Data-efficient Non-parametric Modelling and Control of an Extensible Soft Manipulator

Mohammadreza Kasaei[1], Keyhan Kouhkiloui Babarahmati[1], Zhibin Li[2] and Mohsen Khadem[1]

Abstract— **Data-driven approaches have shown promising results in modeling and controlling robots, specifically soft and flexible robots where developing physics-based models are more challenging. However, these methods often require a large number of real data, and gathering such data is time-consuming and can damage the robot as well. This paper proposed a novel data-efficient and non-parametric approach to develop a continuous model using a small dataset of real robot demonstrations (only 25 points). To the best of our knowledge, the proposed approach is the most sample-efficient method for soft continuum robot. Furthermore, we employed this model to develop a controller to track arbitrary trajectories in the feasible kinematic space. To show the performance of the proposed approach, a set of trajectory-tracking experiments has been conducted. The results showed that the robot was able to track the references precisely even in presence of external loads (up to 25 grams). Moreover, fine object manipulation experiments were performed to demonstrate the effectiveness of the proposed method in real-world tasks. Finally, we compared its performance with common data-driven approaches in seen/useen-before trajectory tracking scenarios. The results validated that the proposed approach significantly outperformed the existing approaches in unseen-before scenarios and offered similar performance in seen-before scenarios.**

I. Introduction

Soft and continuum robots have opened new possibilities for addressing real-world tasks. They can follow smooth curvilinear trajectories, offer great dexterity, and can manipulate objects in constrained environments. They have been proposed for variety of applications including healthcare, agriculture, and search and rescue. However, they exhibit nonlinear behaviour due to their soft structure, distributed friction between the robot components, and unconventional actuation methods [1], [2]. Therefore, accurate modelling of the robots' motion is difficult. Several approaches have been proposed to model and develop controllers for soft robots that can be divided into two major categories: model-based and data-driven.

Developing analytical physics-based models for soft robots that display geometric and behavioural non-linearities is challenging. A systematic review of soft robot modelling approaches is given in [3]. Common modelling methods include polynomial curvature fitting [4], reduced-order finite element models [5], or augmented rigid body models, where the soft robot's motion can be approximated with a rigid link manipulator and then get transformed into a Piecewise Constant Curvature (PCC) model [6]–[8]. Another modelling techniques is to assume the robot is a 1 dimensional continuum structure and employ the Cosserat rod theory to simulate its deformation [9]–[11]. The proposed models are computationally expensive. Some methods overcome this issue by neglecting or simplifying the model dynamics [5], [11]. Additionally, nonlinear behaviour of the robots due to hysteresis or heterogeneity of the materials is often neglected, which reduces the accuracy of the models. Moreover, the proposed models include several parameters representing mechanical characteristics of the robot (e.g., stiffness, shear modulus, and coefficients of friction) that often are not easy to identify independently. Furthermore, there are several promising new designs of soft robots such as parallel flexible robots [12], Electro-Active Polymers (EAP) robots [13], handed shearing auxetics (HSAs) [14], and deployable Kirigami robots [15] that exhibit highly nonlinear behaviour that cannot be simply modelled using conventional methods. Therefore, there is a need for new modelling approaches that accounts for their nonlinear dynamics and material properties.

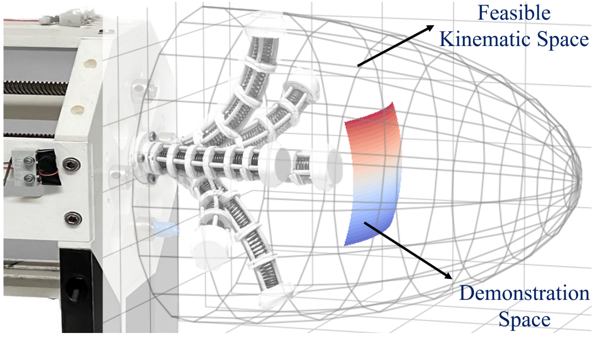


Fig. 1: Feasible kinematic space and demonstration space. The robot is only trained using 25 sparse points in the demonstration space to learn a continuous forward kinematic model and is then tested in the entire feasible kinematic space.

Data-driven and non-parametric modelling techniques can directly address the aforementioned limitations of soft robot models. Commonly, these aim to learn inverse/forward kinematic or dynamics model [16]–[21] or learn a direct control policy for moving the robot using Reinforcement Learning (RL) techniques with/without prior knowledge about geometric models [22]–[24]. The major disadvantages of these methods are the requirement for a large number of training data and the simulation-to-real gap that leads to

[1] Mohammadreza Kasaei, Keyhan Kouhkiloui Babarahmati and Mohsen Khadem are with the School of Informatics, University of Edinburgh, UK. Email: {m.kasaei, kkouhkil, mohsen.khadem}@ed.ac.uk

[2] Zhibin Li is with the Department of Computer Science, University College London, UK. Email: alex.li@ucl.ac.uk

This work is supported by EU H2020 project Enhancing Healthcare with Assistive Robotic Mobile Manipulation (HARMONY, 101017008) and the Medical Research Council [MR/T023252/1]

unsatisfactory results when the model is deployed on the real robot [25]. Additionally, a simulation environment for training these models is not always readily available.

Training data-driven models directly via real robot data has shown to be more effective than a training using simulation. However, it requires a large number of training data points considering numerous robot configurations (for instance, 12000, 7000 and 4096 sample points are used in [26], [27] and [25], respectively). Collecting this data is cumbersome and not always feasible. Furthermore, the data-driven approaches like feed-forward neural networks, RL-based methods, and Gaussian process regression models [28], are unable to generalize their knowledge to unseen environments, specifically for soft robots with complex dynamics.

This paper contributes to developing a novel data efficient and data-driven approach for non-parametric modelling of soft robots, which successfully addresses the aforementioned limitations of the existing data-driven approaches. The algorithm is fully tested on a real and novel extensible soft robot. Our contributions are as follows:

1) We proposed a novel approach to develop continuous forward kinematic models for soft continuum robots by generalizing the concept of Neural ODE [29]: to the best of our knowledge, it is the first time that Neural ODE (a new family of deep neural network models) is employed for modeling soft continuum robots. With this formulations, we can train a continuous FK model using a very few real robot demonstrations (only 25 points) which is a very sample-efficient method for soft continuum robot.
2) We employed the trained model to design and develop a controller: the trained model is used to develop a Jacobian-based controller to steer the robot towards arbitrary trajectories in the feasible kinematic space (unseen before) even in presence of unknown loads. The performance of the model and controller together is compared with the state-of-the-art data driven approaches, including feed-forward neural network (FNN), recurrent neural network (RNN) and, a Reinforcement Learning algorithm (Soft Actor Critic).

The rest of this paper is organized as follows. Section II presents the proposed methodology. In Section III, a set of experiments will be designed and conducted to validate the performance of the proposed approach and the results will be discussed. Afterwards, the performance of the proposed approach will be compared with common existing approaches in Section IV. Finally, The conclusion appears in Section V.

II. Methodology

This section will be started by explaining how the training dataset will be generated using our real robot. Next, we formulate the development of a differential forward kinematic model of the robot as a learning problem using neural ODE [29] and train it using the generated dataset. Finally, the model is used to develop a controller for steering the robot to follow desired trajectories.

A. Generating a training dataset using real robot

The robot used in this paper is an extensible multi-backbone robot [30] and is shown in Fig. 1. The robot is composed of a central flexible backbone made of a compression spring and four flexible rods running in parallel around it. The rods are passed through several spacers and fixed to the tip of the central backbone. By pulling and pushing the rods one can change the shape of the robot. The robot has three inputs. By pulling/pushing all the rods at the same time, it can retract/extend. Additionally, by pulling and pushing two of the rods, the robot can bend in 3D. In this case, the length of the other two rods are constrained and a function of lengths of the pulled/pushed rods. Modelling and control of these robots are challenging [31]. Mainly due to the coupling between the actuation inputs. Additionally, modelling the friction between the rods and spacers is challenging, specially as it increases when the robot bends. Moreover, identifying the model parameters such as mechanical characteristics of the backbones is difficult [32].

Here, we use a limited dataset to model the robot behaviour. To generate the training dataset, an experienced operator performed a set of demonstrations using the real robot. The operator manipulates the lengths of the rods to move the tip in different directions. In these demonstrations, the rods were pulled/pushed up to 3 mm. The feasible kinematic space and demonstration space are depicted in Fig.1. As shown in this figure, the demonstration space is very small in comparison with the feasible kinematic space. The robot inputs, $\boldsymbol{u_t} \in \mathbb{R}^3$, and the corresponding Cartesian coordinates of the robot tip, $\boldsymbol{x_t} \in \mathbb{R}^3$, were recorded at 15 Hz to generate the training dataset, i.e., $\mathcal{D} = \{\boldsymbol{x_t^k}, \boldsymbol{u_t^k}\}_{k=1}^{N}$. In our setup, the position of the robot is estimated using an RGB camera (details discussed in Section .III). It is worth mentioning that generating this dataset was fast and took less than 10 minutes, and our dataset contains N=9100 samples. We only employ 25 points randomly selected from this dataset to model the robot in the next section. The rest of this data is used to train other state-of-the-art machine learning algorithms for comparison.

B. Learning differential kinematics of the robot

We assume that the robot can be modelled using a set of nonlinear differential equations,

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= f(\boldsymbol{x}(t), \boldsymbol{u}(t)), \\ f &: \mathbb{R}^3 \times \mathbb{R}^3 \to \mathbb{R}^3 \end{aligned} \tag{1}$$

with the following initial conditions

$$\begin{aligned} \boldsymbol{x}(t_0) &= \boldsymbol{x}(0), \\ \boldsymbol{u}(t_0) &= \boldsymbol{u}(0). \end{aligned} \tag{2}$$

Here, it is assumed that no analytical expression of the f exists, and we aim to develop a data-efficient neural network to approximate the model. We extend and generalize the idea of Neural ODE (NODE) [29] to learn the differential equations governing the motion of the robot. NODE has been used to approximate a single-input single-output (SISO) system, under the strong assumption that the control signal will remain constant over time [33]. Here, we generalise NODE for multi-input multi-output (MIMO) nonlinear system of equations.

To this end, we discretize the model of the robot given in (1) and reformulate it as a boundary value problem, i.e.,

$$\boldsymbol{x}^+ = f_\theta(\boldsymbol{x}(t), \boldsymbol{u}(t)), \tag{3}$$

with the following boundary conditions

$$\begin{aligned} \boldsymbol{x}_0 &= \boldsymbol{x}(t_0),\ \ \boldsymbol{u}_0 = \boldsymbol{u}(t_k), \\ \boldsymbol{x}_k &= \boldsymbol{x}(t_k),\ \ \boldsymbol{u}_k = \boldsymbol{u}(t_k), \end{aligned} \tag{4}$$

where f_θ represents a neural network approximating f. Assuming f_θ is known, the solution of $\boldsymbol{x}(t_k)$ can be calculated as

$$\boldsymbol{x}(t_k) = \boldsymbol{x}(t_{k-1}) + \int_{t_{k-1}}^{t_k} f_\theta(x(t), u(t))dt, \tag{5}$$

where standard numerical ODE-solvers such as the Runge-Kutta or Adams–Bashforth families of algorithms can approximate $\boldsymbol{x}(t_k)$:

$$\hat{\boldsymbol{x}}(t_k) = \text{ODESolver}(f_\theta, \boldsymbol{x}(t_k), \boldsymbol{x}(t_{k-1})). \tag{6}$$

However, if f_θ is inaccurate or not known, the error in estimating the boundary values can be calculated as

$$\ell_\theta = \|\hat{\boldsymbol{x}}(t_k) - \boldsymbol{x}(t_k)\| \tag{7}$$

Here, we use this error as a loss function to train a neural network that aims to estimate the robot model, i.e, f_θ. To update this model, we randomly selected *only 25 points* from the generated dataset. At each training step, we select one point from the dataset ($[\boldsymbol{x}(t_k), \boldsymbol{u}(t_k)]$). Equation (3-7) are used to estimate the loss function. This error is used in a supervised learning fashion and back propagating through the Neural Network using adjoint sensitivity method [34] for memory efficiency. We note that based on (4) at each training step, the control inputs remain constant $\boldsymbol{u}_0 = \boldsymbol{u}_k$. We have neglected the control inputs dynamics (i.e, $\dot{\boldsymbol{u}} = 0$) and focused on learning the input-output dynamics to improve the learning efficiency. A block diagram of the proposed model is shown in Fig. 2 (green block). In the next subsection, we employ the model for closed-loop control of the robot.

C. Controller Architecture

This subsection describes the design of a robot controller that steers the robot towards predefined paths. Using the trained model in the previous subsection, f_θ, we define a Jacobian matrix, $\mathbf{J}$, that maps the robot end-effector velocity to the configuration space velocities:

$$\dot{\boldsymbol{x}} = \mathbf{J}\dot{\boldsymbol{u}}, \tag{8}$$

where $\boldsymbol{x} \in \mathbb{R}^3$ is the robot's end-effector Cartesian coordinates, $\mathbf{J}$ denotes the 3×3 Jacobian matrix and $\boldsymbol{u} \in \mathbb{R}^3$ represents the input space. Iteratively solving the robot kinematic model can give us a numerical estimate of the Jacobian as:

$$\mathbf{J} = \frac{\triangle \mathbf{x}}{\triangle \boldsymbol{u}} = \begin{bmatrix} \frac{\boldsymbol{x}^T\left(\boldsymbol{u}+\frac{\triangle u_1}{2}\delta_1\right)-\boldsymbol{x}^T\left(\boldsymbol{u}-\frac{\triangle u_1}{2}\delta_1\right)}{\triangle u_1} \\ \cdots \\ \frac{\boldsymbol{x}^T\left(\boldsymbol{u}+\frac{\triangle u_n}{2}\delta_n\right)-\boldsymbol{x}^T\left(\boldsymbol{u}-\frac{\triangle u_n}{2}\delta_n\right)}{\triangle u_n} \end{bmatrix}^T, \tag{9}$$

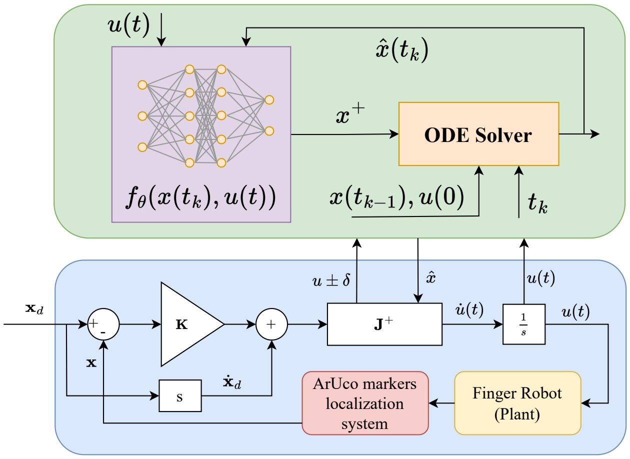


Fig. 2: Block diagram of the proposed controller.

where δ_i denotes the ith unit vector of the canonical basis of the 3-dimensional input space. Now, the general inverse differential kinematic can be defined as follows:

$$\dot{\boldsymbol{u}} = \mathbf{J}^+\dot{\boldsymbol{x}}, \tag{10}$$

where $\mathbf{J}^+$ is the pesudo-inverse of Jacobian. In the remainder of this section, a controller will be designed based on the inverse kinematic of (8), which aims to steer the robot end-effector position, $\boldsymbol{x}(t)$, along the given desired trajectory, $\dot{\boldsymbol{x}}_d(t)$. By applying the following proportional control law to the difference between the desired and actual trajectory, $\tilde{\boldsymbol{x}} = \boldsymbol{x}_d - \boldsymbol{x}$, the error can be reduced to zero:

$$\dot{\mathbf{u}} = \mathbf{J}^+[\dot{\mathbf{x}}_d + \mathbf{K}\tilde{\boldsymbol{x}}], \tag{11}$$

where $\mathbf{K}$ is a symmetric positive definite matrix which represents the proportional gain consists of constant and variable components. The diagonal stiffness profile is defined as follows:

$$\mathbf{K} = \mathbf{K}_{\text{const}} + \mathbf{K}_{\text{var}}(\tilde{\boldsymbol{x}}), \tag{12}$$

where the variable component, $\mathbf{K}_{\text{var}}(\tilde{\boldsymbol{x}})$, is a nonlinear function of error. We define $\mathbf{K}_{\text{var}}(\tilde{\boldsymbol{x}})$ as a diagonal matrix having the following non-zero elements:

$$\mathbf{K}_{\text{var(i,i)}}(\tilde{\boldsymbol{x}}) = e^{(\beta_i \tilde{\boldsymbol{x}}_i)^2},\ \forall i \in [1, 3] \subset \mathbb{N}, \tag{13}$$

where β_i is the constant tuning parameter. It shall also be noted that the stiffness profile in (13) is just one of the possible stiffness profiles that can be implemented in this framework; which does not impose any specific formulation. Fig. 2 shows the overall architecture of the proposed controller. To assess the performance of the proposed system, a set of experiments will be conducted in the following section.

III. Experiments

Here, a set of experiments were to evaluate the performance of the proposed method in different scenarios.

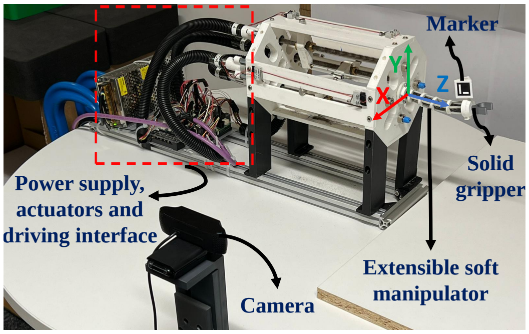

Marker
Y
Z
X
Solid gripper
Power supply, actuators and driving interface
Camera
Extensible soft manipulator

Fig. 3: Extensible soft manipulator setup.

A. Experiment Setup

Fig. 3 shows our experimental setup which consists of an extensible soft manipulator, a Logitech RGB camera mounted on the robot working area, and a user interface to record, start and stop the experiments. As shown in Fig. 3, the robot is composed of a flexible backbone rigidly attached to the spacers along with four rods fixed at the end spacer and passing through the rest of the spacers with enough clearance forms the main body of the robot. Four brushless DC motors (Maxon Motors) alongside quadratic encoders and 150 : 1 reduction gearheads are used to power the robot. Four PID position controller modules (EPOS4 Compact 50/5 CAN) are employed to precisely control the position of the motor based on the encoders feedback. The controller modules communicate with a PC through CAN protocol to set/get the controllers set-points and their configurations. Lead screws carrying 3D printed connectors connected to braided tubes, are attached to the motors to convert the power produced by the motor to pull and push the tubes.

In order to detect the position of the robot's tip with the camera, an ArUco marker [35], [36] attaches to the robot's tip to provide feedback to the control loop depicted in Fig. 2. The hyperparameters and network structure are summarized in Table I.

TABLE I: Hyperparameters and network structure.

Hyperparameter	value
No. of hidden neuron (θ)	112 (64,32,16)
No. of hidden layers	3
Activation functions	ELU
Learning rate	0.001
Type of ode-solver	fixed-adams
Absolute tolerance for ode-solver	1e-9
Relative tolerance for ode-solver	1e-7
Number of iteration	7000

Through all the experiments, the initial length of the rods are set to 0.1 m for and $\boldsymbol{u} = [0\ 0\ 0]^T$ m. The constant proportional gain of the controller $\mathbf{K}_{\mathrm{Const}}$ is set to be $0.45\,\mathrm{Nm}^{-2}$ and $\beta_i = 150$ as through a set of trials, this value achieves the minimum target tracking error.

B. Experiment Design

The following experiments are carried out to evaluate the proposed model and control strategy:

1) **Static Target Tracking:** 9 various static targets are predefined across the workspace of the robot and the robot is expected to reach them with the minimum amount of position error.
2) **Trajectory Tracking:** The robot is set to follow several trajectories in 3D space containing: i) a circle on X-Y plane with the radius of 0.03 m, ii) a square with the side size of 0.06 m on X-Y plane, iii) an eight-figure ($x = a\cos(\frac{2t\pi}{T})$ and $y = \frac{b}{2}\sin(\frac{4t\pi}{T})$, where $a = 0.03$, $b = 0.05$, $T = 100\,\mathrm{s}$, and $t = 0:120\,\mathrm{s}$), iv) an equilateral triangle with the length of 0.06 m for each side and v) a helix with the radius of 0.03 m and pitch of 0.02 m along Z-axis. Three trials are performed for each trajectory to evaluate the repeatability and accuracy of the tracking.
3) **Load Carrying in Trajectory Tracking Mode:** A set of small weights, ranging from 5 to 25 g are added to the tip of the robot to execute the eight-figure trajectory tracking here with the same parameters to evaluate the functionality of the robot in the presence of added loads.
4) **Pick and Place Tasks of Small Tubes**: A set of experiments are carried out to present a probable application use of the robot in sorting small tubes by picking them from pre-defined locations and placing them in the arbitrary ones.

C. Results and Discussions

In static target tracking scenario, the followings are measured and used to evaluate the robot performance: the steady state error (SSE), standard deviation (σ) and settling time (ST). Results are presented in Table II. The results indicate that the position error is less than 1 mm. The low standard deviation demonstrated that the error is stable across all of the experiments.

TABLE II: Static Target Mode

	SSE (mm)	σ (mm)	ST (sec)
$\tilde{x}$	0.87	0.72	6.48
$\tilde{y}$	0.67	0.53	4.94
$\tilde{z}$	0.43	0.36	3.73
$\tilde{\mathbf{X}}$	**0.65**	**0.57**	**5.05**

In trajectory tracking mode, the root mean squared error (RMSE) and standard deviation (σ) of error across five trials are calculated and presented in Table III. Presented results compare the robot tip trajectory with the desired trajectory are shown in Fig. 4. The robot is capable of tracking the circle, square, eight-figure and an equilateral triangle respectively with maximum RMSE of 3.5×10^{-3} m.

TABLE III: Trajectory Tracking Mode

	RMSE			σ		
	$\tilde{x}$ (mm)	$\tilde{y}$ (mm)	$\tilde{z}$ (mm)	$\tilde{x}$ (mm)	$\tilde{y}$ (mm)	$\tilde{z}$ (mm)
Circle	3.10	3.21	3.32	3.07	3.16	3.28
Square	3.22	3.25	3.31	3.13	3.17	3.29
Eight	3.39	3.27	3.12	3.31	3.19	3.07
Triangle	3.48	3.43	3.17	3.23	3.19	3.10
Helix	3.46	3.37	3.23	3.41	3.27	3.15

Fig. 5 presents the results of the eight-figure trajectory tracking experiment when various weights are added at the

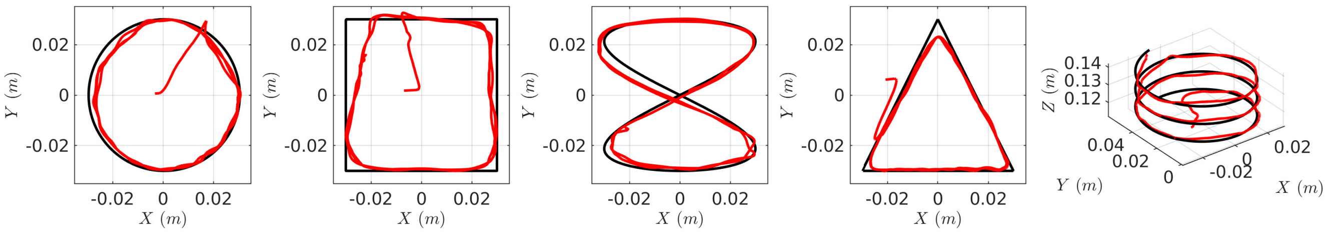
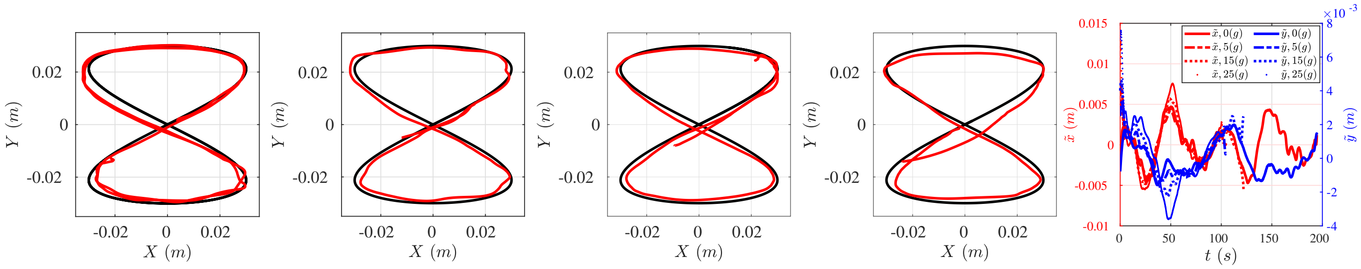


Fig. 4: Trajectory tracking scenarios; the solid black lines represent desired trajectory and the solid red lines represent current end-effector position.



(a) (b) (c) (d) (e)

Fig. 5: Trajectory tracking with added weights to the tip of the robot - Eight-figure : (a) 0 g, (b) 5 g, (c) 15 g and (d) 25 g (solid black line: desired trajectory, solid red line: current end-effector position), (e) Comparison of $\tilde{x}$ and $\tilde{y}$ in the presence of various added weights.

tip of the robot. Robot trajectory is compared with the desired trajectory with no weight and with 5 g, 15 g, and 25 g weights. Trajectory tracking error ($\tilde{x}$ and $\tilde{y}$) is shown as well. The figure also shows that the tracking error increases as more weights get added to the tip of the robot. Additionally, the results highlight the variable stiffness of the robot when the position error increases to make the robot more stiff to compensate the added weights to keep the position error at its minimum value. The robot can safely carry loads up to 25 g, which is sufficient for carrying small tools and grippers.

We performed one task, consists of two various scenarios, to demonstrate the application use of the robot in fine object manipulation in real-world environments such as pharmaceutical laboratories. Therefore, the robot is using a rigid gripper to sort small test tubes containing chemicals and biosamples. The robot is used in uni-lateral tele-operation mode. The designed controller is used to follow targets defined by an operator using a keyboard. In the first scenario (as can be seen in Fig. 6 (top-row)), the user's task is about sorting different tubes with different colours in two different racks vertically (along z-axis). The tubes with the same colour are supposed to be placed in one rack and the other ones are supposed to be sorted in the other rack. As can be seen in Fig. 6 (bottom row), the second scenario, is about sorting the tubes with the same colour, and picking and placing them (one after another) in the empty rack horizontally (along x-axis). Results are presented in the supplementary multimedia file and summarised in Fig. 6. The robot is placed nearby the test tubes and it is used to successfully pick and place them. In practice the proposed robot may be paired with a rigid arm for accurate positioning near the objects. This way we can combine the structural stability of the arm with the flexibility of the soft manipulator (finger) to perform complex manipulation tasks.

IV. Comparison Study

In this section, we will compare the performance of the proposed method against three common existing approaches: feed-forward neural network (FNN), recurrent neural network (RNN) and Reinforcement Learning (RL) in two scenarios: seen-before and unseen-before trajectories tracking. The former scenario is focused on evaluating the performance of the methods while performing tasks in the demonstration space but the latter scenario is focused on evaluating them in generalizing their knowledge while performing tasks in the entire feasible kinematic space. In the seen-before scenario, the robot is asked to track a square trajectory with the side size of 0.0025 m on X-Y plane while in the unseen-before scenario, the robot should track a square helix with the side size of 0.04 m and pitch of 0.01 m.

For FNN, a multilayer perceptrons (MLP) ($u_t \rightarrow x_t$) and for RNN, a nonlinear auto-regressive network with exogenous inputs (NARX) have been implemented ($(x_t, u_t) \rightarrow x_{t+1}$), both have been trained using the generated dataset, $\mathcal{D}$, described in Section II and mean square error have been used as their loss functions. For RL, we used Soft Actor Critic (SAC) [37] via stable baseline3 [38] to train the network. In our setup, at each step, the agent received the cable lengths as the observation and should predict the position of the tip as the action ($u_t \rightarrow x_t$) which is restricted to the demonstration space presented in Fig. 1. Euclidean distance between the predicted position and ground truth is used as the reward function and an episode will be terminated after each prediction.

To have a fair comparison, the network size for all methods are identical. Thus, we trained three FK models

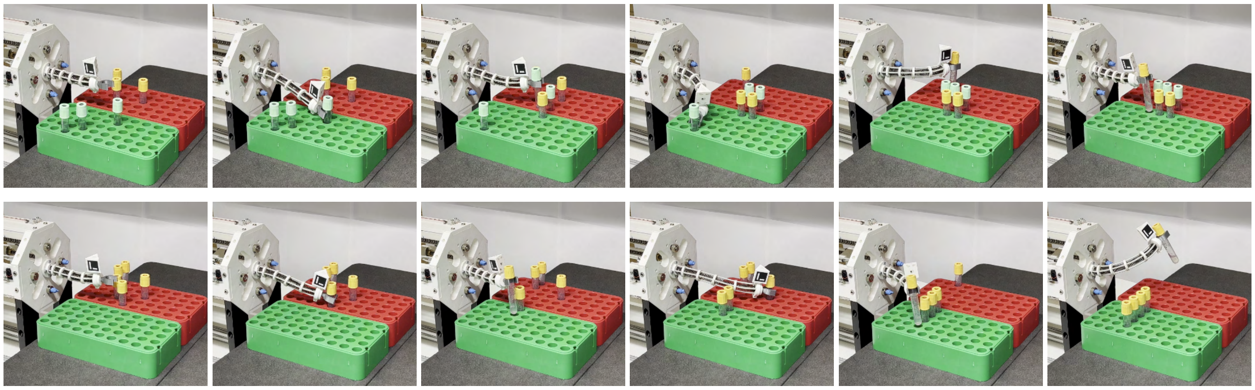


Fig. 6: A real-world application: sorting small-size medical test tubes in the racks. In these experiments, the robot is guided by the targets sent by an operator to sort the test tubes, through picking and placing them in the goal locations.

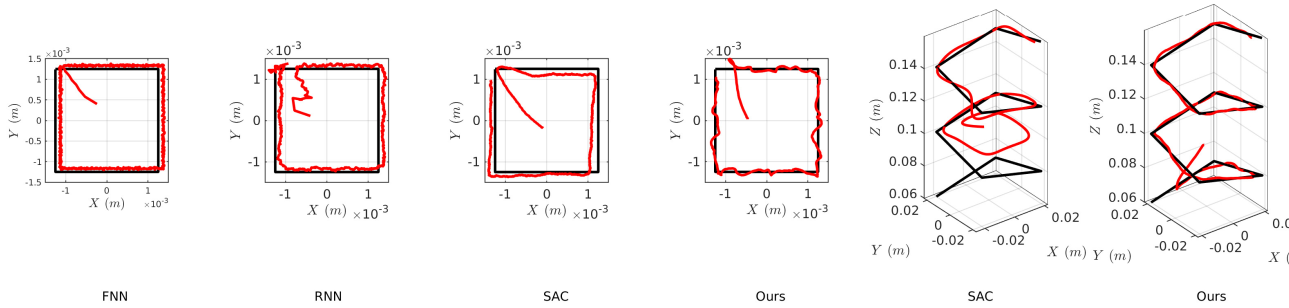


Fig. 7: Comparison results of seen-before and unseen-before scenarios: four 2D plots represent the result of seen-before scenario and two 3D plots represent the result of unseen-before scenario.

and employed them in the same control loop depicted in Fig. 2. To compare their performances, the root mean squared error (RMSE) and standard deviation (σ) of error across five trials are calculated and summarized in Table IV. Representative results are depicted in Fig. 7. As shown in this figure, all the approaches were able to accomplish the seen-before scenario successfully and FNN was the best one. According to the results of this scenario, although the proposed method does not outperform the other approaches, its performance is very well and very competitive among them, despite it has been trained only on 25 points while the other approaches have been trained on the entire dataset. In the unseen-before scenario, FNN and RNN were not able to generalize their knowledge to unseen-before states resulting the robot becoming uncontrollable while SAC and the proposed method successfully accomplished the task. As the results showed, the proposed method significantly outperformed all the other approaches in unseen-before scenario and was able to generalize its knowledge to the entire kinematic space.

TABLE IV: Comparison results in the seen-before and unseen-before trajectory tracking scenarios.

		RMSE			σ		
		$\tilde{x}$ (mm)	$\tilde{y}$ (mm)	$\tilde{z}$ (mm)	$\tilde{x}$ (mm)	$\tilde{y}$ (mm)	$\tilde{z}$ (mm)
seen	**FNN**	**0.142**	**0.099**	**0.005**	**0.057**	**0.064**	**0.017**
	RNN	0.145	0.049	4.985	0.072	0.047	0.018
	SAC	0.362	0.108	4.985	0.076	0.053	0.018
	Ours	0.381	0.196	0.009	0.083	0.032	0.009
unseen	FNN	-	-	-	-	-	-
	RNN	-	-	-	-	-	-
	SAC	6.315	6.988	16.190	6.262	6.988	13.824
	Ours	**6.343**	**6.550**	**5.150**	**6.316**	**6.531**	**4.999**

V. Conclusion

In this paper, we proposed a novel data-efficient and non-parametric approach for modelling and control of extensible soft manipulators. In this approach, a neural network was employed to approximate the robot's model, which can be represented approximately by a set of nonlinear differential equations. We adopt the Neural-ODE to train a continuous model using a very few limited real robot demonstrations (only 25 points). To the best of our knowledge, it is the most sample efficient method in the literature. Using the trained model, we developed a controller for the robot to track arbitrary trajectories within the whole kinematic space. Furthermore, through experimental validations, we showed that various arbitrary 3D points can be reached or complex trajectories can be followed repeatedly. Finally, we compared the performance of the proposed approach with a set of existing data-driven approaches in two scenarios: seen-before and unseen-before trajectories tracking. In unseen-before scenarios, the proposed approach far outperformed the existing approaches, whereas in seen-before scenarios, its performance equalled other methods. As future work, we plan to improve the model and control by including the robot dynamics and external contact forces.

References

[1] N. El-Atab, R. B. Mishra, F. Al-Modaf, L. Joharji, A. A. Alsharif, H. Alamoudi, M. Diaz, N. Qaiser, and M. M. Hussain, "Soft actuators for soft robotic applications: A review," *Advanced Intelligent Systems*, vol. 2, no. 10, p. 2000128, 2020.

[2] C. Lee, M. Kim, Y. J. Kim, N. Hong, S. Ryu, H. J. Kim, and S. Kim, "Soft robot review," *International Journal of Control, Automation and Systems*, vol. 15, no. 1, pp. 3–15, Feb 2017.

[3] C. Armanini, C. Messer, A. T. Mathew, F. Boyer, C. Duriez, and F. Renda, "Soft robots modeling: a literature unwinding," 2021.

[4] C. Della Santina and D. Rus, "Control oriented modeling of soft robots: the polynomial curvature case," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 290–298, 2019.

[5] R. K. Katzschmann, M. Thieffry, O. Goury, A. Kruszewski, T.-M. Guerra, C. Duriez, and D. Rus, "Dynamically closed-loop controlled soft robotic arm using a reduced order finite element model with state observer," in *2019 2nd IEEE international conference on soft robotics (RoboSoft)*. IEEE, 2019, pp. 717–724.

[6] R. K. Katzschmann, C. Della Santina, Y. Toshimitsu, A. Bicchi, and D. Rus, "Dynamic motion control of multi-segment soft robots using piecewise constant curvature matched with an augmented rigid body model," in *2019 2nd IEEE International Conference on Soft Robotics (RoboSoft)*. IEEE, 2019, pp. 454–461.

[7] C. Della Santina, R. K. Katzschmann, A. Biechi, and D. Rus, "Dynamic control of soft robots interacting with the environment," in *2018 IEEE International Conference on Soft Robotics (RoboSoft)*. IEEE, 2018, pp. 46–53.

[8] C. Della Santina, R. K. Katzschmann, A. Bicchi, and D. Rus, "Model-based dynamic feedback control of a planar soft robot: trajectory tracking and interaction with the environment," *The International Journal of Robotics Research*, vol. 39, no. 4, pp. 490–513, 2020.

[9] J. Till, V. Aloi, and C. Rucker, "Real-time dynamics of soft and continuum robots based on cosserat rod models," *The International Journal of Robotics Research*, vol. 38, no. 6, pp. 723–746, 2019.

[10] D. C. Rucker, B. A. Jones, and R. J. Webster III, "A geometrically exact model for externally loaded concentric-tube continuum robots," *IEEE Transactions on Robotics*, vol. 26, no. 5, pp. 769–780, 2010.

[11] B. Thamo, K. Dhaliwal, and M. Khadem, "Rapid solution of cosserat rod equations via a nonlinear partial observer," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 9433–9438.

[12] J. B. Hopkins, J. Rivera, C. Kim, and G. Krishnan, "Synthesis and Analysis of Soft Parallel Robots Comprised of Active Constraints," *Journal of Mechanisms and Robotics*, vol. 7, no. 1, 02 2015.

[13] J. D. Carrico, K. J. Kim, and K. K. Leang, "3d-printed ionic polymer-metal composite soft crawling robot," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 4313–4320.

[14] J. I. Lipton, R. MacCurdy, Z. Manchester, L. Chin, D. Cellucci, and D. Rus, "Handedness in shearing auxetics creates rigid and compliant structures," *Science*, vol. 360, no. 6389, pp. 632–635, 2018.

[15] A. Sedal, A. H. Memar, T. Liu, Y. Mengüç, and N. Corson, "Design of deployable soft robots through plastic deformation of kirigami structures," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2272–2279, 2020.

[16] B. Thamo, D. Hanley, K. Dhaliwal, and M. Khadem, "Data-driven steering of concentric tube robots in unknown environments via dynamic mode decomposition," *IEEE Robotics and Automation Letters*, vol. 8, no. 2, pp. 856–863, 2023.

[17] B. Thamo, F. Alambeigi, K. Dhaliwal, and M. Khadem, "A hybrid dual jacobian approach for autonomous control of concentric tube robots in unknown constrained environments," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 2809–2815.

[18] T. Thuruthel, E. Falotico, M. Cianchetti, F. Renda, and C. Laschi, "Learning global inverse statics solution for a redundant soft robot," in *Proceedings of the 13th International Conference on Informatics in Control, Automation and Robotics*, vol. 2, 2016, pp. 303–310.

[19] J. M. Bern, Y. Schnider, P. Banzet, N. Kumar, and S. Coros, "Soft robot control with a learned differentiable model," in *2020 3rd IEEE International Conference on Soft Robotics (RoboSoft)*. IEEE, 2020, pp. 417–423.

[20] T. G. Thuruthel, E. Falotico, F. Renda, and C. Laschi, "Learning dynamic models for open loop predictive control of soft robotic manipulators," *Bioinspiration & biomimetics*, vol. 12, no. 6, p. 066003, 2017.

[21] D. Bruder, C. D. Remy, and R. Vasudevan, "Nonlinear system identification of soft robot dynamics using koopman operator theory," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 6244–6250.

[22] S. Bhagat, H. Banerjee, Z. T. Ho Tse, and H. Ren, "Deep reinforcement learning for soft, flexible robots: Brief review with impending challenges," *Robotics*, vol. 8, no. 1, p. 4, 2019.

[23] R. Morimoto, S. Nishikawa, R. Niiyama, and Y. Kuniyoshi, "Model-free reinforcement learning with ensemble for a soft continuum robot arm," in *2021 IEEE 4th International Conference on Soft Robotics (RoboSoft)*. IEEE, 2021, pp. 141–148.

[24] D. Büchler, S. Guist, R. Calandra, V. Berenz, B. Schölkopf, and J. Peters, "Learning to play table tennis from scratch using muscular robots," *IEEE Transactions on Robotics*, 2022.

[25] G. Fang, Y. Tian, Z.-X. Yang, J. M. Geraedts, and C. C. Wang, "Efficient jacobian-based inverse kinematics with sim-to-real transfer of soft robots by learning," *IEEE/ASME Transactions on Mechatronics*, 2022.

[26] T. G. Thuruthel, E. Falotico, F. Renda, and C. Laschi, "Model-based reinforcement learning for closed-loop dynamic control of soft robotic manipulators," *IEEE Transactions on Robotics*, vol. 35, no. 1, pp. 124–134, 2018.

[27] T. George Thuruthel, F. Renda, and F. Iida, "First-order dynamic modeling and control of soft robots," *Frontiers in Robotics and AI*, vol. 7, p. 95, 2020.

[28] W. Xu, J. Chen, H. Y. Lau, and H. Ren, "Data-driven methods towards learning the highly nonlinear inverse kinematics of tendon-driven surgical manipulators," *The International Journal of Medical Robotics and Computer Assisted Surgery*, vol. 13, no. 3, p. e1774, 2017.

[29] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," *Advances in neural information processing systems*, vol. 31, 2018.

[30] A. Bajo and N. Simaan, "Finding lost wrenches: Using continuum robots for contact detection and estimation of contact location," in *2010 IEEE International Conference on Robotics and Automation*, 2010, pp. 3666–3673.

[31] L. Wang and N. Simaan, *Investigation of Error Propagation in Multi-backbone Continuum Robots*. Springer International Publishing, 2014, pp. 385–394.

[32] ——, "Geometric calibration of continuum robots: Joint space and equilibrium shape deviations," *IEEE Transactions on Robotics*, vol. 35, no. 2, pp. 387–402, 2019.

[33] Y. D. Zhong, B. Dey, and A. Chakraborty, "Symplectic ode-net: Learning hamiltonian dynamics with control," *arXiv preprint arXiv:1909.12077*, 2019.

[34] L. S. Pontryagin, *Mathematical theory of optimal processes*. CRC press, 1987.

[35] S. Garrido-Jurado, R. M. noz Salinas, F. Madrid-Cuevas, and M. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280 – 2292, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0031320314000235

[36] S. Garrido-Jurado, R. M. noz Salinas, F. Madrid-Cuevas, and R. Medina-Carnicer, "Generation of fiducial marker dictionaries using mixed integer linear programming," *Pattern Recognition*, vol. 51, pp. 481 – 491, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0031320315003544

[37] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*. PMLR, 2018, pp. 1861–1870.

[38] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021. [Online]. Available: http://jmlr.org/papers/v22/20-1364.html