Localised Online Learning-Based Control of a Soft Redundant Manipulator Under Variable Loading

Justin D.L. Ho[1], Kit-Hang Lee[1], Wai Lun Tang[1], Ka-Ming Hui[1], Kaspar Althoefer[2], James Lam[1] and Ka-Wai Kwok*[1]

[1]*Department of Mechanical Engineering, The University of Hong Kong, Hong Kong, Hong Kong*

[2]*Centre for Advanced Robotics @ Queen Mary, School of Engineering and Materials Science, Queen Mary University of London, London, United Kingdom*

* (corresponding author's phone: +852-3917-2636; e-mail: kwokkw@hku.hk).

Localised Online Learning-Based Control of a Soft Redundant Manipulator Under Variable Loading

Soft robots are inherently compliant and manoeuvrable manipulators that can passively adapt to their environment. However, in order to fully make use of their unique properties, accurate control should still be maintained when affected by external loading. Commonly used model-based approaches often have low tolerance to unmodelled loading, resulting in significant error when acted on by them. Therefore, in this study we employ a nonparametric learning-based method that can approximate and update the inverse model of a redundant two-segment soft robot in an online manner. The primary contribution of this work is the application and evaluation of the proposed framework on a redundant soft robot. With the addition of redundancy, a constrained optimisation approach is taken to consistently resolve null-space behaviour. Through this control framework, the controller can continuously adapt to unknown external disturbances during runtime and maintain end-effector accuracy. The performance of the control framework was evaluated by tracking of a 3D trajectory with a static tip load, and a variable weight tip load. The results indicate that the proposed controller could effectively adapt to the disturbances and continue to track the desired trajectory accurately.

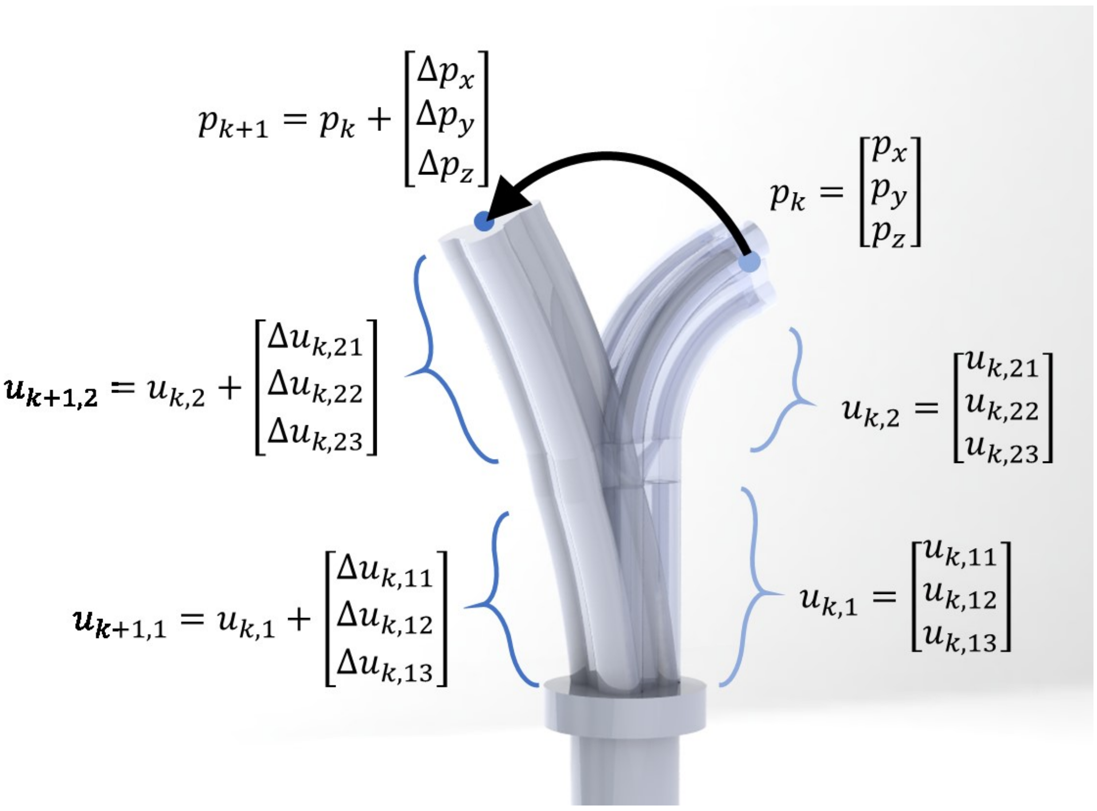
Keywords: soft robotics; modelling; compliance; machine learning;

1. **Introduction**

The introduction of robots constructed from hyper-elastic materials and embedded with fluidically driven chambers have given rise to a new class of robots [1, 2]. These soft robots are inherently compliant, manoeuvrable and are able to passively adapt to dynamic and unstructured environments. As a result, their prevalence in specialised applications like surgical intervention [3-6] has grown, and has drawn interest from other fields such as underwater manipulation [7] and search and rescue [8, 9].

Subsequently, the growth of soft robotics field has sparked research focused on modelling the behaviour of soft continuum robots [9-13]. Analytical models capable of offering the forward mapping from robot actuation to its task space generally provide the basis for accurate and dexterous control of conventional, rigid robots. However, the analogue for modelling soft continuum robots can be prohibitively complex due to the non-linear elasticity, compliance and fluidic actuation of the robot body.

Approximations like the piecewise constant curvature (PCC) approach is commonly used to approximate the kinematic mapping of soft robots [9, 10, 14, 15]. The PCC assumption provides a simplified representation of serial-link continuum robots by assuming their segments are smoothly connected with circular bending profiles. Although the use of PCC still remains predominant due to its obtainable and closed-form solutions [16, 17], any loading to the robot that results in non-circular bending invalidates the PCC assumption, resulting in significant inaccuracies.



$p_{k+1} = p_k + \begin{bmatrix} \Delta p_x \\ \Delta p_y \\ \Delta p_z \end{bmatrix}$
$p_k = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}$
$\boldsymbol{u_{k+1,2}} = u_{k,2} + \begin{bmatrix} \Delta u_{k,21} \\ \Delta u_{k,22} \\ \Delta u_{k,23} \end{bmatrix}$
$u_{k,2} = \begin{bmatrix} u_{k,21} \\ u_{k,22} \\ u_{k,23} \end{bmatrix}$
$\boldsymbol{u_{k+1,1}} = u_{k,1} + \begin{bmatrix} \Delta u_{k,11} \\ \Delta u_{k,12} \\ \Delta u_{k,13} \end{bmatrix}$
$u_{k,1} = \begin{bmatrix} u_{k,11} \\ u_{k,12} \\ u_{k,13} \end{bmatrix}$

Figure 1. Labeled illustration of the robot motion transition for a two-segment soft robot. Pictured is the transition from state $(p_k, u_k) \rightarrow (p_{k+1}, u_{k+1})$. Inflation of the fluidic chambers of the first and second segments are labeled $u_{k,1}$ and $u_{k,2}$, respectively.

In contrast, other modelling techniques, including those based on the Cosserat rod theory, can take into account external loading such as gravity [18] as well as buoyancy and drag loading due to movement in water [19]. However, these approaches are still too computationally complex to apply in real-time robot control. Fully utilising the conformability and manoeuvrability of soft continuum robots while also maintaining end-effector accuracy is still technically challenging.

A stepping stone towards this goal is through learning-based approaches [11], which have gained popularity in soft robotics because of their ability to bypass the difficulties in modelling uncertain internal and external dynamics. Model-free learning approaches that are adaptive to dynamics present in the robot itself can avoid determining the material and geometrical specifications of the controlled soft robot, as they can be made implicit in the obtained data they learn from. Not only does this allow greater freedom in modifying the robot's structure, but it also provides leeway during construction of the robot, as material inconsistencies within segments and joints can affect bending symmetry and further introduce unmodelled dynamics.

Previously, machine learning-based methods have been used with conventional rigid-link robots, and have been able to approximate their inverse models [20-22], producing results comparable to analytical model-based solutions [23, 24]. Relating to soft continuum robots, a neural network (NN) was applied in [25] to control a 1-DOF planar soft robotic manipulator, with the study outlining the adaptability of machine-learning approaches to mechanical discrepancies that analytical approaches do not possess. The study provided a useful preliminary application of machine-learning for soft robots, however the adaptability of the algorithm to external loading was not addressed. Another NN-based control was applied to the Festo Robotino® XT hyper-redundant robot [26], with kinematic redundancies taken into account by considering both of the robot's section trajectories. In that study, two NNs were used in tandem to adapt to hysteresis and other modelling uncertainties. However, the kinematic control was computationally intensive, making useful real-time control unfeasible at that stage. A number of other NN-based approaches have been used to learn the inverse kinematics of soft continuum robots [27-29], however the presence of external disturbances were not accounted for in these studies.

Recently, Lee *et al.* [30] proposed a generic control framework based on [23, 31] that is able to directly learn the inverse model of a soft continuum robot for task-space control in an online manner, without knowing the robot's geometric parameters. The study applied locally-weighted models in order to estimate the inverse kinematics and control the tip orientation (pitch and yaw) of a single-segment soft manipulator, and was able to maintain tip orientation accuracy even in the presence of unknown loading on the robot body. In contrast to NN methods, where the model structure is typically determined before the training process, nonparametric regression allows for the optimal model structure to be determined from the training data. Furthermore, the use of such an online learning algorithm allows for rapid updating of the individual local inverse models, which in turn enables adaptation to any change in external loading.

In this paper, we extend the control framework in [30] to a multi-segment soft continuum robot, also addressing the over-actuated and redundant nature of multiple bending segments. In contrast to the previous study where only the orientation of a single segment actuator was controlled, this study applies 3D positional control to a two-segment actuator. This extension presents the opportunity for more dexterous soft robot tasks, e.g. intraluminal endoscopy, where views behind or around soft-tissue bodies may be otherwise impaired with single segment robots. Workspace exploration is required to train the learning algorithm, with the collection of generated offline pre-training data necessary to learn the proposed controller. Validation of the learned controller is performed through 3D positional trajectory tracking of the soft robot. The primary contributions of this work are as follows:

- Design and implementation of a general learning-based framework, which enables robust control of a multi-segment soft continuum robot by adapting to unmodelled loading via online learning.
- Consistent resolution of null-space behaviour, which can resist variable distributions of sampled learning data.
- Experimental validation of the proposed controller, which evaluates how a six-chamber continuum robot performs a task of 3-D trajectory following with the addition of an unmodeled, variable weight tip load.

2. Methods

2.1. Design of two-segment soft manipulator

The soft robot used in this study is constructed from moulded RTV (Room Temperature Vulcanization) silicone rubber (Dragon Skin 10, Smooth-on Inc.) segments and 3D printed joining components. Each segment comprises of three cylindrical fluidic chambers spaced 120° apart around the section perimeter, with a total of six chambers. Each chamber is constrained radially by a helically wound fibre in order to limit the chamber expansion to only the axial direction. This facilitates omnidirectional bending of the robot segments when different inflation pressures are provided to each chamber. Each segment can achieve a maximum bending angle of approximately 100° in any direction. Two segments were connected in series by a 3D printed (Stereolithographic) coupling, which enables each air tube of the distal (relative to the robot base) segment to feed to the centre cavity of the proximal segment. By using a two-segment soft robot, we have a system with greater dexterity which allows for improved 3D positional control of the tip and presents an opportunity to employ the null-space behaviour to meet a secondary goal. A 3D printed tip is attached to the top of the distal segment to allow mounting of the positional tracking sensor. The bottom of the robot is fixed to a 3D printed base that remains stationary. The outer diameter of the robot is 27 mm and has a total length of 155 mm. An overview of the soft robot is shown in **Figure 2(d)**.

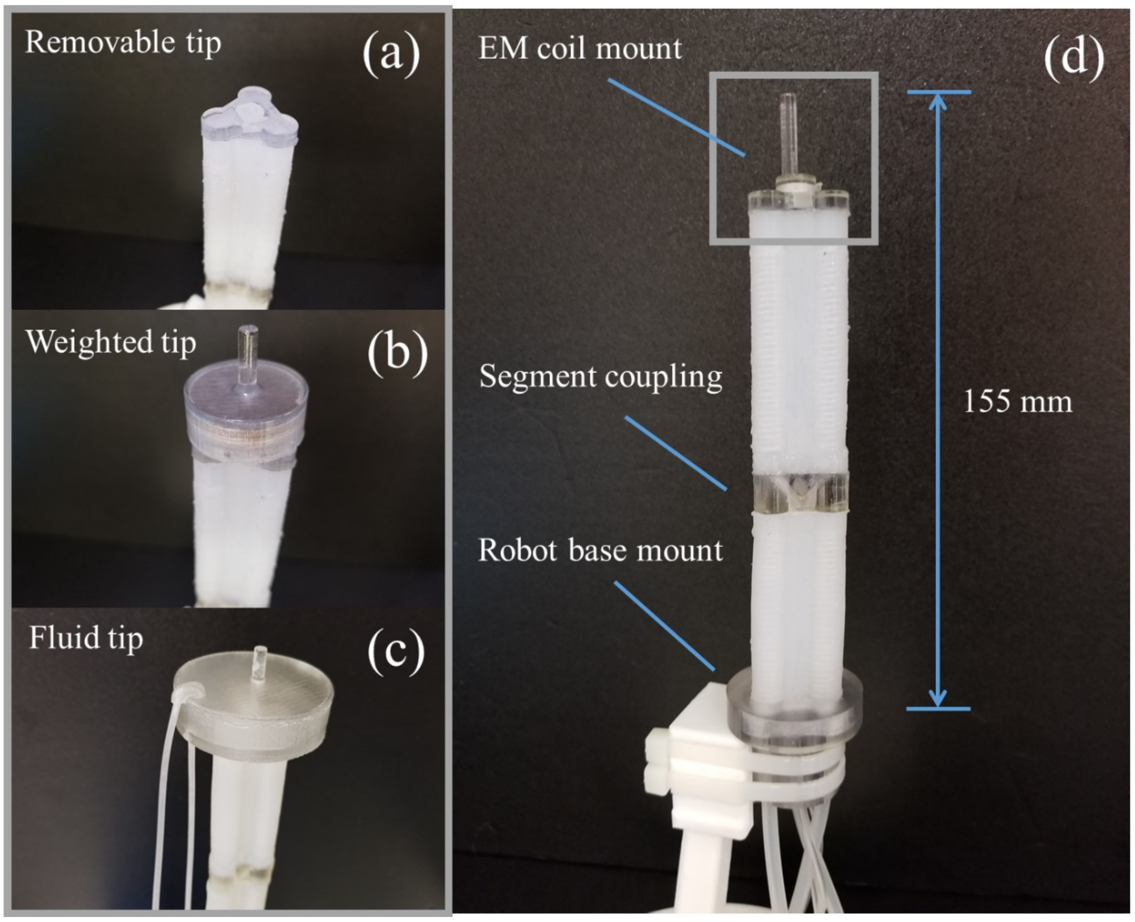


Figure 2. (a) Plug-in tip allows easy switching to other tip types for experimental validation: **(b)** 14.2 g weighted tip, **(c)** fluid tip weighing 14 g when empty, with maximum weight of 32 g when full of water. **(d)** Soft robotic continuum robot with SLA 3D printed coupling and mounts.

2.2. Robot parameter definition

To mathematically describe the motion of the robot, we let $u_k \in U$ be the chamber volumes at time step k where U denotes the control space. The distal tip position of the soft robot at equilibrium of the fluidic chambers is represented by the task space coordinate $p_k \in \mathrm{R}^3$. The discrete time transition between robot tip states and chamber volumes within the time step k to $k+1$ is $p_{k+1} = p_k + \Delta p_k$ and $u_{k+1} = u_k + \Delta u_k$, respectively. **Figure 1** is an illustration of this motion transition between two robot configurations. To describe the new robot tip position after a change of chamber volume from the previous tip position, a forward transition model of the soft robot can be defined as:

$$p_{k+1} = f(p_k, \Delta u_k) \tag{1}$$

The inverse transition model we aim to estimate determines the required change in chamber volume for a movement of the robot tip Δp_k:

$$\Delta u_k = \phi(\Delta p_k, u_k) \tag{2}$$

Note that due to the redundancy of the system, the mapping between the forward and inverse model is not one-to-one, which makes it challenging to find a consistent inverse transition model.

2.3. Overview of online learning algorithm

The objective is to control the soft robot accurately in the task space motion transition coordinate Δs_k, while under unknown loading. For this reason, an online learning algorithm based on the work found in [23] is adapted for usage on our redundant two-segment soft continuum robot. The underlying goal of the algorithm is to estimate the global inverse mapping of the soft robot by combining a large set of localized linear controllers. This technique is based on the key idea that in a localized region of robot configuration, a valid inverse solution can be obtained because the inverse kinematics mapping forms a convex function with respect to the variables $\{\Delta p, p, \Delta u, u\}$.

The proposed control framework is split into two main phases: 1) incremental learning of the local forward models, followed by 2) batch learning of the linear inverse controllers that are only valid within a local region.

The first phase aims to acquire an appropriate local linearization of the forward motion mapping $(p_k, \Delta u_k, u_k) \rightarrow \Delta p_k$, which is nonlinear in general. Such local linearization can determine how many linear models are required to approximate the motion mapping, as well as the valid region of each linear model. Thus, a localized regression method, namely Locally Weighted Projection Regression (LWPR) [32], is employed to learn the forward motion mapping. LWPR provides a piecewise linear function approximation of the nonlinear mapping and automatically determines the valid local regions that each models should affect. The learnt piecewise linear models are in the form:

$$\Delta p_k = [p_k, \Delta u_k, u_k]\hat{\beta}_{FK}^{i} \tag{3}$$

where $\hat{\beta}_{FK}^{i}$ are the linear parameters of the local forward models.

For a non-redundant robotic system, direct usage of the locally learnt forward models is possible due to the one-to-one mapping between the configuration space and operation space, meaning that linear combinations of the locally learnt models will be consistently resolved. However, for a redundant system, invalid solutions would arise from non-convex training data, which brings the need to consider the spatial localisations of the learnt models. Therefore, for each piecewise linear model in the LWPR forward model, we assign a linear controller to approximate the global inverse mapping, using the same local valid regions. We wish to determine the local inverse transition models also positioned in p_k and u_k space; this can be described by the following linear controller:

$$\Delta \mathrm{u}_k = \left[\Delta \mathrm{p}_{ref,k}, p_k, u_k\right]\beta_{IK}^{i} \tag{4}$$

where β_{IK}^{i} is the parameter of the local inverse model, for which its calculation will be described in further detail in the following sections.

2.4. Algorithm implementation

2.4.1. Combining the local controllers

To construct a global controller, such that the required actuator transition Δu_k can be found for a desired tip transition $\Delta p_{ref,k}$, a weighted linear combination of the local inverse transition models is solved in the form:

$$\Delta u_k = \frac{\sum_{i=1}^{n} w^i(p_k, u_k)\left[\Delta p_{ref,k}, p_k, u_k\right]\beta_{IK}^{i}}{\sum_{i=1}^{n} w^i(p_k, u_k)} \tag{5}$$

In the first step of the proposed control framework, the local forward models are learnt through LWPR, which determines an appropriate number of models as well as their locally valid region. Each local model and its region, also known as a receptive field (RF), is governed by a Gaussian kernel, and is weighted to each training data point based on the robot configuration variables p_k and u_k, calculated by the following equation:

$$w^i(p_k, u_k) = \exp\left\{\frac{1}{2}\left(\begin{bmatrix} p_k \\ u_k \end{bmatrix} - c^i\right)^T D^i \left(\begin{bmatrix} p_k \\ u_k \end{bmatrix} - c^i\right)\right\} \tag{6}$$

where i denotes the RF being weighed against, c^i is the RF's center in p_k and u_k space, and D^i is the distance matrix which governs the shape of the RF. Besides the weighted mean of Δu_k, the weighting $w^{\mathrm{i}}(p_k, u_k)$ is also employed to determine relative influence of each data point to the RFs when calculating the global inverse solution.

2.4.2. Selecting null-space behaviour with constrained optimization

When solving for a global controller, there is no guarantee that a consistent inverse solution is found among different local controllers due to the infinite possible solutions in a redundant system. In [23], two particular methods are outlined to overcome the redundancy problem: the first is by biasing the original training data to only allow a single inverse solution, and the second is to introduce a reward/cost function to draw the system to a desired solution. Although the first approach can be useful for simplified tasks, the benefits of using a redundant actuator are mostly lost when restricted to a single solution and can result in the task not being accomplished properly.

Therefore, to ensure consistent null-space behaviour through the second approach, the task is formulated as a constrained optimization problem, where we aimed to minimize the cost function below:

$$C_k(\Delta u_k) = \left(\Delta u_k - \Delta u_{0,k}\right)^T N\left(\Delta u_k - \Delta u_{0,k}\right) \tag{7}$$

where $\Delta u_{0,k} = v(p_k, u_k)$ is the actuator-space attractor that draws the robot to a desired configuration, and $N > 0$ is a positive definite weighting matrix. The cost function in (7) assigns cost to each incoming training data point. It assigns higher cost for incoming Δu_k values that are further away from the desired pose, as defined in the function $v(p_k, u_k)$. This enables systematic resolution of the redundancy problem, while allowing a flexible definition of the user-desired null-space behaviour. A secondary control objective can hence be achieved by associating the null-space attractor $\Delta u_{0,k}$ to the robot characteristics or the task's requirement. One typical example of the null-space behaviour is to attract the robot configuration towards a rest pose u_0, as defined in (8). For a fluid-driven robot, a reasonable choice of rest position is to minimize the overall inflated chamber pressures:

$$\Delta u_{0,k} = K_a(u_k - u_0) \tag{8}$$

where K_a is the attractor gain. Thus, all the training data is given a cost that is based on the configuration space variables u_k, p_k, and Δu_k, with the result that all robot configurations will converge to a consistent solution.

To solve the constrained optimization problem in (7), we first define:

$$\min_{\Delta u_k} \quad C_k(\Delta u_k) \tag{9}$$

$$\text{subject to} \quad \Delta u_k = \left[\Delta p_{ref,k}, p_k, u_k\right]\beta_{IK}^{i}$$

The constrained optimization problem can be solved by converting the cost function into an immediate reward:

$$r(u_k) = \sigma_i \exp[-0.5\sigma_i^2 C_k(\Delta u_k)] \tag{10}$$

where σ_i^2 is the mean cost for a particular local model, used to increase learning speed:

$$\sigma_i^2 = \sum_{k=1}^{n} w^i(p_k, u_k) C_k \bigg/ \sum_{k=1}^{n} w^i(p_k, u_k) \tag{11}$$

where *n* is the total number of training samples. By changing to an immediate reward, we can find a solution which minimizes the following sum:

$$\sum_{k=1}^{n} r(u_k) w^i(p_k, u_k)\left(\Delta u_k - \left[\Delta p_{ref,k}, p_k, u_k\right]\beta_{IK}^{i}\right) \tag{12}$$

The local model parameter in (4) is found by the reward-weighted regression formula, for each local controller model, i:

$$\beta_{IK}^{i} = \left(X^T W^i X\right)^{-1} X^T W^i Y \tag{13}$$

where $X_k = \left[\Delta p_{ref,k}, p_k, u_k\right]$ and $Y_k = [\Delta u_k]$ are rows of the training datasets X and Y, respectively, and $W^i = \operatorname{diag}\left(r(u_1)w_1^i, \dots, r(u_n)w_n^i\right)$. The weighting coefficient matrix, W^i, is a diagonal matrix that applies weighting between each local controller, i, to each training sample. For each local controller, the corresponding weighting coefficient matrix is determined by multiplying the reward of each training data point by the weighting of that point relative to the local controller. The reward as shown in **(10)** regulates the redundancy of the system by giving more importance to training data that exhibits the desired null-space behaviour as defined in **(8)**. The weighting function defined in **(6)** more strongly considers the weights of training points closer in p_k, u_k space to the local controller. Finally, a consistent global controller is derived from the weighted average of all the learnt local inverse models as in (5). The pseudo-code for the online learning process of the controller is detailed in **Algorithm 1**.

Algorithm 1: Algorithm for online learning of consistent inverse motion mapping.

```
1    For each new training sample $[\Delta p_{ref,k}, p_k, \Delta u_k, u_k]$
2        Add $(p_k, \Delta u_k, u_k) \rightarrow \Delta p_{ref,k}$ to the forward model through LWPR
3        Update no. of forward models $m$ and their local weightings $w^i(p_k, u_k)$
4        Evaluate actuator-space attractor:
             $\Delta u_{0,k} = K_a(u_k - u_0)$
5        Compute cost:
             $C_k(\Delta u_k) = \left(\Delta u_k - \Delta u_{0,k}\right)^T N\left(\Delta u_k - \Delta u_{0,k}\right)$
6        For each model $i = 1,2,3, \dots, m$
7            Calculate mean cost:
                 $\sigma_i^2 = \sum_{k=1}^{n} w^i(p_k, u_k) C_k \,/\, \sum_{k=1}^{n} w^i(p_k, u_k)$
8            Calculate reward of each data point:
                 $r(u_k) = \sigma_i \exp\left(-0.5\sigma_i^2 C_k(u_k)\right)$
9            Solve the following weighted regression problem with steps 10-14:
                 $\sum_{k=1}^{n} r(u_k) w^i(p_k, u_k)\left(\Delta u_k - [\Delta p_{ref,k}, p_k, u_k]^T \beta_{IK}^i\right)$
10           Add sample point to weighted regression so that:
                 $X_k = [\Delta p_{ref,k}, p_k, u_k]$
11               $Y_k = [\Delta u_k]$
12               $W^i = \mathrm{diag}\left(r(u_1) w_1^i, \dots, r(u_n) w_n^i\right)$
13           Update inverse mapping parameter by reward-weighted regression:
                 $\beta_{IK}^i = \left(X^T W^i X\right)^{-1} X^T W^i Y$
14       end
15   end
```

3. Experiments, results & discussion

3.1. Experimental platform

The two-segment robot was actuated pneumatically by a set of stepper-motor driven linear actuators. Each linear actuator consisted of a stepper motor coupled to a leadscrew, which controlled the stroke of a pneumatic cylinder. The robot has 6 input degrees-of-freedom (DOFs), with each of the 6 chambers of the soft robot paired with a single linear actuator.

The soft robot was actuated volumetrically, with the stepper motor positions used as a proxy for the actual cylinder volume. Each chamber was pre-pressurised to 0.040 MPa to improve the bending response of the soft robot to input pressure change. An omnidirectional bending angle of up to 100° was attainable by each segment of the soft robot. The tip position of the robot was tracked by an electromagnetic (EM) tracking system (NDI Medical Aurora Tabletop Field Generator). Two 6-DOF tethered sensor (0.8 mm diameter x 9 mm length) were attached to the robot: one at the robot tip, and another at the base of the robot as illustrated in **Figure 3**. The EM tracking system provides a tracking accuracy of 0.80 mm for position, and 0.70° for orientation at an update rate of 40 Hz. It provides the necessary positional data for feedback control and also online learning of the controller. The local online learning algorithm was implemented in the Matlab environment, and applied the open-source library for LWPR [33].

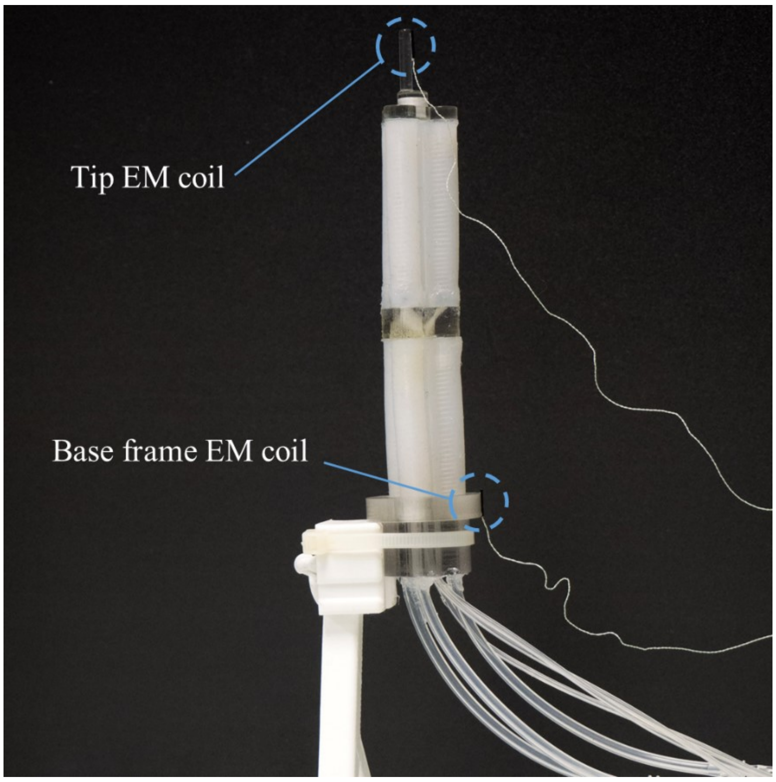


Figure 3. EM positional tracking coils mounted at the robot tip and base. The 6-DoF coil at the base offers a static frame of reference for all the measured tracking data in real time.

3.2. Training data acquisition and model pre-training

In order to effectively generate a functional global controller, pre-training data that sufficiently characterizes the robot's workspace and possible configurations should be obtained.

Learning of the forward models was first performed offline with uniformly distributed random waypoints in actuator-space that were generated and connected by straight line trajectories. This formed the pre-training exploration data. For the purpose of this study, 80 random waypoints were sufficient to provide a large enough selection of forward mappings so that consistent inverse controllers could be learnt. An alternative controller initialization can be achieved by motor babbling, where small, random movements of the robot are used to learn the controller online. However, offline pre-training was favoured in this study to better evaluate the null-space behaviour of the controller. Additionally, if purely online learning is used, redundant configurations are less likely to be observed, limiting the manoeuvrability of the robot and generality of the system. In this study, the tip position, p_k, is a 3x1 vector of the x-y-z Cartesian tip position tracked by the tip EM coil as in **Figure 3.** The chamber volumes, u_k, are a 6x1 vector which describes the current inflation state of each chamber of the robot (3 chambers per robot segment).

Validation of the learned forward models was performed by splitting the obtained training data into a training and test set, at 80% and 20%, respectively. The root-mean-square error (RMSE) of the predicted forward model outputs, Δp, versus the number of training iterations (epochs) for the training data, test data, and combined data is shown in **Figure 4**. A total of 15 training iterations were processed, resulting in 139 receptive fields generated for each output dimension. The RMSE of all three types of data was lowest at 11 epochs, with the testing set converging to approximately 1 mm. To further validate the pre-trained model, the predicted outputs of the global inverse model was compared against the learned forward model using the combined test set. The resulting regression plots and histograms for each task space dimension are shown in **Figure 5**. The error bounds for each dimension of the learned inverse model were under 0.5 mm.

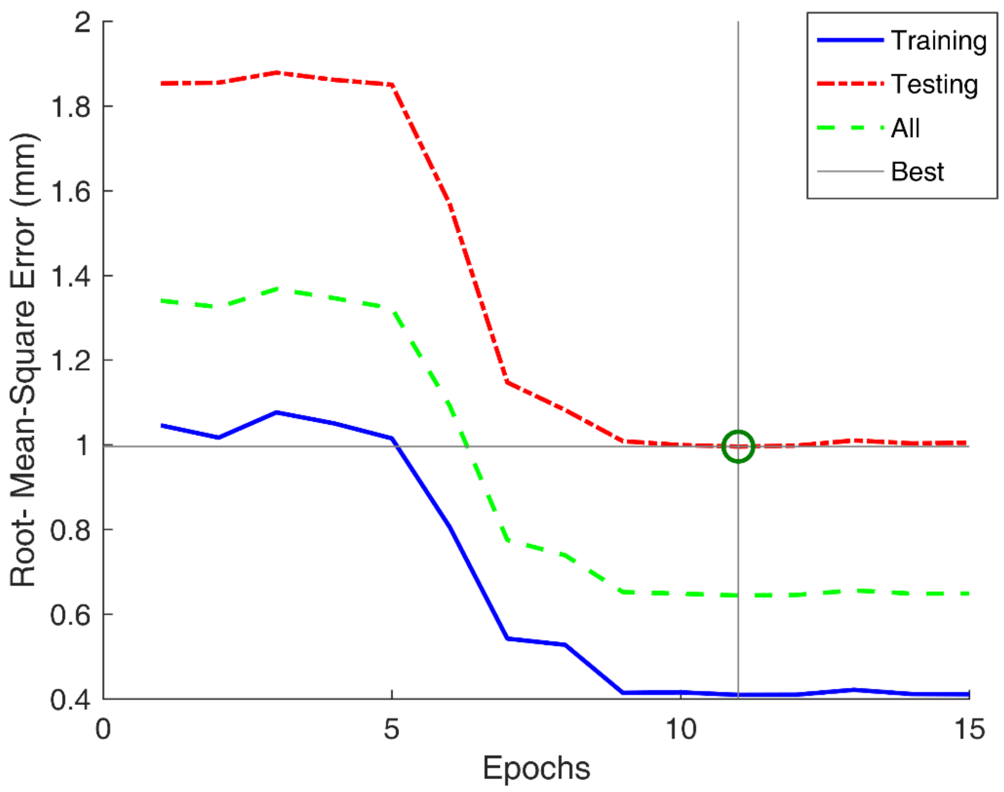


Figure 4. Validation of the forward model that was trained through LWPR. The error reached the lowest value at the 11th epoch. The training and testing data was split 80% and 20%, respectively, of the original data set. The root-mean-square error is with respect to the forward model output, Δp. The error of all three sets of data was lowest at 11 epochs, indicated by the vertical grey line.

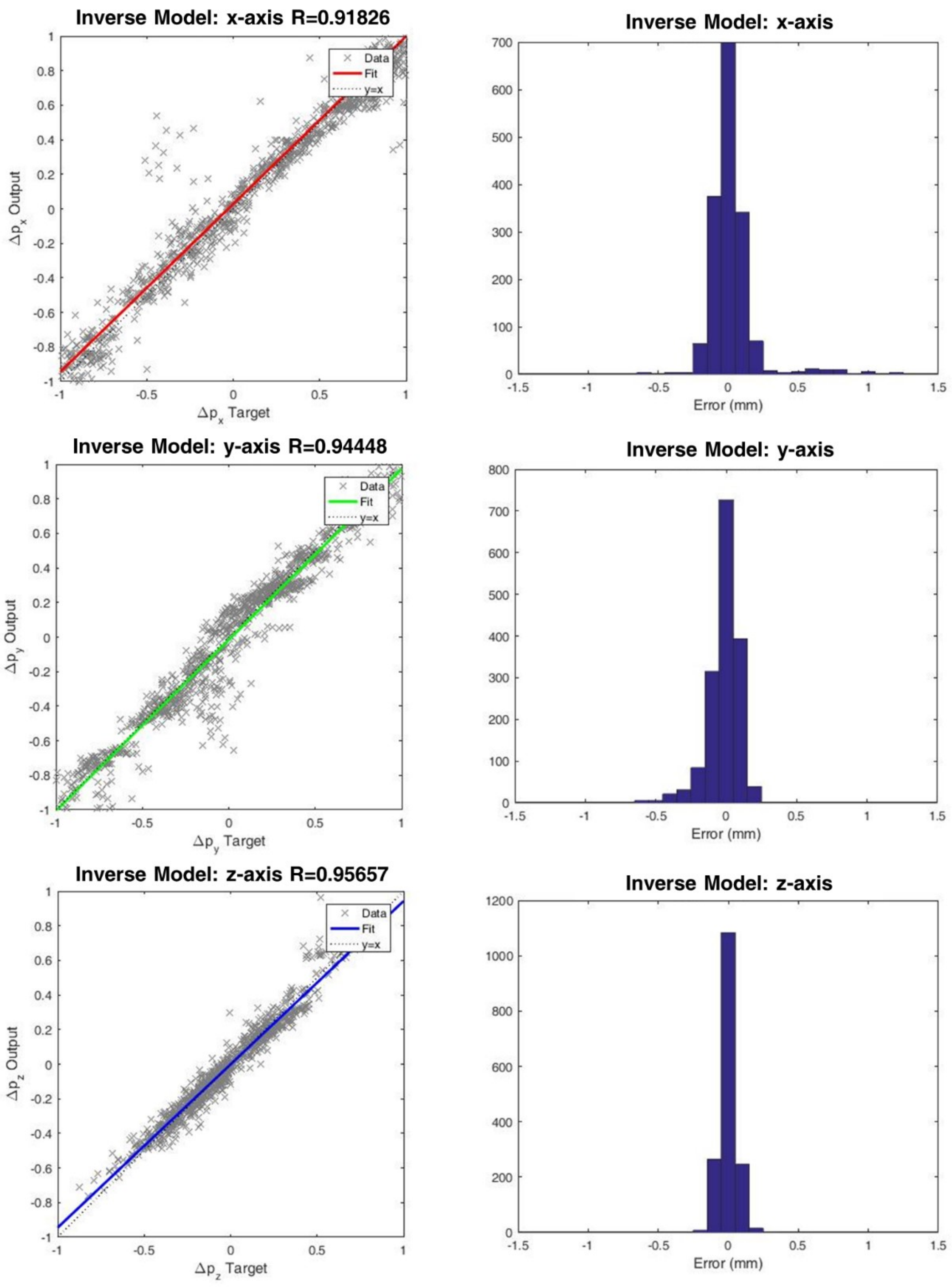


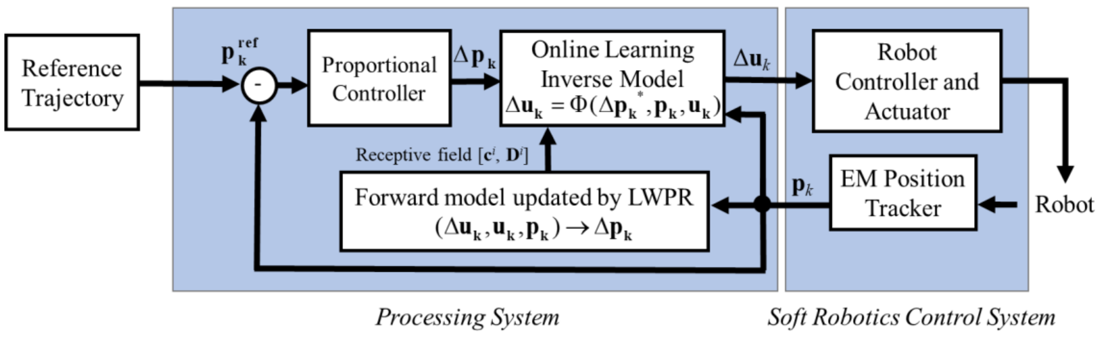
Figure 5. Regression plots **(Left)** and histograms **(Right)** for the tip transition variable Δp in each coordinate axis (x, y, z).

3.3. Controller implementation

Evaluation of the proposed control framework is performed on a two-segment soft continuum robot. Two tracking exercises are presented to assess the accuracy of the learnt inverse kinematics as well as its ability to adapt to an unknown disturbance. The trajectory following is achieved through resolved motion rate control [34], and the desired task space displacement Δs_k is defined by the proportional feedback controller:

$$\Delta p_k = K_P\left(p_k^{ref} - p_k\right) \tag{14}$$

where K_P is the proportional gain, p_k^{ref} is the desired tip position, p_k is the 3D tip position at the current time-step. This desired task space displacement is the input to the learned global controller, which outputs the estimated stepper motor commands Δu. For the accuracy evaluation, the tracking error is calculated by the Euclidean distance between the desired tip position and the achieved tip position at each time step. The block diagram of the implemented control loop is shown in **Figure 6**.



Reference Trajectory
$\mathbf{p}_k^{ref}$
-
Proportional Controller
$\Delta\mathbf{p}_k$
Online Learning Inverse Model
$\Delta\mathbf{u}_k = \Phi(\Delta\mathbf{p}_k^*, \mathbf{p}_k, \mathbf{u}_k)$
$\Delta\mathbf{u}_k$
Robot Controller and Actuator
Receptive field $[\mathbf{c}^i, \mathbf{D}^i]$
Forward model updated by LWPR
$(\Delta\mathbf{u}_k, \mathbf{u}_k, \mathbf{p}_k) \rightarrow \Delta\mathbf{p}_k$
$\mathbf{p}_k$
EM Position Tracker
Robot
Processing System
Soft Robotics Control System

Figure 6. Schematic showing the proposed control system architecture that facilitates online updating of the learned controller. The controller is constantly updated with incoming real-time data provided by the EM position tracker.

3.4. Trajectory tracking experiments –static tip load

A comparison between the online-updating controller and the offline-learned controller was performed by trajectory tracking of a 3D path under two scenarios: 1) only using the pre-trained model with no online learning (‘offline’), 2) online learning while an unknown tip mass is added to the robot.

The goal of these two experiments is to evaluate the effects of updating the pre-trained model in an online manner and providing a comparison to only using the pre-trained model. The test trajectory is a rectangular shape of sides 25 mm x 100 mm projected to the 3D workspace of the robot, which was approximated from the pre-training data. In each test scenario, the controller was run for 3 complete cycles, which had a total runtime of 400 s. Initially in each test, the robot was allowed to track to the first point of the desired trajectory until the error converged, at which point data acquisition was initiated and the desired trajectory point began to increment. The same error-proportional gain and pre-trained model was used for both experiments. In the following experimental sections, *offline* denotes the absence of online learning during trajectory tracking, and *online* denotes that online learning was enabled.

3.4.1. No tip load - offline

For the first experiment, only the offline pre-trained model was implemented into the robot controller. Online learning was disabled, and robot was free to track the target trajectory with no additional external disturbances. The tracking performance of the first experiment is presented in **Figure 7**. The average error was not observed to improve between the first and last trajectory cycle, with a mean absolute error of ±4.56 mm, and ±5.53 mm, respectively. A recurring error pattern could be seen in each cycle, which depicts the repeatability of the learned controller. The tracking error could be attributed to a lack of densely populated receptive fields in those regions resulting in poorly defined inverse solutions. Other controller errors are also expected due to the hysteretic effects of the soft robot body which the proportional controller could not compensate for.

No tip load, **OFFLINE**

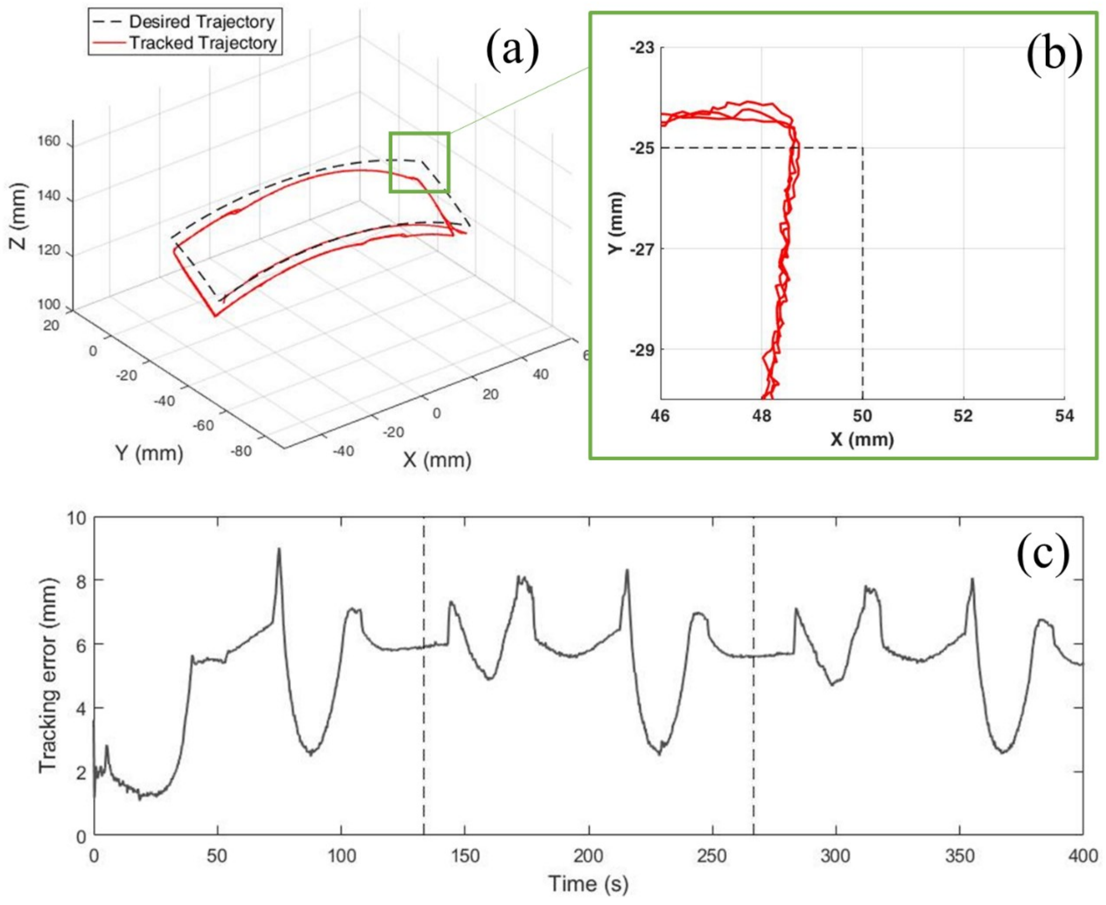


Figure 7. Experimental results for trajectory tracking with no additional tip loading using the pre-trained model with no online learning. **(a)** The actual tracked trajectory overlaid on the desired trajectory. **(b)** Close-up view of the corner tracking. **(c)** The Euclidean tip tracking error over time. The dotted lines indicate the start and end of each trajectory cycle.

3.4.2. Static tip load – online learning

For the second experiment, an additional tip mass was added to the robot tip, as illustrated in **Figure 2(b)**. The total additional mass was 14.2 g and was not previously presented to the model during pre-training. The same pre-trained model applied in experiment 1 was used as a baseline for the online learning in this experiment. When online learning, a fixed number of training points are used to weight the influence of the local models. For this experiment, a maximum of 425 incoming training points was used in a first-in-first-out basis, where the oldest data points were removed first when exceeding the maximum of 425. For each cycle of trajectory tracking, approximately 400 new training points were accumulated. The average online update frequency was 23 Hz. With the additional tip weight, the starting tracking error increased from approximately 2 mm as seen in the first experiment to 5 mm. By the inclusion of online learning in this controller, the real-time data obtained from the tracked tip position and actuator volumes could be input to the online learning algorithm, enabling incremental improvements to the overall learned inverse model. This could be observed in the results presented in **Figure 8** and **Table 1**. The mean absolute tracking error of every cycle could be seen to decrease significantly, starting at ±4.42 mm in the first cycle and reducing to ±1.63 mm in the third cycle.

Overall, online learning of the original pre-trained model could be seen to improve the tracking performance through continuous online updating of the inverse model, even in the presence of a previously unknown external disturbance.

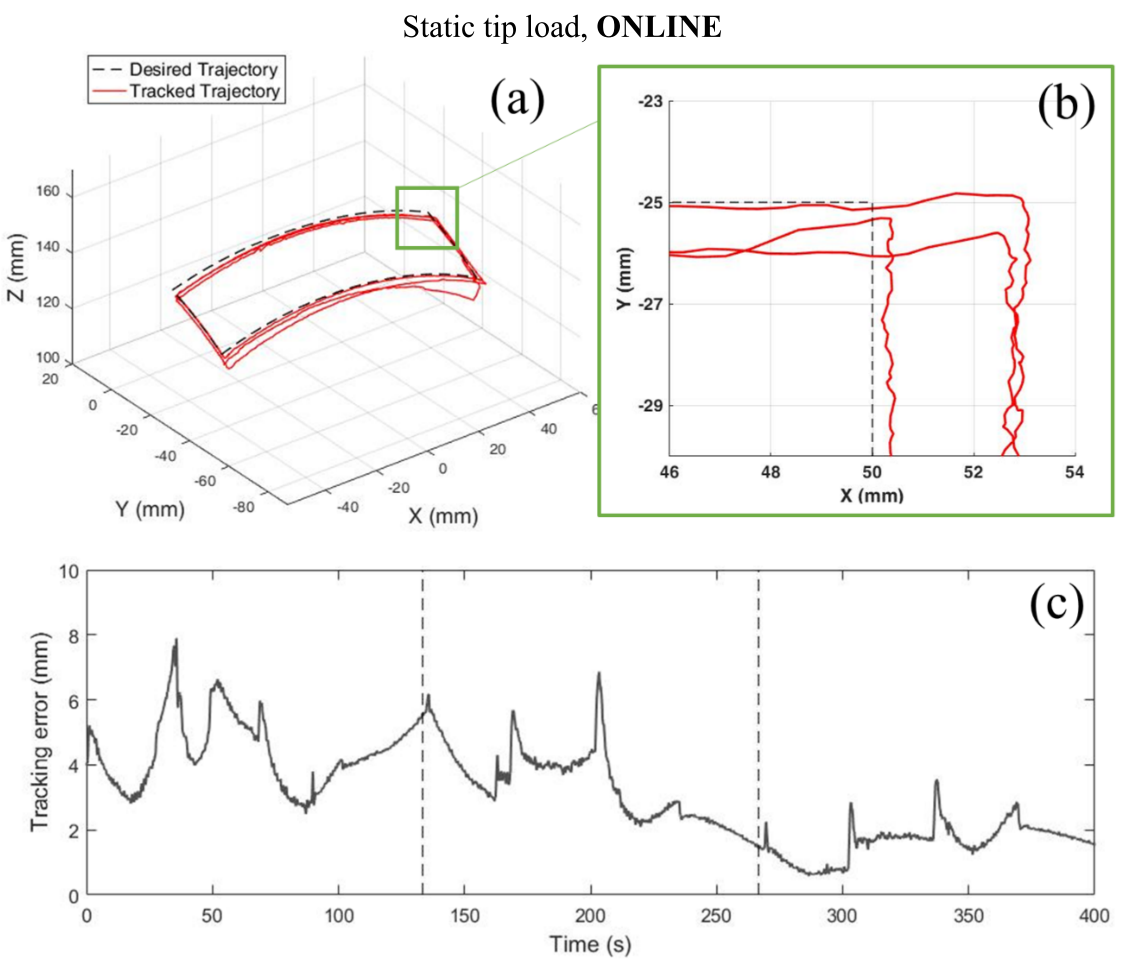


Figure 8. Experimental results for trajectory tracking with additional tip loading, using the pre-trained model and updated with online learning. The algorithm is able to adapt to the tip disturbance in real-time, providing improved tracking performance. **(a)** The actual tracked trajectory overlaid on the desired trajectory. **(b)** Close-up view of the corner tracking. **(c)** The Euclidean tip tracking error over time. The error can be seen to consistently decrease over the 3 cycles.

Table 1. Summary of trajectory tracking performance for no tip load (offline) and static tip load (online) scenarios.

Controller setting	*Error absolute mean and standard deviation (mm)*			*Maximum absolute error (mm)*		
	1st cycle	*2nd cycle*	*3rd cycle*	*1st cycle*	*2nd cycle*	*3rd cycle*
No tip load ***OFFLINE***	±4.56 (σ = 1.99)	±5.72 (σ = 1.18)	±5.53 (σ = 1.12)	±9.02	±8.34	±8.07
Static tip load ***ONLINE***	±4.42 (σ = 1.08)	±3.33 (σ = 1.14)	±1.63 (σ = 0.54)	±7.90	±6.87	±3.55

3.5. Trajectory tracking experiments – varying tip load

To further investigate the control framework's behaviour, a series of trajectory following tasks were performed while a variable fluid tip load as in **Figure 2(c)** was added to the robot tip. The fluid tip has an empty weight of 14 g, and has a maximum weight of 32 g when full (corresponding to an internal volume of 18 mL). Three experiments were performed for 3D trajectory tracking with the varying tip load: 1) only the empty (0% filled) fluid tip container added to the robot tip with no online learning, 2) increasing fluid load with no online learning, 3) increasing fluid load with online learning enabled.

For this set of experiments, the test trajectory is a rectangular shape with sides 40 mm x 60 mm that was projected on the workspace of the robot. The robot was allowed to run for 4 cycles. The same control parameters were used in the three scenarios, with only the option of online learning differing between them. The training data used for the pre-trained model were based on 320 random waypoints, which resulted in 352 receptive fields generated. A tabulated summary of the tracking results is shown in **Table 2**, and the actual tracked trajectories and absolute Euclidean tracking errors over time are shown in **Figure 10**.

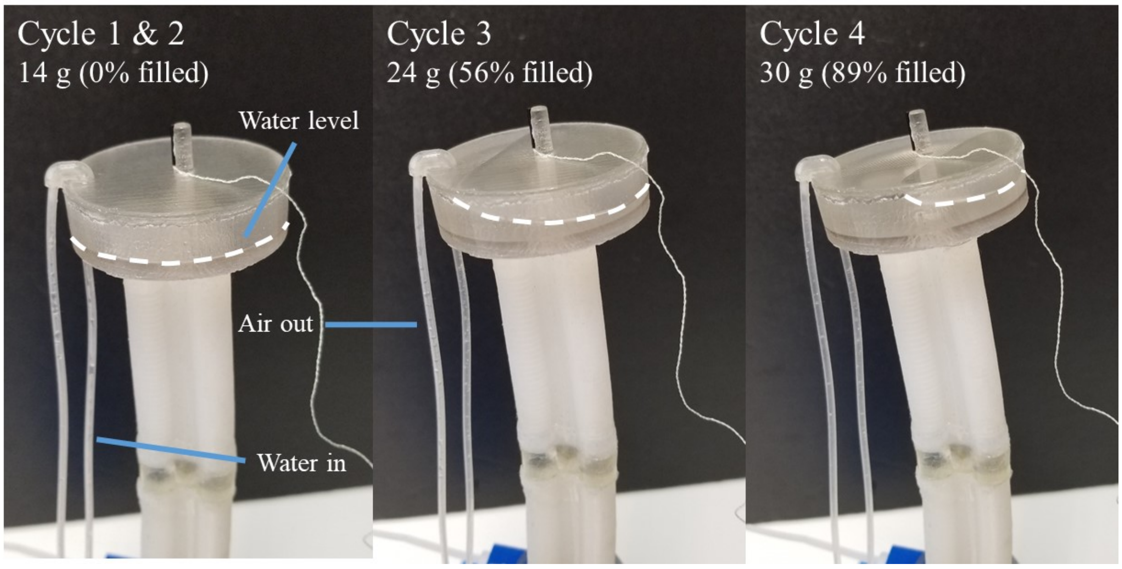


Figure 9. Variable fluid tip load used for experiments. The tip load is varied by injecting water at an approximate rate of 0.6 mL/s at the beginning of cycle 3 and 4. The empty fluid tip is 14 g, and has a maximum weight of 32 g when full. For the trajectory tracking experiments under varying tip load, 10 g is added to the tip load in cycle 3, and 6 g is added in cycle 4.

3.5.1. Empty fluid container tip – offline

In the first scenario, only the pre-trained model was used, with no online updates made during the experiment. This '*offline'* controller setting is akin to implementing a model-based kinematic model, e.g. PCC, where no online updates are made to the model during runtime. The fluid container tip was empty for all four cycles of trajectory tracking, weighing approximately 14 g. Overall, the tracking performance for each cycle was seen to be relatively periodic as seen in **Figure 10(a)**, with the mean absolute error remaining around the 6-7 mm range. No notable improvement could be seen between each cycle, however the mean absolute and max absolute error increased between cycle 1 and 2. This is likely because the robot was allowed to track to the first point until error converged before data acquisition began and the remainder of the trajectory was tracked. The primary source of error in the trajectory tracking can be attributed to the additional unmodelled tip load due to the empty fluid container tip. The tip load induces unmodelled loading to the entire robot body, creating a large disparity between the original pre-trained kinematic model's estimation, and the actual robot configuration.

Unlike the static tip load experiment in **Section 3.4.2.** where the tracking error would reduce over each cycle due to the online learning, we can see a consistent offset of the tracked trajectory versus the desired trajectory.

3.5.2. Increasing fluid load - offline

For the second scenario, a varying tip load was applied to the robot tip by increasing the fluid volume in the fluid tip. To fill the fluid tip, water was injected through the 'water in' tube labelled in **Figure 9** at a rate of approximately 0.6 mL/s. For the first two cycles of trajectory tracking, the fluid tip was empty (0% filled), which is the same conditions as the first two cycles of the previous experiment in **Section 3.5.1.** In cycle 3 and 4, the fluid levels were increased in accordance to **Figure 9**: from the beginning of cycle 3, an additional 10 g of water was added to the fluid tip at a rate of ~0.6 mL/s, with a total tip load of 24 g. This corresponds to 56% of the entire fluid tip cavity filled. At the start of cycle 4, an additional 6 g of water was added to the fluid tip, corresponding to a total additional tip weight of 30 g, or 89% filled. At 30 g, the fluid tip is an additional 72% of the robot body mass (41.71 g), presenting substantial loading to the robot tip. Depicted in **Figure 9** is the deformation caused by the fluid load when the robot is at the *neutral, unactuated* position. When tracking the trajectory, the moment caused by the load is larger due to the robot bending, and induces significant unmodelled deformation.

In the first two cycles, it can be seen that the tracking error and path taken was very similar to the results in **Section 3.5.1.** This is because the controller setting and tip load are the same between the two experiments in the first two cycles. When fluid level was increased in the tip load in cycle 3, the errors also increased, eventually leading to instability in the 4th cycle which is seen in the left-hand side of the tracked trajectory path in **Figure 9(b)**. A major source of the instability can be attributed to the inability of the offline controller to track the desired trajectory due to large corrective overshoot from the error induced by the tip load. Also, the fluid tip is only partially filled, leading the centre of mass to constantly change as the robot configuration changes, further amplifying any instability.

3.5.3. Increasing fluid load – online learning

In the third scenario, online learning was enabled during trajectory tracking while the fluid load was increased in accordance to **Figure 9**. The same pre-trained model use in the previous two scenarios was also used here. For online learning, the maximum number of data points was set to 550. For each cycle of trajectory tracking, approximately 300 new training points were accumulated. Similar behaviour to the online static tip load experiment in **Section 3.4.2.** can be seen, with the error reducing in each cycle. Over the four cycles, the average tracking error reduced from ±4.16 mm to ±0.98 mm. In contrast to the independent test in **Section 3.5.2.** that demonstrates offline tracking with increasing fluid load, the online learning controller was able to avoid instability, and even reduce the overall tracking error. The tracked trajectory and errors can be seen in **Figure 9(c)** and **Figure 10(c)**, respectively.

In this third scenario, the update rate was limited to approximately 7 Hz, i.e. each online update took ~0.143 s to complete. A notable limitation of the online learning is that the update speed is directly tied to the number of stored training data points and local models, because the weighting of each data point to each local controller must be made at each update, in accordance to the weighting function **(6)**. The online update rate for this controller was significantly lower than that in **Section 3.4.1.** because 352 local controllers were used in the pre-trained model, compared to 139 local controllers. This extensive computation time is a bottleneck for the online learning framework, as too many local models or stored data points would cause the update rate to slow to impractical speeds. A potential method for easing the computational intensity is through the use of training data sparsification. This would involve limiting and selectively processing training data obtained online so that only the 'most important' training points would be used.

In general, the online learning experiments performed in this study highlight the difficulties of using a standard, non-adaptive controller for control of a soft robot under external disturbance. High unmodelled morphological change can cause typical feedback controllers utilizing Jacobians to exhibit inaccurate or unstable trajectory tracking because they assume low error configurations, which is not true for soft robots under any notable levels of loading. However, through online learning the robot configuration error can be minimized by effectively updating the Jacobian to adapt to disturbances based on real-time tracking data. For more extreme cases of deformation, the controller can potentially fail to track the target trajectory. This could be caused by the robot configuration lying far outside of the pre-trained local linear models, or due to limitations of the robot actuation (e.g. upper pressure limit of the robot chambers).

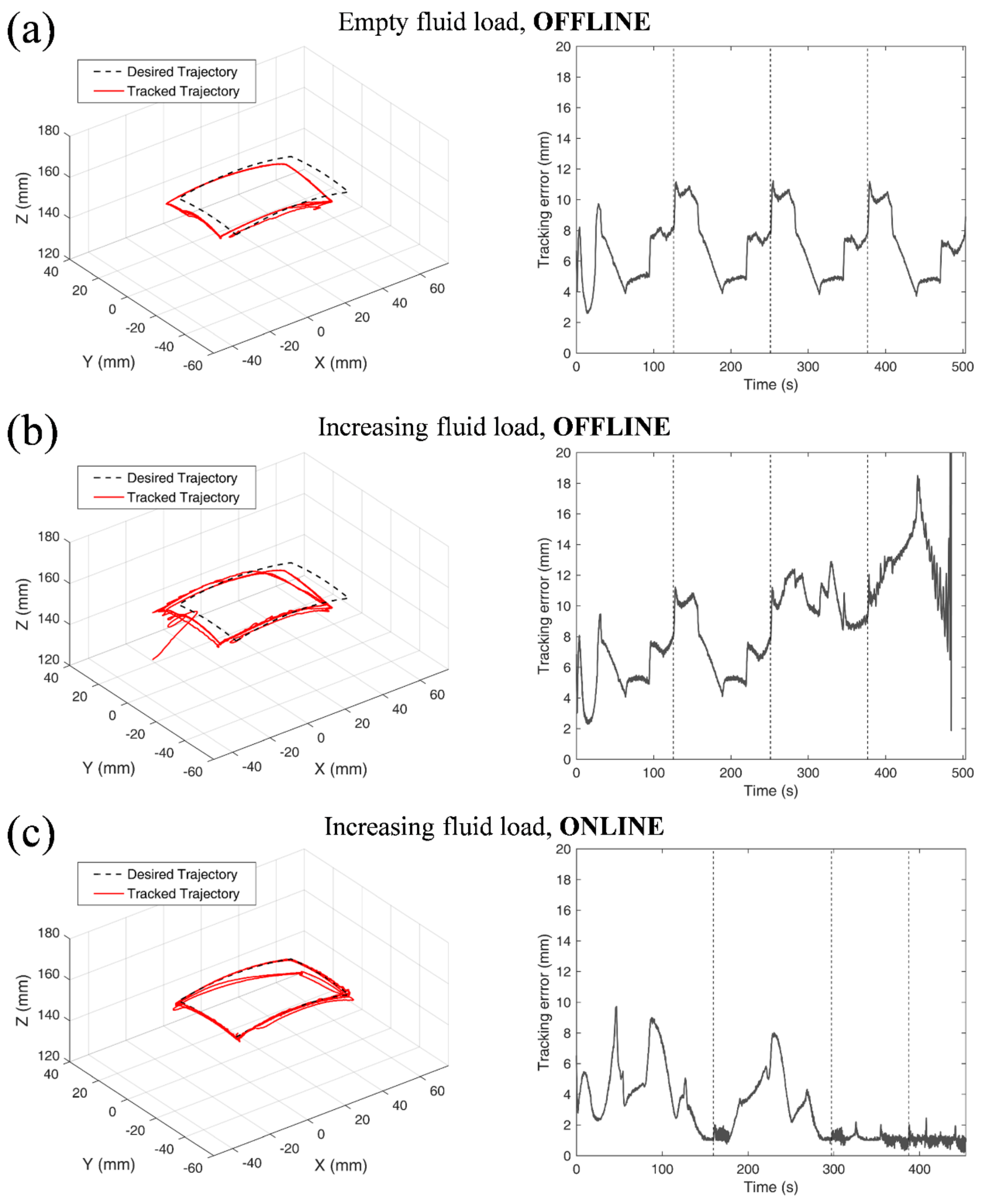


Figure 10. Experimental results for trajectory tracking with different controller and tip load conditions. The robot was allowed to follow the trajectory for 4 cycles, as indicated by the vertical dotted lines. **(a)** Offline trajectory tracking with empty fluid container tip weighing 14 g. A repeating error pattern is observed due to the static load. **(b)** Offline trajectory tracking with fluid tip load increased by 10 g in cycle 3, and 6 g in cycle 4. The tracking error increased with fluid load, becoming unstable in the 4th cycle. **(c)** Trajectory tracking with online learning enabled. The fluid tip load was increased by 10 g in cycle 3 and 6 g in cycle 4. Instability was avoided and error was also reduced.

Table 2. Summary of trajectory tracking performance for experiments with variable fluid tip load.

	Error absolute mean and standard deviation (mm)				*Maximum absolute error (mm)*			
Controller setting	*1st cycle*	*2nd cycle*	*3rd cycle*	*4th cycle*	*1st cycle*	*2nd cycle*	*3rd cycle*	*4th cycle*
Empty fluid load **OFFLINE**	±6.01 (σ = 1.77)	±7.19 (σ = 2.17)	±7.08 (σ = 2.12)	±6.94 (σ = 2.09)	±9.73	±11.16	±11.23	±11.19
Increasing fluid **OFFLINE**	±5.82 (σ = 1.64)	±7.24 (σ = 2.00)	±10.36 (σ = 1.22)	±12.84 (σ = 2.20)	±9.47	±11.22	±12.88	unstable
Increasing fluid **ONLINE**	±4.16 (σ = 2.19)	±3.10 (σ = 1.94)	±1.07 (σ = 0.24)	±0.98 (σ = 0.26)	±9.72	±8.01	±2.19	±2.44

4. Conclusions & future work

In this study, we proposed and validated an online learning-based control framework to control a hyper-redundant two-segment soft robot in a 3D positional task space. The use of an online data-driven learning approach enables high adaptability to unmodelled characteristics both internal and external to the soft robot, while resolving consistent redundancy behaviour. A pre-trained inverse model was learned for the two-segment soft robot and applied in a proportional motion rate controller. For the static tip load case with online learning, the robot controller was able to adapt quickly to an unknown static tip weight/load, with the average absolute error reducing from ±4.42 mm to ±1.63 mm over three cycles of the tested 3D trajectory. A more demanding trajectory tracking task was also performed with a varying fluid tip load. Without online learning, the robot became unstable and was unable to compensate for the maximum weight by the 4th cycle. However, with the addition of online learning the robot was not only able to avoid instability, but was also able to reduce the mean absolute tracking error to < 1 mm.

Our future work includes further extension of the proposed control framework to three or more segments of a soft robot and incorporation of a greater number of task space variables to improve the manipulability of the robotic system. In the future experimental settings, we would also aim to replace the tethered EM tracking system with a self-contained sensing modality, such as a camera [35] or a fibre optic system [36] such as those based on fiber Bragg gratings [37]. This would allow evaluation of the proposed learning algorithm in application-based scenarios. Additionally, secondary objectives can be incorporated into the algorithm's cost calculation, such as obstacle avoidance, and could provide customizability for task-specific performance. Improvement to the computational speed of the learning framework can also be made, with a possible solution being sparsification, which could be used to select training data so that only the most relevant data is used.

In terms of application, soft manipulators are inherently non-ferromagnetic and have more easily disposable bodies which present interesting opportunities to be used in harsh environments where traditional robots are unable to be used. An example of this is under magnetic resonance imaging (MRI), where the strong magnetic field involved disallows any traditional robots. Towards MRI-guided robotic interventions [38], the integration of the proposed online learning algorithm and a soft robotic manipulator could enable safe and adaptive navigation in surgery.

Disclosure statement

No potential conflict of interest was reported by the authors.

Acknowledgments

This work is supported in parts by Signate Life Sciences (Hong Kong) Limited, the Croucher Foundation, the Research Grants Council (RGC) of Hong Kong (Ref. No. 27209151, No. 17227616, and No. 17202317).

References

[1] C. Laschi, B. Mazzolai, and M. Cianchetti, "Soft robotics: Technologies and systems pushing the boundaries of robot abilities," *Science Robotics,* vol. 1, no. 1, 2016.
[2] D. Rus and M. T. Tolley, "Design, fabrication and control of soft robots," *Nature,* vol. 521, p. 467, 05/27/online 2015.
[3] E. Tumino, "Endotics system vs colonoscopy for the detection of polyps," *World Journal of Gastroenterology,* vol. 16, no. 43, 2010.
[4] B. Vucelic *et al.*, "The aer-o-scope: proof of concept of a pneumatic, skill-independent, self-propelling, self-navigating colonoscope," *Gastroenterology,* vol. 130, no. 3, pp. 672-7, Mar 2006.
[5] F. Cosentino, E. Tumino, G. R. Passoni, E. Morandi, and A. Capria, "Functional evaluation of the endotics system, a new disposable self-propelled robotic colonoscope: in vitro tests and clinical trial," *The International journal of artificial organs,* vol. 32, no. 8, pp. 517-527, 2009.
[6] M. Cianchetti *et al.*, "Soft Robotics Technologies to Address Shortcomings in Today's Minimally Invasive Surgery: The STIFF-FLOP Approach," *Soft Robotics,* vol. 1, no. 2, pp. 122-131, 2014.
[7] C. Laschi, M. Cianchetti, B. Mazzolai, L. Margheri, M. Follador, and P. Dario, "Soft Robot Arm Inspired by the Octopus," *Advanced Robotics,* vol. 26, no. 7, pp. 709-727, 2012.
[8] M. T. Tolley *et al.*, "A resilient, untethered soft robot," *Soft Robotics,* vol. 1, no. 3, pp. 213-223, 2014.
[9] R. J. Webster and B. A. Jones, "Design and Kinematic Modeling of Constant Curvature Continuum Robots: A Review," *The International Journal of Robotics Research,* vol. 29, no. 13, pp. 1661-1683, 2010.
[10] I. D. Walker, "Continuous Backbone “Continuum” Robot Manipulators," *ISRN Robotics,* vol. 2013, pp. 1-19, 2013.
[11] T. George Thuruthel, Y. Ansari, E. Falotico, and C. Laschi, "Control Strategies for Soft Robotic Manipulators: A Survey," *Soft robotics,* 2018.
[12] H. Wang, C. Wang, W. Chen, X. Liang, and Y. Liu, "Three-dimensional dynamics for cable-driven soft manipulator," *IEEE/ASME Transactions on Mechatronics,* vol. 22, no. 1, pp. 18-28, 2017.
[13] P. Qi, C. Liu, A. Ataka, H. K. Lam, and K. Althoefer, "Kinematic Control of Continuum Manipulators Using a Fuzzy-Model-Based Approach," *IEEE Transactions on Industrial Electronics,* vol. 63, no. 8, pp. 5022-5035, 2016.
[14] B. A. Jones and I. D. Walker, "Kinematics for multisection continuum robots," *IEEE Transactions on Robotics,* vol. 22, no. 1, pp. 43-55, 2006.
[15] D. B. Camarillo, C. F. Milne, C. R. Carlson, M. R. Zinn, and J. K. Salisbury, "Mechanics Modeling of Tendon-Driven Continuum Manipulators," *IEEE Transactions on Robotics,* vol. 24, no. 6, pp. 1262-1273, 2008.
[16] R. J. Webster, J. P. Swensen, J. M. Romano, and N. J. Cowan, "Closed-form differential kinematics for concentric-tube continuum robots with application to visual servoing," in *Experimental Robotics*, 2009, pp. 485-494: Springer.
[17] S. Neppalli, M. A. Csencsits, B. A. Jones, and I. D. Walker, "Closed-form inverse kinematics for continuum manipulators," *Advanced Robotics,* vol. 23, no. 15, pp. 2077-2091, 2009.
[18] D. Trivedi, A. Lotfi, and C. D. Rahn, "Geometrically exact dynamic models for soft robotic manipulators," in *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, 2007, pp. 1497-1502: IEEE.

[19] F. Renda, M. Giorelli, M. Calisti, M. Cianchetti, and C. Laschi, "Dynamic model of a multibending soft robot arm driven by cables," *IEEE Transactions on Robotics,* vol. 30, no. 5, pp. 1109-1122, 2014.
[20] M. Rolf, J. J. Steil, and M. Gienger, "Goal babbling permits direct learning of inverse kinematics," *IEEE Transactions on Autonomous Mental Development,* vol. 2, no. 3, pp. 216-229, 2010.
[21] A. T. Hasan, A. M. S. Hamouda, N. Ismail, and H. Al-Assadi, "An adaptive-learning algorithm to solve the inverse kinematics problem of a 6 DOF serial robot manipulator," *Advances in Engineering Software,* vol. 37, no. 7, pp. 432-438, 2006.
[22] R. Köker, C. Öz, T. Çakar, and H. Ekiz, "A study of neural network based inverse kinematics solution for a three-joint robot," *Robotics and autonomous systems,* vol. 49, no. 3-4, pp. 227-234, 2004.
[23] J. Peters and S. Schaal, "Learning to Control in Operational Space," *The International Journal of Robotics Research,* vol. 27, no. 2, pp. 197-212, 2008.
[24] D. Nguyen-Tuong and J. Peters, "Model learning for robot control: a survey," *Cogn Process,* vol. 12, no. 4, pp. 319-40, Nov 2011.
[25] M. Giorelli, F. Renda, M. Calisti, A. Arienti, G. Ferri, and C. Laschi, "Neural network and jacobian method for solving the inverse statics of a cable-driven soft arm with nonconstant curvature," *IEEE Transactions on Robotics,* vol. 31, no. 4, pp. 823-834, 2015.
[26] A. Melingui, O. Lakhal, B. Daachi, J. B. Mbede, and R. Merzouki, "Adaptive neural network control of a compact bionic handling arm," *IEEE/ASME Transactions on Mechatronics,* vol. 20, no. 6, pp. 2862-2875, 2015.
[27] O. Lakhal, A. Melingui, and R. Merzouki, "Hybrid approach for modeling and solving of kinematics of a compact bionic handling assistant manipulator," *IEEE/ASME Transactions on Mechatronics,* vol. 21, no. 3, pp. 1326-1335, 2016.
[28] 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*, 2016, vol. 2, pp. 303-310.
[29] D. Braganza, D. M. Dawson, I. D. Walker, and N. Nath, "A neural network controller for continuum robots," *IEEE transactions on robotics,* vol. 23, no. 6, pp. 1270-1277, 2007.
[30] K. H. Lee *et al.*, "Nonparametric Online Learning Control for Soft Continuum Robot: An Enabling Technique for Effective Endoscopic Navigation," *Soft Robot,* vol. 4, no. 4, pp. 324-337, Dec 2017.
[31] A. D'Souza, S. Vijayakumar, and S. Schaal, "Learning inverse kinematics," in *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, 2001, vol. 1, pp. 298-303: IEEE.
[32] S. Vijayakumar, A. D'souza, and S. Schaal, "Incremental online learning in high dimensions," *Neural computation,* vol. 17, no. 12, pp. 2602-2634, 2005.
[33] S. Klanke, S. Vijayakumar, and S. Schaal, "A library for locally weighted projection regression," *Journal of Machine Learning Research,* vol. 9, no. Apr, pp. 623-626, 2008.
[34] D. E. Whitney, "Resolved motion rate control of manipulators and human prostheses," *IEEE Transactions on Man-machine Systems,* vol. 10, no. 2, pp. 47-53, 1969.
[35] B. Espiau, F. Chaumette, and P. Rives, "A new approach to visual servoing in robotics," *IEEE Transactions on Robotics and Automation,* vol. 8, no. 3, pp. 313-326, 1992.

[36] S. Sareh, Y. Noh, M. Li, T. Ranzani, H. Liu, and K. Althoefer, "Macrobend optical sensing for pose measurement in soft robot arms," *Smart Materials and Structures,* vol. 24, no. 12, p. 125024, 2015.
[37] S. C. Ryu and P. E. Dupont, "FBG-based shape sensing tubes for continuum robots," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, 2014, pp. 3531-3537: IEEE.
[38] K. H. Lee *et al.*, "MR Safe Robotic Manipulator for MRI-guided Intra-cardiac Catheterization," *IEEE/ASME Transactions on Mechatronics,* vol. PP, no. 99, pp. 1-1, 2018.