Learning Dynamics and Trajectory optimization for Octopus Inspired Soft Robotic Manipulators

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]*KURI Institute, Khalifa University, Abu Dhabi, UAE*

{t.thuruthel, e.falotico, cecilia.laschi}@santannapisa.it, federico.renda@kustar.ac.ae

Abstract:

The soft capabilities of biological appendages like the arms of *Octopus Vulgaris* and Elephant trunk has inspired roboticists to develop their robotic equivalents. Although there have been considerable efforts to replicate their morphology and behaviour patterns, we are still lagging behind in 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 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 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 done on an octopus inspired soft manipulator simulation derived from a piecewise constant strain approximation.

1. INTRODUCTION

Development of dynamic controllers for soft robotics application is a relatively unexplored area even with the rising progress and applications of soft robotics technologies. This is mainly because of the complexity involved with 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 manipulator capabilities. 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 error prone and parameter sensitive. 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 dynamic system and highly robust and well thought optimization techniques.

Due to the complexity involved with 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 for closed loop control in the joint space. The control architecture is 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 is used to compensate for the dynamic uncertainties and thereby reducing the uncertainty bound that improves the performance of the feedback controller.

One of the first model-based approach for dynamics controllers used the 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 experimental evaluation of a sliding mode controller for closed loop configuration space control was proposed in [6], however only on a planar manipulator. Comparisons to a simple feedback linearization based PD controller in the configuration space was 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 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 is 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 dependency 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. The performance of such reactive controllers for high frequency motion is still questionable considering the fact that model uncertainties are unavoidable and the slow response of soft systems. Another interesting work using model predictive controllers was described in [10]. Due to their unique manipulator design the kinematics and the dynamics of the system could be written similar to the case of rigid robots. The advantage of solving the control problem as an optimization problem is that it alleviates the need for a high level path planner.

This paper proposes a methodology for learning forward dynamic models of a soft manipulator and to use this learned model for developing open loop predictive controllers. 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 soft manipulator developed based on the piecewise constant strain approximation, we try to demonstrate that learning based controllers are easier to develop, apply and transferable, while being accurate at the same time.

1.1 SIMILAR WORKS ON RIGID ROBOTS

Contrary to the case of soft robots, learning forward or inverse dynamics of rigid robots have 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 additional burden in terms of specifying the desired state trajectories. Considering the response delay of soft systems and taking 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 with high dimensional systems. Some of the most popular methods use learning approaches like LWPR [15], LWBR [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. 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 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 in particular and probably even superior to 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 us to develop arbitrarily complex models of the manipulator from data and 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 controllers can be successfully implemented even with long control horizons. This is also the first demonstration of the usage of a learned dynamic controller in task space for a soft or continuum robotic manipulator and also the only implementation on a three dimensional manipulator.

2. SIMULATION SETUP

2.1 Model Description

The simulation setup is built on an advanced dynamic model for soft robotics recently developed based on a discrete Cosserat formulation of the soft robot arm dynamics. In [23, 37], the continuous Cosserat model for soft robotics previously presented in [24] has been discretized by assuming a piecewise constant strain (PCS) condition, which allows an analytical integration of the continuous kinematics and dynamic equations. This led to a discrete formulation of the soft robots dynamics that turns out to be the geometrically-equivalent generalization of the traditional rigid robotics dynamics. From a simulation point of view, the main features of the model are the following:

- Full multi-section dynamics capable of predicting the motion of complex soft robot arms.
- Piecewise Constant Strain assumption allowing for a full 6 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. 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 $g(\cdot): X \mapsto g(X) \in SE(3)$ 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 $g(\cdot)$ given by $X \mapsto \hat{\xi}(X) = g^{-1}\partial g/\partial X = g^{-1}g' \in \mathfrak{se}(3)$, 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 $g(\cdot)$ is represented by the twist vector field $X \mapsto \eta(X) \in \mathbb{R}^6$ defined by $\hat{\eta}(X) = g^{-1}\partial g/\partial t = g^{-1}\dot{g}$.

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

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

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

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

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 $\mathrm{ad}^* = -\mathrm{ad}^T$.

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

$$\mathcal{M}\dot{\eta} + \mathrm{ad}_{\eta}^{*}(\mathcal{M}\eta) = \mathcal{F}_i{}' + \mathrm{ad}_{\xi}^{*}\mathcal{F}_i + \bar{\mathcal{F}}_a + \bar{\mathcal{F}}_e \tag{4}$$

Where $\mathcal{F}_i(X) = \frac{\partial \mathfrak{U}}{\partial \mathrm{X}}$ is the wrench of internal forces, $\bar{\mathcal{F}}_a(X,t)$ is the distributed actuation loads, $\bar{\mathcal{F}}_e(X)$ is the external wrench of distributed applied forces and $\mathcal{M}(X)$ is the screw inertia matrix. A linear viscoelastic constitutive model has been chosen for the internal wrench: $\mathcal{F}_i(X) = \Sigma(\xi - \xi^0) + \Upsilon\dot{\xi}$, where Σ and Υ are constant screw stiffness and viscosity matrices and ξ^0 is the strain filed of the reference configuration. Regarding the distributed actuation load, in the case of cable driven actuation, we have: $\bar{\mathcal{F}}_a(X,t) = -\mathcal{F}_a' - \mathrm{ad}_\xi^*\mathcal{F}_a$, where $\mathcal{F}_a$ 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:

$$\bar{\mathcal{F}}_e(X) \ = \left(1 - \frac{\rho_w}{\rho}\right)\mathcal{M}\mathrm{Ad}^{-1}_{g_r g(X)}\mathcal{G} - \mathcal{D}|\eta|_v\eta \tag{5}$$

$$\mathcal{M}_a = \mathcal{M} + \mathcal{A}. \tag{6}$$

Where $|\cdot|_v$ takes the norm of the translational part of the operand, ρ_w is the water density, $\mathcal{G}$ is the gravity twist, g_r 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 $\mathcal{M}_a$ in (dynamics equation) allows modeling inertial hydrodynamics forces exerted along the arm. Finally, we have introduced the Adjoint representation (Ad) of Lie group $\mathrm{SE}(3)$, while the coAdjoint map is defined by $\mathrm{Ad}^* = \mathrm{Ad}^{-T}$.

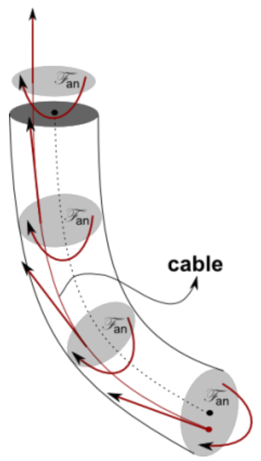


Figure 1. Depiction of the tendon actuation for one section.

The discretization of the continuous model outlined above is achieved through the piecewise constant strain assumption. At any instant t, considering the strain field ξ 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 ξ_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 and the matrix differential equation 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 [37], we obtain the discrete kinematics equation:

$$g(X) = g(L_{n-1})e^{(X-L_{n-1})\hat{\xi}_n} = g(L_{n-1})g_n(X) \tag{7}$$

$$\eta(X) = \mathrm{Ad}_{g_n(X)}^{-1}\big(\eta(L_{n-1}) + \mathrm{AD}_{g_n}(X)\dot{\xi}_n\big) \tag{8}$$

$$\dot{\eta}(X) = \mathrm{Ad}_{g_n(X)}^{-1}\left(\dot{\eta}(L_{n-1}) - \mathrm{ad}_{\mathrm{AD}_{g_n}\dot{\xi}_n}\eta(L_{n-1}) + \mathrm{AD}_{g_n}(X)\ddot{\xi}_n\right) \tag{9}$$

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

$$g_n(X) = e^{x\hat{\xi}_n} = I_4 + x\hat{\xi}_n + 1/\theta_n^2\,(1-\cos(x\theta_n))\hat{\xi}_n^2 + 1/\theta_n^3\,(x\theta_n - \sin(x\theta_n))\hat{\xi}_n^3 \tag{10}$$

$$\begin{aligned}\mathrm{Ad}_{g_n(X)} \quad &= e^{x\mathrm{ad}_{\xi_n}}\\ &= I_6 + 1/2\theta_n\,(3\sin(x\theta_n) - x\theta_n\cos(x\theta_n))\mathrm{ad}_{\xi_n}\\ &+ 1/2\theta_n^2\,(4 - 4\cos(x\theta_n) - x\theta_n\sin(x\theta_n))\mathrm{ad}_{\xi_n}^2\\ &+\ 1/2\theta_n^3\,(\sin(x\theta_n) - x\theta_n\cos(x\theta_n))\mathrm{ad}_{\xi_n}^3\\ &+ 1/2\theta_n^4\,(2 - 2\cos(x\theta_n) - x\theta_n\sin(x\theta_n))\mathrm{ad}_{\xi_n}^4\end{aligned} \tag{11}$$

$$\begin{aligned}\mathrm{AD}_{g_n}(X) \quad &= \int_{L_{n-1}}^{X}\mathrm{Ad}_{g_n(s)}ds\\ &= xI_6 + 1/2\theta_n^2\,(4 - 4\cos(x\theta_n) - x\theta_n\sin(x\theta_n))\mathrm{ad}_{\xi_n}\\ &+\ 1/2\theta_n^3\,(4x\theta_n - 5\sin(x\theta_n) + x\theta_n\cos(x\theta_n))\mathrm{ad}_{\xi_n}^2\\ &+ 1/2\theta_n^4\,(2 - 2\cos(x\theta_n) - x\theta_n\sin(x\theta_n))\mathrm{ad}_{\xi_n}^3\\ &+ + 1/2\theta_n^5\,(2x\theta_n - 3\sin(x\theta_n) + x\theta_n\cos(x\theta_n))\mathrm{ad}_{\xi_n}^4\end{aligned} \tag{12}$$

With θ_n is the norm of the rotational part of the constant strain ξ_n and $x = X - L_{n-1}$. A schematics of the piece-wise constant strain kinematics is shown in figure 2.

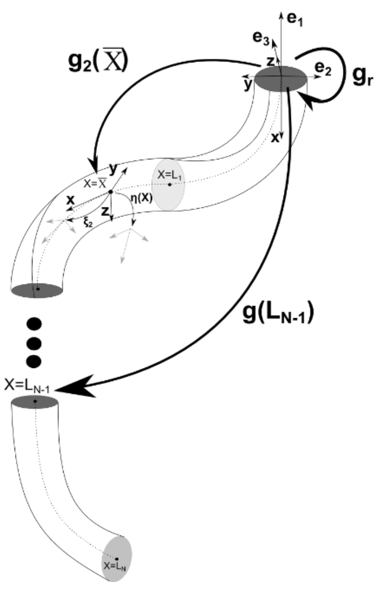


Figure 2. Schematic of the kinematics of the piece-wise constant strain model.

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

$$\eta(X) = \sum_{i=1}^{n} \left(\prod\nolimits_{j=n}^{i} \mathrm{Ad}^{-1}_{g_j(\min(L_j, X))} \right) \mathrm{AD}_{g_i}(\min(L_i, X)) \dot{\xi}_i = \sum_{i=1}^{n} S_i(X) \dot{\xi}_i = J(X) \dot{\vec{\xi}} \tag{13}$$

Where we have defined the softs robot geometric Jacobian $J(X) \in \mathbb{R}^6 \otimes \mathbb{R}^{6N}$ and the soft robots joint vector $\vec{\xi} = [\xi_1^T,\ \xi_2^T,\ \cdots,\ \xi_N^T]^T \in\ \mathbb{R}^{6N}$. Similarly, by taking the time derivative of (velocity Jacobian equation), equation (discrete acceleration equation) can be written as:

$$\dot{\eta}(X) = J(X) \ddot{\vec{\xi}} + \dot{J}(X) \dot{\vec{\xi}} \tag{14}$$

At this point, we are able to obtain the discrete Cosserat dynamics by projecting the continuous dynamics (continuous dynamics) into the joint space with the Jacobian transpose J^T, substituting the discrete model of velocity (velocity Jacobian equation) and acceleration (acceleration Jacobian equation) and integrating over the different piece of the soft arm. Mathematically, we obtain:

$$M(\vec{\xi}) \ddot{\vec{\xi}} + \left(C_1 \left(\vec{\xi}, \dot{\vec{\xi}} \right) - C_2 \left(\vec{\xi}, \dot{\vec{\xi}} \right) \right) \dot{\vec{\xi}} = \vec{\tau}(\vec{\xi}\,) + N(\vec{\xi}) \mathrm{Ad}_{g_r}^{-1} \mathcal{G} - D \left(\vec{\xi}, \dot{\vec{\xi}} \right) \dot{\vec{\xi}} \tag{15}$$

Where we recognize the structure of the Lagrangian model of rigid serial manipulators and we have introduced the soft robotics generalized actuation vector: $\vec{\tau} = [\tau_1^T,\ \tau_2^T, \cdots,\ \tau_N^T]^T\ \in\ \mathbb{R}^{6N}$. The different terms of (soft Lagrangian dynamics) are specified below block-element-wise [37].

$$M_{(n,m)} = \sum_{i=max(n,m)}^{N} \int_{L_{i-1}}^{L_i} S_n^T \mathcal{M}_a S_m \, dX \tag{16}$$

$$C_{1_{(n,m)}} = \sum_{i=max(n,m)}^{N} \int_{L_{i-1}}^{L_i} S_n^T \mathrm{ad}^*_{J\dot{\vec{\xi}}} \mathcal{M}_a S_m \, dX \tag{17}$$

$$C_{2_{(n,m)}} = \sum_{i=max(n,m)}^{N} \int_{L_{i-1}}^{L_i} S_n^T \mathcal{M}_a \mathrm{ad}_{\sum_{j=m+1}^{i} S_j \dot{\xi}_j} S_m \, dX \tag{18}$$

$$D_{(n,m)} = \sum_{i=max(n,m)}^{N} \int_{L_{i-1}}^{L_i} S_n^T \mathcal{D} S_m \left| J \dot{\vec{\xi}} \right|_v dX \tag{19}$$

$$N_{(n)} = (1 - \rho_w / \rho) \sum_{i=n}^{N} \int_{L_{i-1}}^{L_i} S_n^T \mathcal{M} \mathrm{Ad}_g^{-1} \, dX \tag{20}$$

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

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

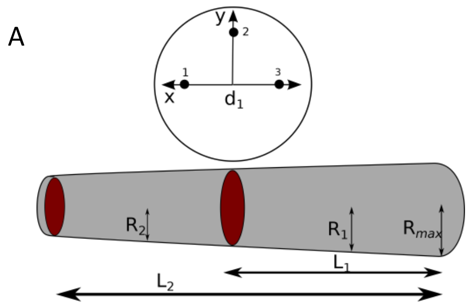
Where $\mathcal{F}_{an}$ indicates the contribution of the cables attached at L_n (Figure 1) and $\mathcal{F}_{in}$ is the constant internal load of the section n.

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

2.2 Soft Arm Design Description

The simulated prototype is shown in figure (prototype) and described in [24], to which we refer to 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 degrees (Figure (prototype)). 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 the 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.



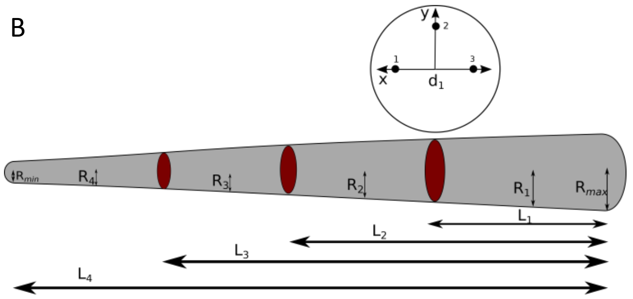


Figure 3. Schematic of the soft manipulator used in the simulation. (a) Two section manipulator (b)Four section manipulator

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 gr	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

$\boldsymbol{L_4}$	418 mm	Added Mass Coef. y B_y	1.5
Young Modulus $\boldsymbol{E}$	110 K Pa	Added Mass Coef. z B_z	1.5
Shear Viscosity Modulus $\boldsymbol{\mu}$	300 Pa sec	ρ_w	1.02 kg/dm^3
Poisson Ratio $\boldsymbol{\nu}$	0.5	ρ	1.08 kg/dm^3

3. LEARNING THE FORWARD DYNAMICS

Assume that the infinite dimensional configuration space can be approximated using an n-dimensional state space. The kinematics of the manipulator can now be represented as:

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

Where, x is the task space variable. In order to obtain the inverse kinematics, a necessary 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 x.

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

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

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

Here, $\tau \in \mathbb{R}^m$ are the control inputs. M, V, P represents the inertia matrix, Centripetal-Coriolis forces and potential energy stored due to gravity/deformation respectively. $\bar{M}, \bar{V}, \bar{P}$ are the corresponding matrixes 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 (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 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. For rigid robots, the dimension of the joint space (equivalently to the configuration space) is finite and therefore the number of sensors required is fixed.

Taking cue from our previous works on learned controllers [27, 28], we modify the mapping in terms of absolute values, there obtaining the new mapping: $(\boldsymbol{\tau^c}, \boldsymbol{x^p}, \boldsymbol{x^c}) \rightarrow \boldsymbol{x^n}$, 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. Another way to see it is that the current acceleration is a function of the previous, current and next position values and the current velocity is a function of the previous and current position. 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 (see figure 4). The advantages of a

NARX network for long-term time series prediction has been widely discussed [29, 30]. Additionally, representing the dynamic model using only absolute terms also helps in encoding the boundary conditions in the data which further helps in the stability of the network. Finally, this way we can avoid taking first order and second order derivatives of the position term which would increase the variance of any noise present. 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

For developing a multi-step prediction model for the forward dynamics we are using a nonlinear autoregressive network with exogenous inputs (NARX). The advantage of using a recurrent-dynamic network over a recursive feedforward-dynamic network is twofold; recurrent 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 thereby prone to overfitting and instability) and the prediction is slightly faster. In the MATLAB implementation of the NARX network a single second prediction takes 41 milliseconds while the same prediction takes 47 milliseconds using a recursive open loop multilayer perceptron. The architecture of the dynamic model with the recurrent network is shown in Figure 4. Note that the inputs are normalized inside the network. The network has a single hidden 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.

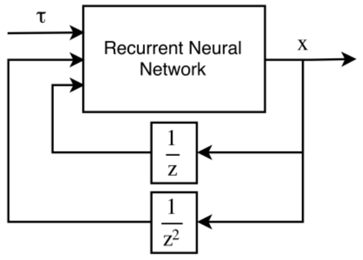


Figure 4. The architecture of the dynamic model using the NARX network.

3.1.1 Sampling and Training

For the purpose of this paper samples for learning is 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 percent 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 Newtons. Sampling is done at a fixed frequency of 100 Hz and consists of 7000 samples, mounting to a duration of 70 seconds. The corresponding explored workspace is shown in Figure 5. Additionally, more goal directed explorations can be done after learning the forward dynamics with the initial sample for more efficient and complete exploration.

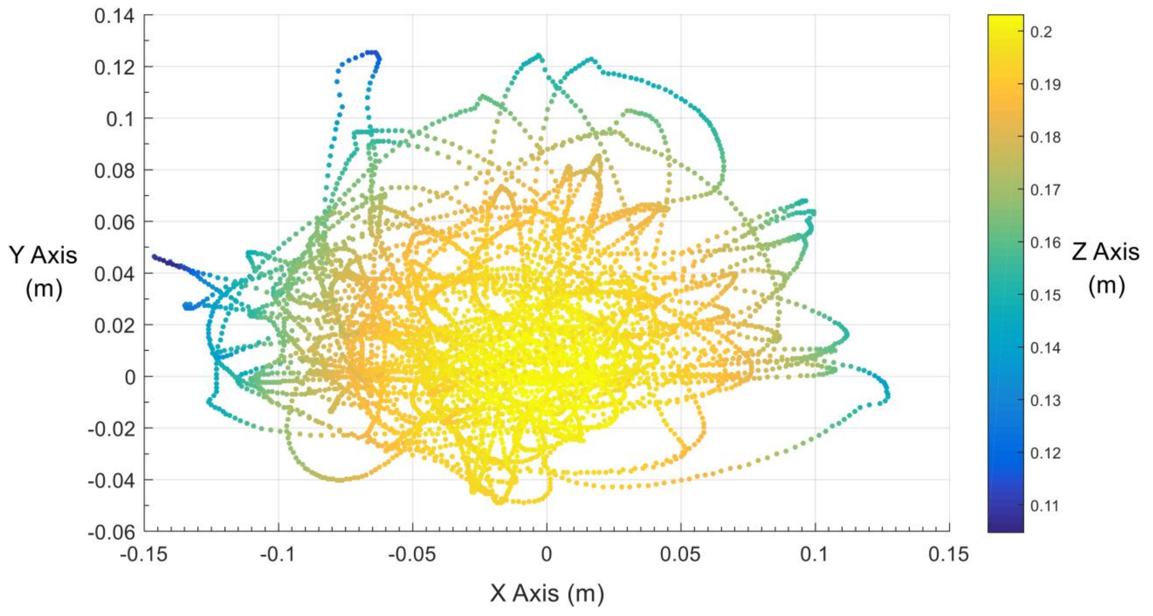


Figure 5. Workspace of the manipulator obtained by the random exploration.

Training of the network is done in two steps. Initially, the network is training 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:

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

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 set is divided into continuous (to keep time correlations intact) blocks in the ratio 70:30 for the first step and a training, testing and validation set in the ration 70:15:15 for final step. The performance function is now represented as:

$$\mathrm{MSE} = \frac{1}{T}\sum_{t=0}^{T}\left\|X_t - f(\hat{X}_{t-1}, U_t)\right\|^2 \tag{26}$$

Here, $\hat{X}$ 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 next section describes how the number of task space variables is decided.

3.1.2 Deciding task space dimension

As mentioned before, the dimension of task variables that determine the underlying dynamic model is up to the user to decide. Clearly, more the information provided about the state of the manipulator, better would be the prediction. The mean prediction error for different number of task space parameters are shown in Figure 6. The prediction is done by the recurrent network for the whole sample data by a single multi-step simulation (A 70 second simulation). For the three dimensional case only the Cartesian position of the end effector is used for prediction. For the six dimension case two scenarios are compared; the first one uses the Cartesian position of the tip of both sections and the second case uses only the Cartesian positions of the second section tip (end effector) and mid-section (can be replaced by the orientation of the tip). For the twelve dimensional case Cartesian positions of each sections tip and mid-section is used. For all the cases, the network size is fixed (35 neurons).

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 increase the chaoticity of manipulator dynamics and thereby deteriorates the prediction accuracy. For all cases better accuracy can be obtained by increasing the task space dimension.

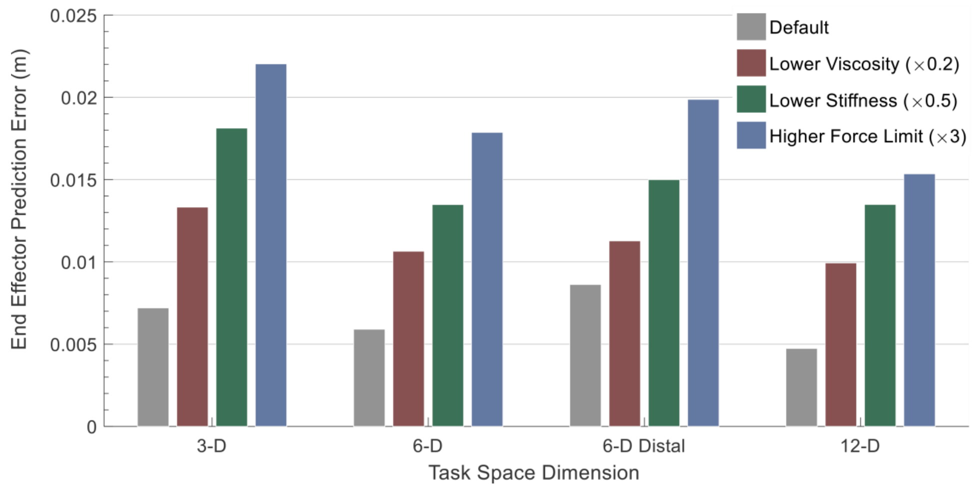


Figure 6. Mean multistep prediction error using the NARX network for different manipulator characteristics.

The time evolution of prediction error for the single 70 second 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 twelve dimensional case. The stability advantages of the NARX network over the open loop network obtained from the first learning can be seen in Figure 8, where the inputs forces are all set to zero. The error accumulation problem causes the open loop network to become highly unstable even for this simple case.

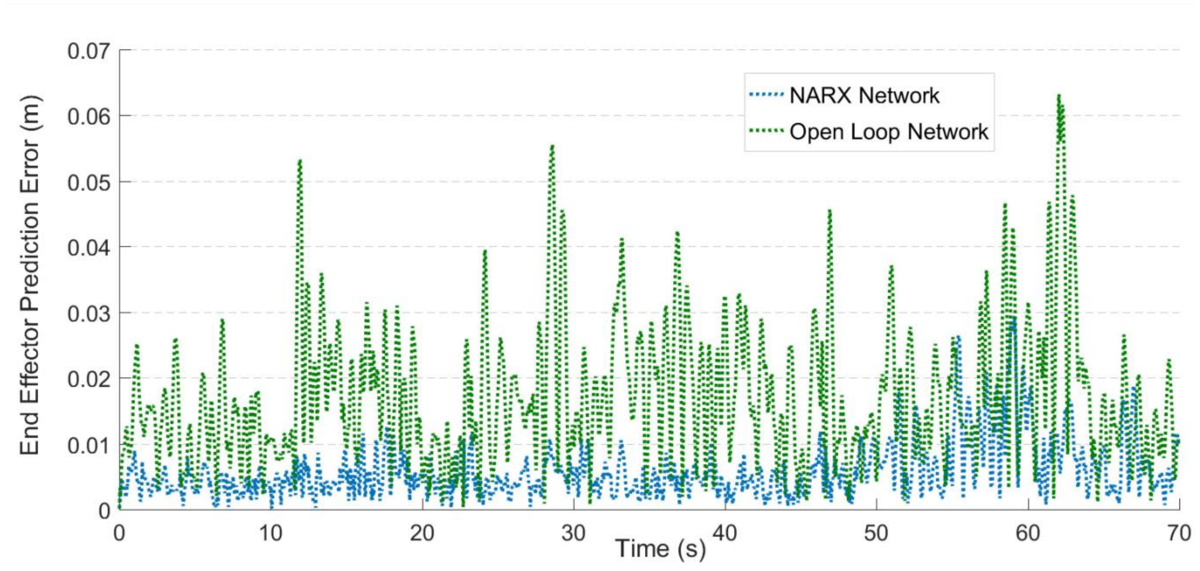


Figure 7. Time evolution of the multistep prediction error for the recurrent network and open loop network.

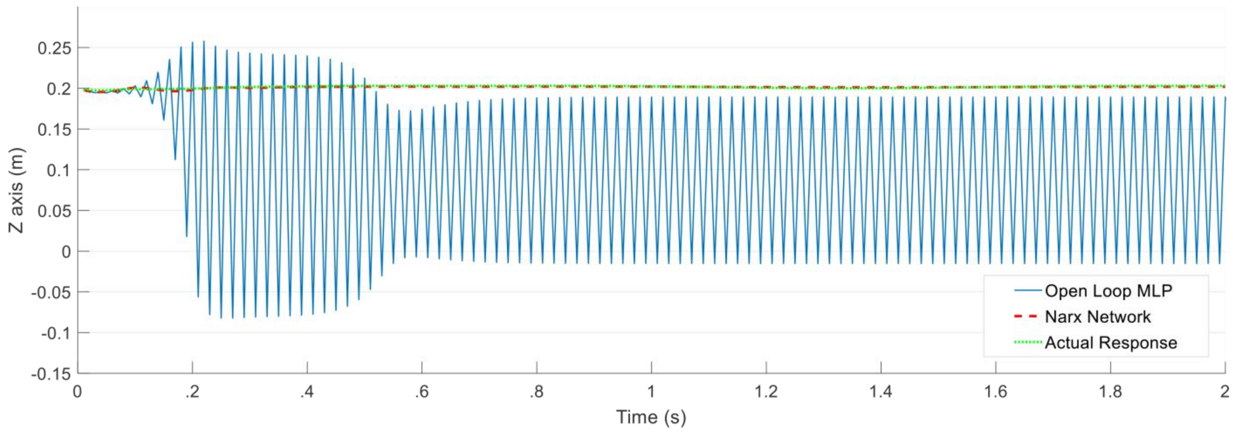


Figure 8. Instability of the open loop network for a zero actuation case. The response of the recurrent network and the simulation is given for reference.

Although the learned dynamic model may not be as accurate as a detailed analytical formulation, the recurrent neural network runs much faster. A two second simulation of the forward model takes 63 milliseconds using the recurrent neural network, whereas, the same simulation using the PCS model takes 523 seconds.

4. TRAJECTORY OPTIMIZATION

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

Let the fixed control horizon be t_f discretised by a fixed step size of $\mathrm{d}t$ (10 millisecond 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. 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 reducing the control frequency to $1/t_s$ (t_s is 50 millisecond in our case). The control inputs for each time step $\mathrm{d}t$ to can now be written as:

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

Here, M is the number of actuators and t is the current time. Note that there are other ways to reduce the dimensionality of the optimization problem which ensure smoother transition of the control inputs (by linear or polynomial interpolation), however since we are working only on simulations, this is not necessary. The time dependent control policy is represented by the low dimensional vector $\bar{\tau}$:

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

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

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

The control objective is formulated to reach a desired position as the end of the control horizon while simultaneously optimizing the control effort.

For solving 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. 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. Running on a computer with Intel(R) Core(TM) i7-3630QM CPU @ 2.40 GHz and 8 Gb RAM using parallel processing on 4 cores, a 10 step iteration for a control horizon of 1 second (60 variables) takes 6.2 seconds in average. This is not fast enough for implementing 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 controller 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 seconds) with minimal control effort. The actuators forces are limited to 2.5 Newtons and the optimization algorithm is run for 20 iterations. The results are summarized in Table 2. The optimization on average takes 25 seconds.

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

The simulations conducted next are done so to showcase four important characteristics that we consider important. The first simulation is to showcase 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, which is simulated by adding artificial noise to the sample data. The third simulation is 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 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, they 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 forces applicable by the cable are also limited to 1 Newton. 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 cables (shown in brackets in the figure) and letting the manipulator stabilize for 10 seconds. For showing 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 seconds. The predicted boundaries of the manipulator are shown in Figure 9(a) with the actual end effector position obtained using the numerical simulation using 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.

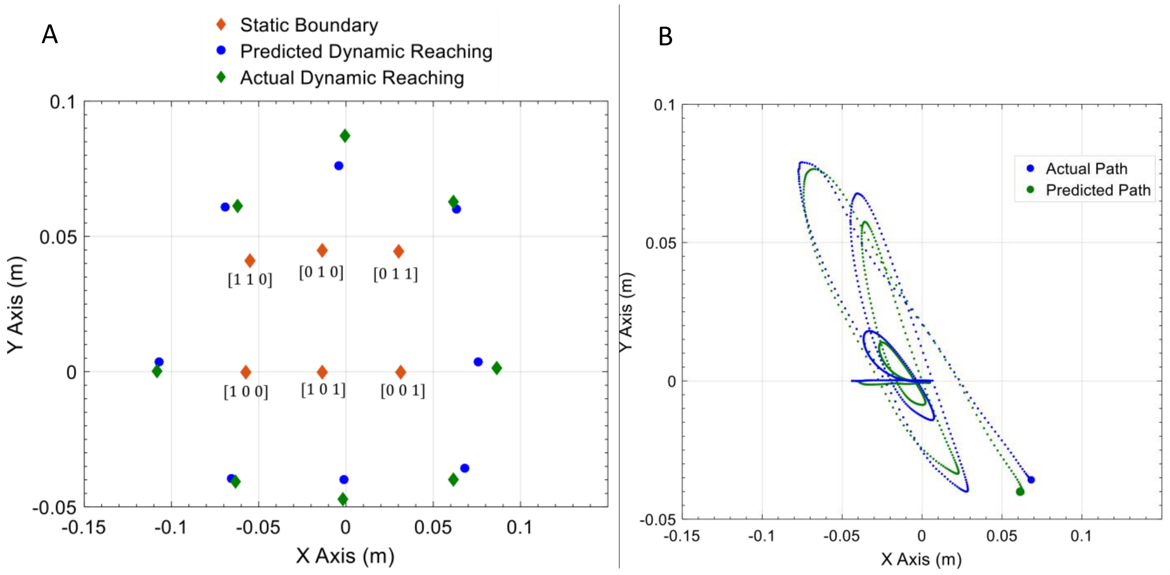


Figure 9(a). Static reachable boundaries of the manipulator and the reachability of the manipulator with a dynamic controller. (b) Illustration of the complex path the manipulator takes to reach one example target.

The average errors and standard deviation of the controller in the reaching task is shown in Table 3 for the eight reaching targets. Since the target points are handpicked and since the forces and time horizon is limited, the errors are not true indicators of the performance of the learned model or the trajectory optimization algorithm. The validity of the learned model can be seen from the difference between the prediction and simulation positions.

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

5.2 Effect of Noise

One of the main concerns of modelling 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 extend 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 0.005m is added to the task space positions obtained from the simulation. Since the representation of the network in only in absolute coordinates, we do not need to take the first order and second order 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 this data and the performance of the trajectory optimization algorithm is compared with the previously learned model. The control horizon is only 2 seconds for this case, however the actuator limits are increased to 2 Newtons 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 for the original model for comparison. The same eight targets in the previous section are used.

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

5.3 Dexterity

Another advantage of a high dimensional robot coupled with a dynamic controller is that it can provide highly dexterous and fast motion which is also inherently safe with only few actuators. Furthermore, the compliance of the body makes them very well suited for tasks like dynamic grasping and placing without the need for highly accurate motion controllers. To partially demonstrate this we devise a scenario where the manipulator aims to reach a target position while avoiding certain obstacles in the 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.

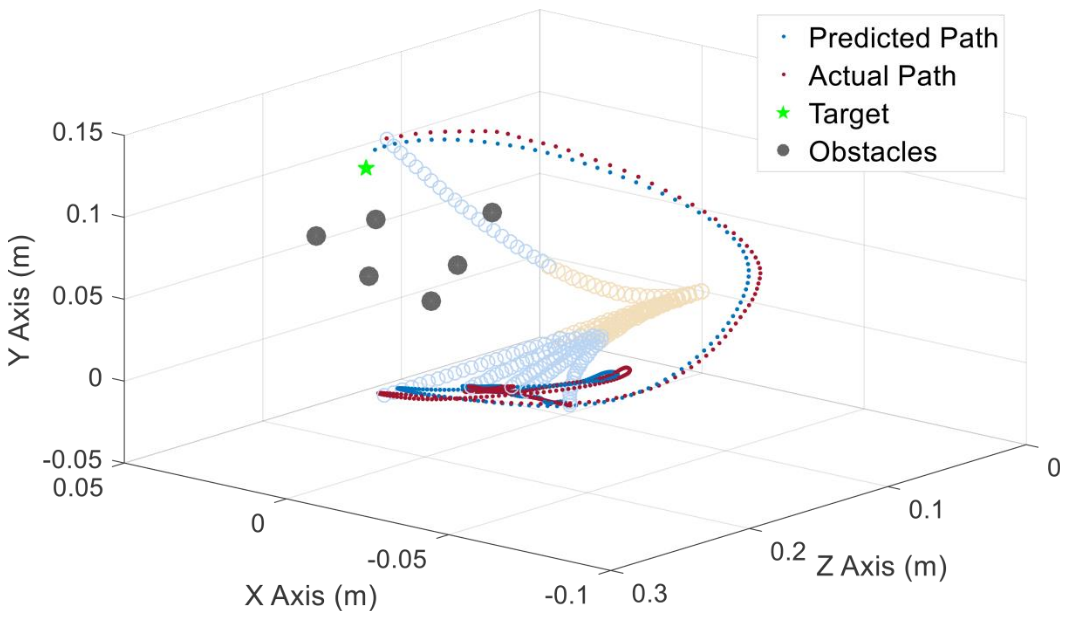


Figure 10. Highly dexterous motion achievable due to the manipulator properties and controller formulation.

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 second 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, same experiments of reaching static target using the end effector is done for 10 randomly selected points. The control horizon is for 2 seconds, the forces are constrained to 2 Newtons and the optimization algorithm is run for 10 iterations. The results are summarized in Table 5. Figure 11 shows one example case of the manipulator performing the reaching task.

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

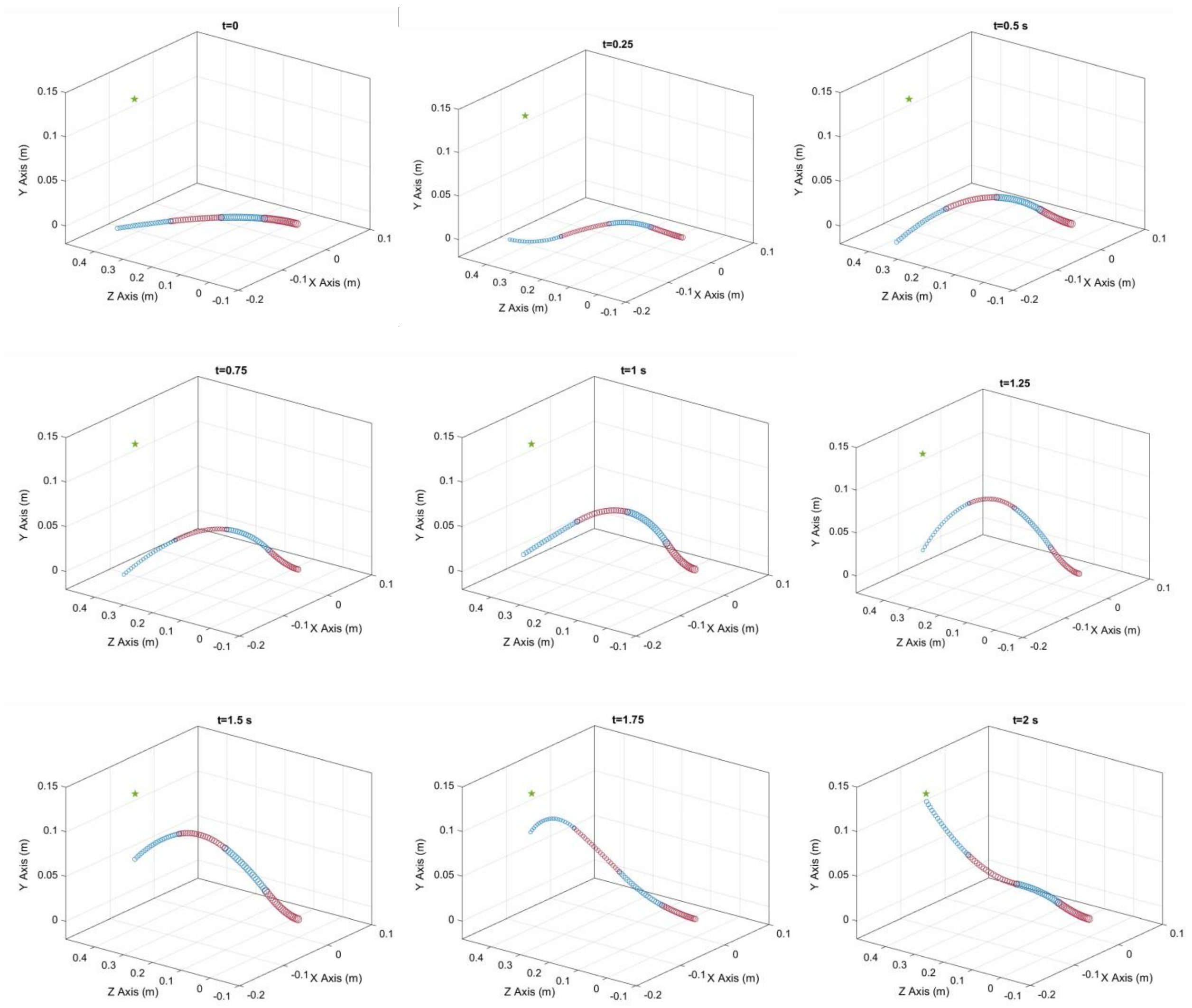


Figure 11. Reaching task executed by the four section manipulator. Each section is differently colored, with the base and only actuated section fixed at [0,0,0]. The target is shown in green.

7. CONCLUSION

This paper presents a novel approach for learning forward dynamic model of soft robotic manipulators. With the help of 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 have risky assumptions about the model. Additionally, the methodology is scalable and general and can be applied to any kind of continuum/soft robotic manipulator. The limiting factor for this approach is the sensor availability and sampling frequency. Systems with highly chaotic behaviour are also undesirable, but they are intractable even with model based approaches.

The use of predictive controllers is more important for soft robots especially due to their longer response delay. Reactive controllers would perform poorly due to this. Also, high accuracy can be achieved even with low control frequencies. However since the controller is purely open-loop, it is not robust to external disturbances. This could be addressed with techniques like model based reinforcement learning. For example in [32], control policies and value functions are approximated using nonparametric regression techniques. Reinforcement learning approaches for optimal controllers using learned dynamic model was done in [33] and [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 modelling errors.

REFERENCES

[1] Renda F, Giorelli M, Calisti M, Cianchetti M, Laschi C. Dynamic Model of a Multibending Soft Robot Arm Driven by Cables. *IEEE Transactions on Robotics* 2014; 30: 1109-1122.

[2] Braganza D, Dawson D, Walker I, Nath N. A Neural Network Controller for Continuum Robots. *IEEE Transactions on Robotics* 2007; 23: 1270-1277.

[3] Xian B, Dawson D, DeQueiroz M, Chen J. A Continuous Asymptotic Tracking Control Strategy for Uncertain Nonlinear Systems. *IEEE Transactions on Automatic Control* 2004; 49: 1206-1206.

[4] Kapadia A, Walker I. Task-space control of extensible continuum manipulators. *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*.

[5] Kapadia A, Walker I, Dawson D, Tatlicioglu E. A model-based sliding mode controller for extensible continuum robots. *Proceedings of the 9th WSEAS Int. Conf. on Signal processing, robotics and automation*. 2010; pp. 113-120.

[6] Kapadia A, Fry K, Walker I. Empirical investigation of closed-loop control of extensible continuum manipulators. *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems* 2014.

[7] Marchese A, Tedrake R, Rus D. Dynamics and trajectory optimization for a soft spatial fluidic elastomer manipulator. *The International Journal of Robotics Research* 2015; 35: 1000-1019.

[8] Falkenhahn V, Hildebrandt A, Neumann R, Sawodny O. Model-based feedforward position control of constant curvature continuum robots using feedback linearization. *2015 IEEE International Conference on Robotics and Automation (ICRA)* 2015.

[9] Falkenhahn V, Hildebrandt A, Neumann R, Sawodny O. Dynamic Control of the Bionic Handling Assistant. *IEEE/ASME Transactions on Mechatronics* 2016: 1-1.

[10] Best C, Gillespie M, Hyatt P, Rupert L, Sherrod V, Killpack M. A New Soft Robot Control Method: Using Model Predictive Control for a Pneumatically Actuated Humanoid. *IEEE Robotics & Automation Magazine 2016;* 23: 75-84.

[11] Nguyen-Tuong, Duy, Matthias Seeger, and Jan Peters. "Model Learning With Local Gaussian Process Regression". Advanced Robotics 23.15 (2009): 2015-2034.

[12] Choi, Younggeun, Shin-Young Cheong, and Nicolas Schweighofer. "Local Online Support Vector Regression For Learning Control". 2007 International Symposium on Computational Intelligence in Robotics and Automation (2007): n. pag. Web. 21 Jan. 2017.

[13] Nikhil Bhushan and Reza Shadmehr. 1998. Evidence for a forward dynamics model in human adaptive motor control. In Proceedings of the 11th International Conference on Neural Information Processing Systems (NIPS'98), M. J. Kearns, S. A. Solla, and D. A. Cohn (Eds.). MIT Press, Cambridge, MA, USA, 3-9.

[14] Wolpert, DM and Ghahramani, Z and Jordan, MI (1995) Forward dynamic models in human motor control: psychophysical evidence. In: Advances in Neural Information Processing Systems 7. Bradford Series . MIT Press, Cambridge, MA, USA, pp. 43-50.

[15] C. G. Atkeson, "Nonparametric model-based reinforcement learning," in Advances in Neural Information Processing Systems, 1998, pp. 1008–1014.

[16] J. Bagnell and J. Schneider, "Autonomous helicopter control using reinforcement learning policy search methods," Robotics and Automation, IEEE International Conference on, vol. 2, pp. 1615—-1620, 2001.

[17] M. P. Deisenroth, P. Englert, J. Peters, and D. Fox, "Multi-task policy search for robotics," in IEEE International Conference on Robotics and Automation. IEEE, 2014, pp. 3876–3881.

[18] Polydoros, Athanasios S. and Lazaros Nalpantidis. "A Reservoir Computing Approach For Learning Forward Dynamics Of Industrial Manipulators". 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2016): n. pag. Web. 25 Jan. 2017.

[19] J. Kocijan, R. Murray-Smith, C. Rasmussen, A. Girard. Gaussian process model based predictive control. Proceeding of the American Control Conference, 2004.

[20] Nguyen-Tuong, Duy, Matthias Seeger, and Jan Peters. "Computed Torque Control With Nonparametric Regression Models". 2008 American Control Conference (2008): n. pag. Web. 30 Jan. 2017.

[21]Sigaud O, Salaün C, Padois V. On-line regression algorithms for learning mechanical models of robots: A survey. *Robotics and Autonomous Systems* 2011; 59: 1115-1129.

[22]Nguyen-Tuong D, Peters J. Model learning for robot control: a survey. *Cognitive Processing* 2011; 12: 319-340.

[23] Renda, Federico et al. "Discrete Cosserat Approach For Soft Robot Dynamics: A New Piece-Wise Constant Strain Model With Torsion And Shears". 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2016): n. pag. Web. 30 Jan. 2017.

[24] Renda, Federico et al. "Dynamic Model Of A Multibending Soft Robot Arm Driven By Cables". IEEE Transactions on Robotics 30.5 (2014): 1109-1122. Web.

[25] F. Boyer, F. Renda "Poincaré's equations for Cosserat media: application to shells" Journal of Nonlinear Science, (2017) 27: 1, 10.1007/s00332-016-9324-7.

[26] C.H. Edwards and D.E. Penney. Differential Equations and Linear Algebra. Always learning. Pearson Education, Limited, 2013.

[27] Thuruthel T, Falotico E, Cianchetti M, Laschi C. Learning Global Inverse Kinematics Solutions for a Continuum Robot. *ROMANSY 21 - Robot Design, Dynamics and Control* 2016: 47-54.

[28] Thuruthel T, Falotico E, Cianchetti M, Renda F, Laschi C. Learning Global Inverse Statics Solution for a Redundant Soft Robot. *Proceedings of the 13th International Conference on Informatics in Control, Automation and Robotics* 2016.

[29] Menezes J, Barreto G. Long-term time series prediction with the NARX network: An empirical evaluation. Neurocomputing 2008; 71: 3335-3343.

[30] Eugen Diaconescu. 2008. The use of NARX neural networks to predict chaotic time series. WSEAS Trans. Comp. Res. 3, 3 (March 2008), 182-191

[31] Fletcher, R. *Practical Methods Of Optimization*. 1st ed. Chichester: Wiley, 1987. Print.

[32]C. G. Atkeson, J. Morimoto. Nonparametric representation of policies and value functions: A trajectory-based approach. Advances in Neural Information Processing Systems, 2002.

[33]P. Abbeel, A. Coates, M. Quigley, A. Y. Ng. An application of reinforcement learning to aerobatic helicopter flight. Advances in Neural Information Processing Systems, 2007.

[34]A. Y. Ng, A. Coates, M. Diel, V. Ganapathi, J. Schulte, B. Tse, E. Berger, E. Liang. Autonomous inverted helicopter flight via reinforcement learning. Proceedings of the 11th International Symposium on Experimental Robotics, 2004.

[35]S. Levine and V. Koltun. Learning complex neural network policies with trajectory optimization. In International Conference on Machine Learning (ICML), 2014.

[36] J.M. Selig. Geometric Fundamentals of Robotics. Monographs in Computer Science. Springer New York, 2007.

[37] F. Renda, F. Boyer, J. Dias and L. Seneviratne. Discrete Cosserat Approach for Multi-Section Soft Robots Dynamics. arXiv:1702.03660 [cs.RO].