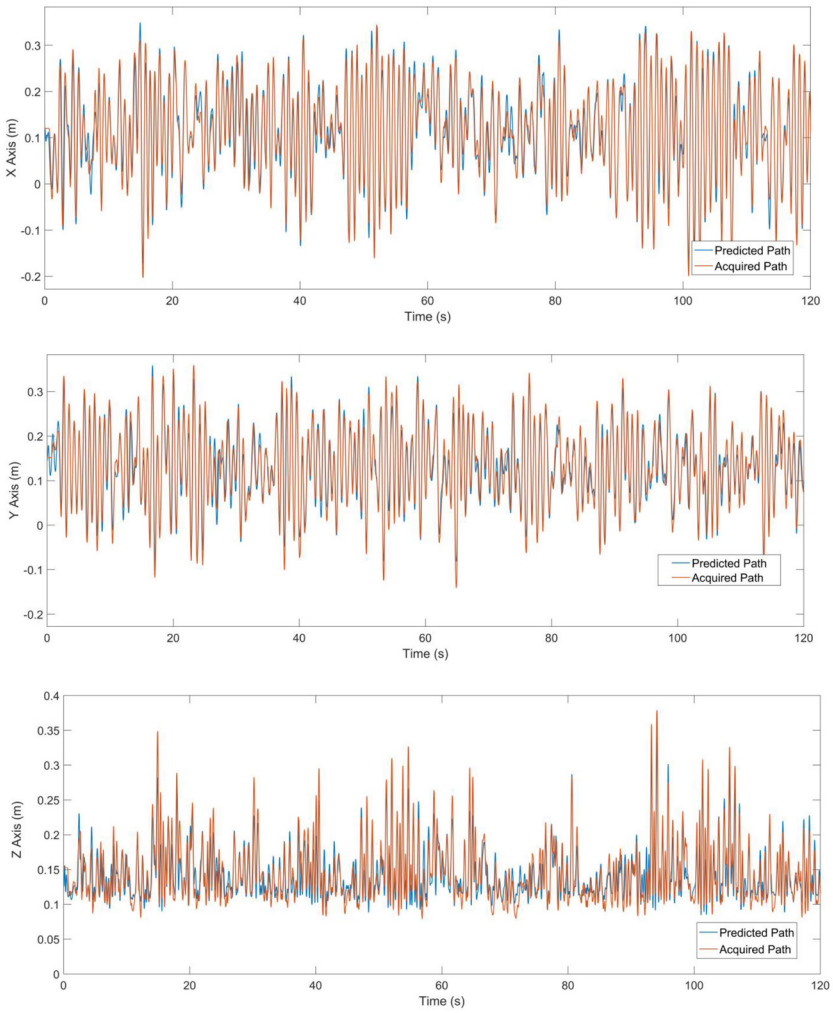

Figure 15. The predicted and actual paths for a single multi-step-ahead prediction using the learned recurrent network.

experiment 20 times. The average trajectory of the path along with its standard deviation is shown in figure 14. Although the manipulator is highly nonlinear and compliant, the manipulator motions are highly repeatable. When compared with the static case for the same manipulator, the motions are more repeatable [44]. The maximum deviation observed among all trials at any point in time was only 25 mm. This could be because for dynamic motions we deal with the much more linear dynamic friction effects and hysteresis effects are larger for the static case.

6.2. Learning forward model

The learning procedure is the same as in section 3.1. The hidden layer size is 40 and the states of the manipulator are measured from the three marker positions. Preprocessing of the sample data for missing markers and resampling of the data to 50 Hz is done before learning. Since the three markers are rigidly attached to the tip of the manipulator, the learned dynamic model can have a maximum of six DoF. Just as in the simulations, we analyze the performance of the learning using information from one, two and three markers. This corresponds to models with three, five and six DoF respectively. The learning performance for different dimensions of the task space is shown in table 6. The performance metric is evaluated as the prediction error in estimating the position of a common marker for a single multi step-ahead prediction for 240 s. The prediction versus data acquired for a duration of 120 s is shown in figure 15 for the six-DoF dynamic model. From the prediction performance it can be seen that although the predicted patterns are similar to the acquired values, the performance errors are significant. However, this is mainly due to phase differences between the actual and predicted values. This phenomenon was also observed in the experimental results. Another interesting observation is the absence of error accumulation even with large predicted errors in between for this long step-ahead prediction. This could be because of the training algorithm, which has to ensure that the error dynamics is stable for the sample data. On the other hand it could also due to the presence of dynamic attractors in certain regions of the state space of the manipulator.

6.3. Tests

Using the learned model and the trajectory optimization procedure described before, we conduct three experiments to validate the methodology. The first experiment is reaching a single static target at the

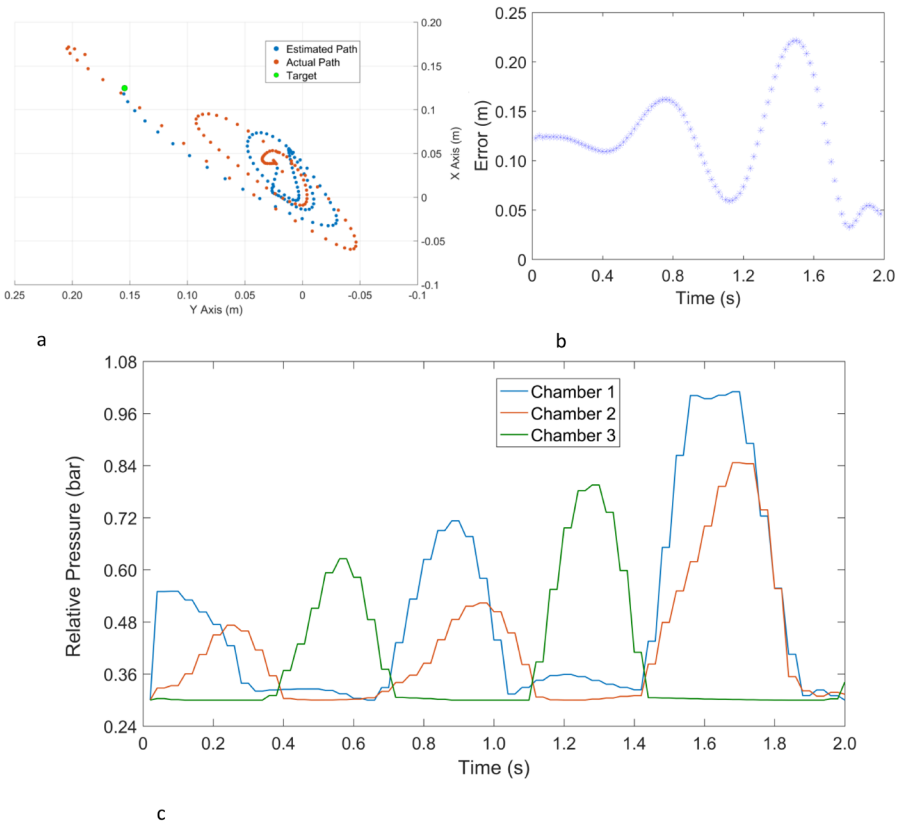

Figure 16. (a) An example trajectory of the end effector for the reaching task. (b) Average error for all 20 points during the reaching task over the control horizon. (c) The input signal to the chambers for an example case. Note that this need not be the actual pressure inside the chambers.

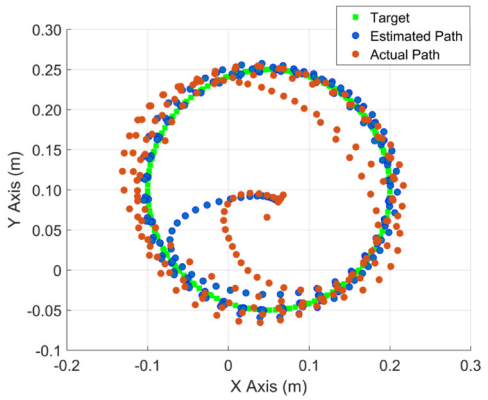

Figure 17. Trajectory of the end effector for the circular path task.

end of the control horizon and the second task involves following a continuous path. For all the tests the open loop controller works at 50 Hz with t_s set at 40 ms (see section 4). Lower bounds on pressure inputs are kept non-zero to reduce the effects of friction—this is because it was observed that the maximum prediction error occurred at the starting of the control cycle and it increased when the initial applied pressures were less.

6.3.1. *Dynamic reaching*

The first task involves reaching a target in the 3D Cartesian space picked randomly from the sampled workspace. The optimization is performed to reach the target at exactly 2 s. The average reaching error at the end of the control horizon was on average 0.046 m. With reference to the workspace dimensions this corresponds to an error of 4.8% in the *X*-axis, 3.2% in the *Y*-axis and 6.8% in the *Z*-axis. It was observed that maximum deviations from the predictions occurred during at the start of the task (when the manipulator is at rest). This was also observed during the learning phase. We believe this is primarily because of the static friction effects which are not easy to model and learn [41]. Also, in the sampled data the effects of static friction are only present in the initial step. Nonetheless, these estimation biases can be compensated by iterative learning techniques like the one used in [7] or with closed loop controllers. An example trajectory for reaching the target is shown in figure 16(a). The average tracking error for the whole time period is shown in figure 16(b). The average errors for all the reaching moments seem to show a smooth pattern. This indicates that there are some unmodeled effects which uniformly affect the motion of all the reaching movements and therefore can be easily compensated. A simple example would be to delay the control actions by 0.20 s which would on average improve the reaching performance to 0.032 m. The input pressures on the chambers for an example case are given in figure 16(c).

6.3.2. *Tracking path*

The second task involves following a continuous path. For this we define a circular trajectory of 0.15

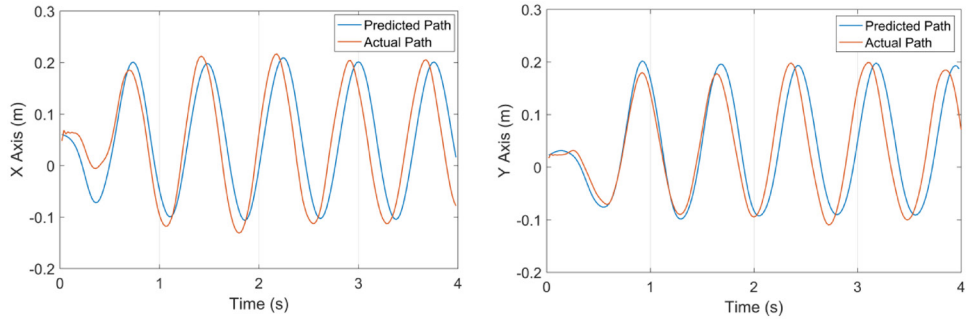

Figure 18. Estimated and actual path of the end effector in the tracking task.

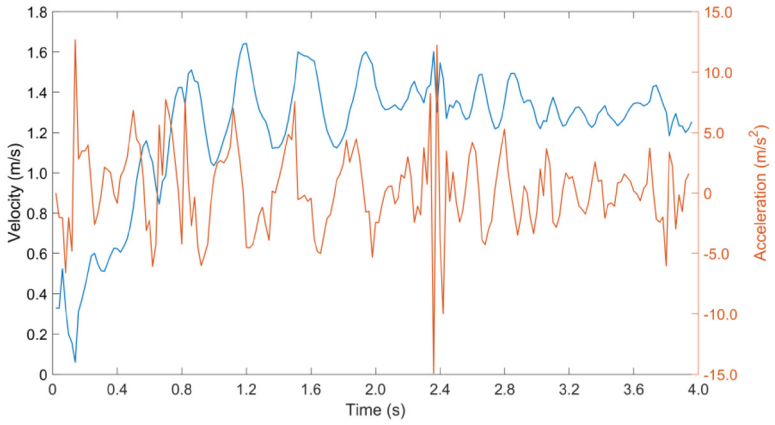

Figure 19. Velocity and acceleration of the end effector in the circular path task.

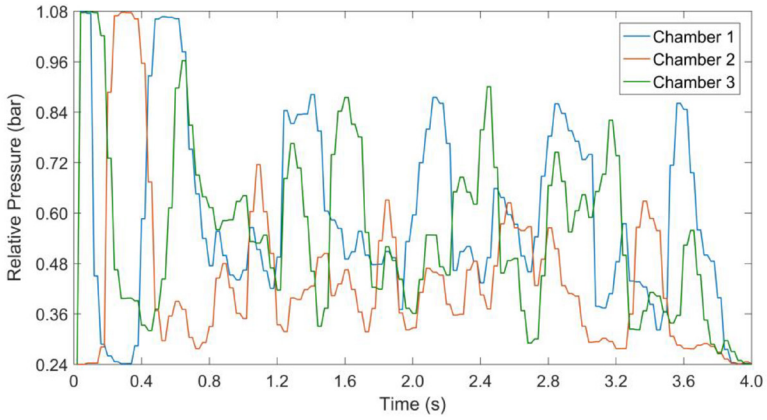

Figure 20. The input signal to the chambers for the circular task. The initial pressure is high since the manipulator starts from a stationary configuration.

m radius with a frequency of 1.25 Hz and a duration of 4 s. The optimization algorithm is formulated only to reduce the error in the X and Y coordinates. This is because we do not have an analytical model to calculate the dynamically reachable points. The target path for the end effector, the estimated path obtained from the optimization algorithm and the actual path, is shown in figure 17. The effects of static friction can be observed in apparent biasing between the estimated path and the observed path. However, this error does not accumulate over time and cause instability in tracking even though we use an open loop controller. The desired and actual paths in the X- and Y-axes are shown in figure 18 for comparison. The observed lag between the peaks of the predicted and observed paths between 40 and 80 ms (two to four time steps) for the whole duration. The observed velocities and acceleration of the tip during the task is shown in figure 19 and the input signals to the chamber are shown in figure 20. Note that the manipulator reaches a velocity of 1–1.5 m s^{-1} which corresponds to 2–3 cm/control step. Therefore, even a phase lag/lead of 1 time step (0.02 s) can lead to large tracking errors (videos of both the tasks are available in the supplementary materials, available at stacks.iop.org/BB/12/066003/mmedia).

7. Conclusion

This paper presents a novel approach for learning the forward dynamic model of soft robotic manipulators. With the help of a traditional trajectory optimization algorithm we show that this dynamic model can be used for open loop predictive control of the manipulator even for long control horizons. Since the approach is completely model free, there is no need to develop complex analytical models or make risky assumptions about the model. It also allows us to formulate a direct mapping from the input variables to the task space, thereby simplifying the development procedure. Since no intermediate actuator space/configuration space information is required, the method allows learning by only tracking the task space variables. Additionally, the methodology is scalable and can be applied to a wide range of continuum/soft robotic manipulator. For the single-section real manipulator used in the experiments, the sampling period was only 240 s, meaning that a complete dynamic model would be quickly learned from scratch without any assumptions. However, there are still some inaccuracies in the learned dynamic model. We believe that the major drawback to be addressed is the prediction of static friction effects. The effects were more prominent during the reaching task, where the constraints on the actuation efforts made the manipulator start slowly. To reduce this effect, we have kept the non-zero lower bounds on the pressure inputs thereby improving the prediction accuracy. However, this reduces the dynamic range of the manipulator. For the more dynamic circular task, the effects just resulted in an initial bias which remained consistent over time. A similar model-based implementation on a planar soft manipulator reached a static target of 4 cm diameter consistently, but with an additional iterative learning step for each targets [7]. Correspondingly, our approach can achieve similar accuracy but with a 3D manipulator and without any adaptation after the initial learning.

An open loop controller is good for evaluating the accuracy of the learned model since the performance of an open loop controller is as good as that of the model. However, as it can be seen from the experimental results, such an approach cannot correct for some biases incurred during tracking. Surely a closed loop approach would be the next step for a more robust controller, which would also need a good forward model. This could be addressed with techniques like model-based reinforcement learning. For example, in [32] control policies and value functions were approximated using nonparametric regression techniques. Reinforcement learning approaches for optimal controllers using learned dynamic model were used in [33, 34]. In [35], policies were represented with neural networks and generated by trajectory optimization. This could allow us to extend this approach to develop a closed loop predictive controller which would be more robust to external disturbances and modeling errors. That being said, open loop controllers can be very effective in particular tasks due to the large non-linearities of a soft robot. It is possible to formulate open loop controllers for closed trajectories that are self-stabilizing using only mechanical feedback [42] and these would be ideal as cheap industrial manipulators that can handle delicate materials and work in structured and unstructured environments. Another interesting modification of the current approach would be to incorporate static controllers in order to reach the final target statically.

In conclusion, we present a purely machine learning-based approach towards dynamic control of a soft robotic manipulator. Preliminary results are promising and demonstrate the advantages of data-driven methods and the varying behaviors possible with a dynamic controller for soft robots.

Acknowledgment

The authors would like to acknowledge the support by the European Commission through the ISUPPORT project (HORIZON 2020 PHC-19, #643666).

ORCID iDs

Thomas George Thuruthel https://orcid.org/0000-0003-0571-1672

References

[1] Renda F, Giorelli M, Calisti M, Cianchetti M and Laschi C 2014 Dynamic model of a multibending soft robot arm driven by cables *IEEE Trans. Robot.* **30** 1109–22
[2] Braganza D, Dawson D, Walker I and Nath N 2007 A neural network controller for continuum robots *IEEE Trans. Robot.* **23** 1270–7
[3] Xian B, Dawson D, DeQueiroz M and Chen J 2004 A continuous asymptotic tracking control strategy for uncertain nonlinear systems *IEEE Trans. Autom. Control* **49** 1206
[4] Kapadia A and Walker I 2011 Task-space control of extensible continuum manipulators *2011 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*
[5] Kapadia A, Walker I, Dawson D and Tatlicioglu E 2010 A model-based sliding mode controller for extensible continuum robots *Proc. of the 9th WSEAS Int. Conf. on Signal Processing, Robotics and Automation* pp 113–20
[6] Kapadia A, Fry K and Walker I 2014 Empirical investigation of closed-loop control of extensible continuum manipulators *2014 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*
[7] Marchese A, Tedrake R and Rus D 2015 Dynamics and trajectory optimization for a soft spatial fluidic elastomer manipulator *Int. J. Robot. Res.* **35** 1000–19
[8] Falkenhahn V, Hildebrandt A, Neumann R and Sawodny O 2015 Model-based feedforward position control of constant curvature continuum robots using feedback linearization *2015 IEEE Int. Conf. on Robotics and Automation (ICRA)*
[9] Falkenhahn V, Hildebrandt A, Neumann R and Sawodny O 2016 Dynamic control of the bionic handling assistant *IEEE/ASME Trans. Mechatron.* **22** 6–171
[10] Best C, Gillespie M, Hyatt P, Rupert L, Sherrod V and Killpack M 2016 A new soft robot control method: using

model predictive control for a pneumatically actuated humanoid *IEEE Robot. Autom. Mag.* **23** 75–84
[11] Nguyen-Tuong D, Seeger M and Peters J 2009 Model learning with local Gaussian process regression *Adv. Robot.* **23** 2015–34
[12] Choi Y, Cheong S-Y and Schweighofer N 2007 Local online support vector regression for learning control *2007 Int. Symp. on Computational Intelligence in Robotics and Automation*
[13] Bhushan N and Shadmehr R 1998 Evidence for a forward dynamics model in human adaptive motor control *Proc. of the 11th Int. Conf. on Neural Information Processing Systems (NIPS'98)* ed M J Kearns *et al* (Cambridge, MA: MIT Press) pp 3–9
[14] Wolpert D M and Ghahramani Z and Jordan M I 1995 Forward dynamic models in human motor control: psychophysical evidence *Advances in Neural Information Processing Systems* (*Bradford Series* vol 7) (Cambridge, MA: MIT Press) pp 43–50
[15] Atkeson C G, Jordan M K M I and Solla S 1998 Nonparametric model-based reinforcement learning *Advances in Neural Information Processing Systems* vol 10 (Cambridge, MA: MIT Press) pp 1008–14
[16] Bagnell J and Schneider J 2001 Autonomous helicopter control using reinforcement learning policy search methods *IEEE Int. Conf. Robot. Autom.* **2** 1615–20
[17] Deisenroth M P, Englert P, Peters J and Fox D 2014 Multi-task policy search for robotics *IEEE Int. Conf. on Robotics and Automation* (IEEE) pp 3876–81
[18] Polydoros A S and Nalpantidis L 2016 A reservoir computing approach for learning forward dynamics of industrial manipulators *2016 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*
[19] Kocijan J, Murray-Smith R, Rasmussen C and Girard A 2004 Gaussian process model based predictive control *Proc. American Control Conf.*
[20] Nguyen-Tuong D, Seeger M and Peters J 2008 Computed torque control with nonparametric regression models *2008 American Control Conf.*
[21] Sigaud O, Salaün C and Padois V 2011 On-line regression algorithms for learning mechanical models of robots: a survey *Robot. Auton. Syst.* **59** 1115–29
[22] Nguyen-Tuong D and Peters J 2011 Model learning for robot control: a survey *Cogn. Process.* **12** 319–40
[23] Renda F *et al* 2016 Discrete Cosserat approach for soft robot dynamics: a new piece-wise constant strain model with torsion and shears *2016 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS) (30 January 2017)* (https://doi.org/10.1109/IROS.2016.7759808)
[24] Mihajlov M, Ivlev O and Gräser A 2008 Modelling and identification for control design of compliant fluidic actuators with rotary elastic chambers: hydraulic case study *17th IFAC World Congress (Seoul, Korea, 6–11 July 2008)*
[25] Boyer F and Renda F 2017 Poincaré's equations for Cosserat media: application to shells *J. Nonlinear Sci.* **27** 1
[26] Edwards C H and Penney D E 2013 *Differential Equations and Linear Algebra Always Learning* (London: Pearson Education)
[27] Thuruthel T, Falotico E, Cianchetti M and Laschi C 2016 Learning global inverse kinematics solutions for a continuum robot *ROMANSY 21—Robot Design, Dynamics and Control* pp 47–54
[28] Thuruthel T, Falotico E, Cianchetti M, Renda F and Laschi C 2016 Learning global inverse statics solution for a redundant soft robot *Proc. of the 13th Int. Conf. on Informatics in Control, Automation and Robotics*
[29] Menezes J and Barreto G 2008 Long-term time series prediction with the NARX network: an empirical evaluation *Neurocomputing* **71** 3335–43
[30] Diaconescu E 2008 The use of NARX neural networks to predict chaotic time series *WSEAS Trans. Comp. Res.* **3** 182–91
[31] Fletcher R 1987 *Practical Methods of Optimization* 1st edn (Chichester: Wiley)
[32] Atkeson C G and Morimoto J 2003 Nonparametric representation of policies and value functions: a trajectory-based approach *Advances in Neural Information Processing Systems* vol 15, eds Becker Thrun S S and Obermayer K (Cambridge, MA: MIT Press) pp 1611–18
[33] Abbeel P, Coates A, Quigley M, Ng A Y, Schlkopf B, Platt J and Hoffman T 2007 An application of reinforcement learning to aerobatic helicopter flight *Advances in Neural Information Processing Systems* vol 19 (Cambridge, MA:MIT Press) pp 1–8
[34] Ng A Y, Coates A, Diel M, Ganapathi V, Schulte J, Tse B, Berger E and Liang E 2004 Autonomous inverted helicopter flight via reinforcement learning *Proc. of the 11th Int. Symp. on Experimental Robotics*
[35] Levine S and Koltun V 2014 Learning complex neural network policies with trajectory optimization *Int. Conf. on Machine Learning (ICML)*
[36] Selig J M 2007 Geometric fundamentals of robotics *Monographs in Computer Science* (Berlin: Springer) (https://doi.org/10.1007/b138859)
[37] Renda F, Boyer F, Dias J and Seneviratne L 2017 Discrete Cosserat approach for multi-section soft robots dynamics arXiv:1702.03660 [cs.RO]
[38] Trivedi D, Lotfi A and Rahn C 2007 Geometrically exact dynamic models for soft robotic manipulators *2007 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*
[39] Renda F, Cianchetti M, Abidi H, Dias J and Seneviratne L 2017 Screw-based modeling of soft manipulators with tendon and fluidic actuation *ASME. J. Mech. Robot.* **9** 041012
[40] Rucker D C, Jones B A and Webster R J III 2010 A geometrically exact model for externally loaded concentric tube continuum robots *IEEE Trans. Robot.* **20** 769–80
[41] Olsson H, Åström K, Canudas de Wit C, Gäfvert M and Lischinsky P 1998 Friction models and friction compensation *Eur. J. Control* **4** 176–95
[42] Plooij M, Wolfslag W and Wisse M 2014 Open loop stable control in repetitive manipulation tasks *2014 IEEE Int. Conf. on Robotics and Automation (ICRA) (Hong Kong)* pp 949–56
[43] Manti M, Pratesi A, Falotico E, Cianchetti M and Laschi C 2016 Soft assistive robot for personal care of elderly people *2016 6th IEEE Int. Conf. on Biomedical Robotics and Biomechatronics (BioRob) (26 June 2016)* (IEEE) pp 833–8
[44] George Thuruthel T, Falotico E, Manti M, Pratesi A, Cianchetti M and Laschi C 2017 *Soft Robot.* (https://doi.org/10.1089/soro.2016.0051)