IEEE Robotics and Automation Letters (RAL) paper presented at the
2020 IEEE International Conference on Robotics and Automation (ICRA)
31 May - 31 August, 2020. Paris, France

A Probabilistic Model-based Online Learning Optimal Control Algorithm for Soft Pneumatic Actuators

Zhi Qiang Tang[1], Ho Lam Heung[1], Kai Yu Tong[1] and Zheng Li[2]

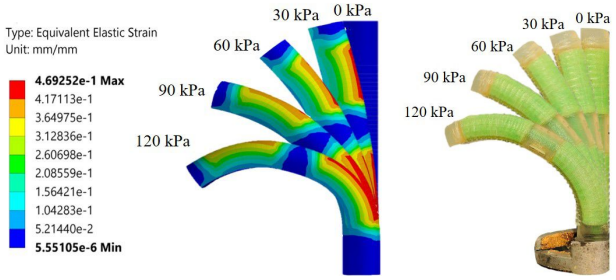
***Abstract*— Soft robots are increasingly being employed in different fields and various designs are created to satisfy relevant requirements. The wide ranges of design bring challenges to soft robotic control in that a unified control framework is difficult to derive. Traditional model-driven approaches for soft robots are usually design-specific which highly depend on specific design structures. Our approach to such challenges involves a probabilistic model that learns a mapping from the soft actuator states and controls to the next states. Then an optimal control policy is derived by minimizing a cost function based on the probabilistic model. We demonstrate the efficiency of our approach through simulations with parameter analysis and real-robot experiments involving three different designs of soft pneumatic actuators. Comparisons with previous model-based controllers are also provided to show advantages of the proposed method. Overall, this work provides a promising design-independent control approach for the soft robotics community.**

***Index Terms*— Modeling, Control, and Learning for Soft Robots, Model Learning for Control, Optimization and Optimal Control**

I. Introduction

Soft robots are increasingly emerging nowadays and have many applications in a variety of fields, including astronautics [1], deep sea exploration [2], rehabilitation [3], [4], surgery [5], [6], etc. To satisfy requirements from various applications, soft robots are designed with different materials, shapes and structures. Such diverse design architectures bring challenges to soft robotic control. Although different methods have been explored to control soft robots, the development of control algorithms still remains a challenge due to the nonlinear characteristics, infinite degrees-of-freedom and wide ranges of design [7].

In this paper, we investigate a probabilistic modeling approach to control soft robots. Previous relevant works are about data-driven approaches for soft robots. In [8] linear regression and artificial neural network models were used in a closed-loop PID controller to control the bending angle of a soft pneumatic actuator. However, the derived empirical models were static models trained on offline dataset, which may not be adaptive to changing environment. As an improvement, an online updated dynamic Gaussian mixture model-based controller was proposed in [9] to drive a catheter tip to desired positions. However, the number of Gaussian components needs manually selected and the control action remains to be optimized. On the other hand, we utilized random sampling control methods in this work. Researchers in [10] also presented a sampling-based optimal control method in which Gaussian process regression was used to learn model dynamics. However, the method was only validated in the simulation world. In [11] a neural network-based dynamics model with a random sampling approach was proposed to control legged millirobots. However, this approach lacked online adaptation to new tasks and unexpected environmental perturbations.



(a) (b)

Fig. 1. Bending behavior of a soft pneumatic actuator under different air pressure inputs. (a) FEM-simulated bending. FEM: finite element method. (b) Experimental bending behavior.

In this work, we focus on controlling the bending movements of soft actuators. Figure 1 shows FEM-simulated and experimental bending behaviors of a soft pneumatic actuator under different air pressure inputs. The probabilistic dynamics model is implemented as a Gaussian process and updated online by real-time sensory data. Based on this model, optimal control actions are derived by minimizing a cost function, which drive soft actuators to achieve desired positions. The effectiveness of our proposed control algorithm is evaluated on various soft pneumatic actuators. An overview of our proposed closed-loop control framework is shown in Fig.2. Some similarities exist between our method and a PILCO (Probabilistic Inference for Learning COntrol) algorithm proposed in [12], such as both adopting an online updated Gaussian process model and a saturating cost function. However, differences are obvious: (1) PILCO requires iterative probabilistic planning and updating policy parameters to learn a policy which is computationally expen-

This work was supported by the CUHK T Stone Robotics Institute (C4930759), the Knowledge Transfer Project Fund of the Chinese University of Hong Kong (KPF18HLF12), and the Innovation and Technology Fund of Hong Kong (ITS/065/18FP).

[1]Zhi Qiang Tang, Ho Lam Heung and Kai Yu Tong are with the Department of Biomedical Engineering, The Chinese University of Hong Kong, Shatin, Hong Kong SAR.tangzhiqiang@link.cuhk.edu.hk, 1155033948@link.cuhk.edu.hk, kytong@cuhk.edu.hk

[2]Zheng Li is with Department of Surgery and the Chow Yuk Ho Technology Centre for Innovative Medicine, The Chinese University of Hong Kong, Shatin, Hong Kong SAR.lizheng@cuhk.edu.hk

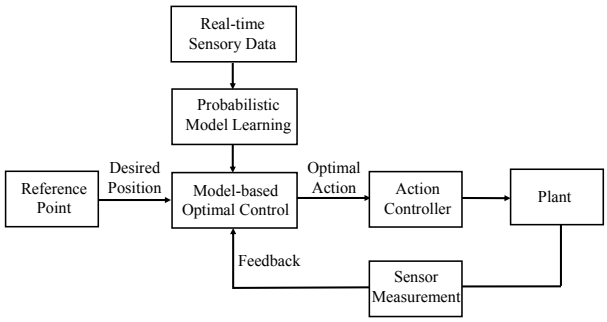


Fig. 2. Diagram of the proposed closed-loop control framework.

sive with the increase of dataset. Our approach takes random sampling methods to achieve a similar level of performance with low computation time, which is more practicable in real-time applications; (2) Current applications of the PILCO algorithm are limited in rigid robots while our approach is designed for soft robots. The intrinsic dynamics of soft robots are significantly different from that of rigid robots. Detailed comparisons between our approach and PILCO will be presented in Section VI.

The main contributions of this paper are as follows:

- A design-independent control approach which is evaluated rigorously for various soft pneumatic actuators performing reference point tracking tasks;
- A probabilistic modeling method for soft robots which requires no prior knowledge of the system dynamics;
- An optimal control policy with online learning to perform new tasks and deal with unexpected environmental situations.

The rationale behind our approach is that probabilistic models can capture model uncertainty and inherent nonlinearity regardless of specific system structures. Generally, model-based methods are more promising to efficiently extract valuable information from available data than model-free methods [13]. However, deterministic model-based methods would suffer severely from model errors because the model is assumed to resemble the real system accurately. The Gaussian process infers a distribution over all plausible models given the observed data instead of one (possibly erroneous) model and thus can make the controller more robust to modeling errors.

The rest of this paper is organized as follows. Section II describes the details of model learning. Section III introduces the optimal control policy based on the derived model. Section IV conducts algorithm simulation with parameter analysis. Section V present physical experimental results. Comparisons are discussed in Section VI. Section VII concludes the article.

II. Model Learning

Our probabilistic model is represented as a Gaussian process. Compared to parametric models which are problem specific and need manually specified, Gaussian process regression is a nonparametric regression method which is promising to automatically extract latent dynamics from data and can be served as a learning model for a broad range of tasks [12], [14]. In this paper, we formulate the system dynamics

$$\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_t) + \boldsymbol{w} \tag{1}$$

with states $\boldsymbol{x} \in \mathbb{R}^S$, control actions $\boldsymbol{u} \in \mathbb{R}^A$, system noise $\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{0},\ \boldsymbol{\Sigma}_w)$, and unknown transition dynamics f. Also we assume that the variables are independent and identically distributed (i.i.d). We use tuples $\tilde{\boldsymbol{x}}_t = [\boldsymbol{x}_t^T\ \boldsymbol{u}_t^T] \in \mathbb{R}^{S+A}$ as training inputs and state differences $\boldsymbol{\Delta}_t = \boldsymbol{x}_{t+1} - \boldsymbol{x}_t \in \mathbb{R}^S$ as training targets. The Gaussian process f can be completely determined by its mean function $m(\tilde{\boldsymbol{x}})$ and covariance function $k(\tilde{\boldsymbol{x}},\ \tilde{\boldsymbol{x}}')$. In this paper, we take the mean function to be zero ($m(\tilde{\boldsymbol{x}}) \equiv 0$) and a popular choice of covariance function:

$$k(\tilde{\boldsymbol{x}},\ \tilde{\boldsymbol{x}}') = \sigma_f^2 \exp\big(-\frac{1}{2}(\tilde{\boldsymbol{x}}-\tilde{\boldsymbol{x}}')^T\boldsymbol{\Lambda}^{-1}(\tilde{\boldsymbol{x}}-\tilde{\boldsymbol{x}}')\big) + \sigma_w^2\delta(\tilde{\boldsymbol{x}},\ \tilde{\boldsymbol{x}}') \tag{2}$$

where σ_f^2 is the signal variance, σ_w^2 is the noise variance, $\boldsymbol{\Lambda} = diag([l_1^2, \cdots, l_{S+A}^2])$ with length-scales l_i and $\delta(\tilde{\boldsymbol{x}},\ \tilde{\boldsymbol{x}}')$ is the Kronecker delta function. Given n training inputs $\tilde{\boldsymbol{X}} = [\tilde{\boldsymbol{x}}_1, \cdots, \tilde{\boldsymbol{x}}_n]$ and corresponding training targets $\boldsymbol{y} = [\boldsymbol{\Delta}_1, \cdots, \boldsymbol{\Delta}_n]^T$, the Gaussian process hyper-parameters ($\boldsymbol{\theta} = \{\sigma_f^2,\ \sigma_w^2,\ l_i\}$) are learned by evidence maximization [14], which maximizes $\log p(\boldsymbol{y}|\tilde{\boldsymbol{X}},\ \boldsymbol{\theta})$

$$\log p(\boldsymbol{y}|\tilde{\boldsymbol{X}},\ \boldsymbol{\theta}) = -\frac{1}{2}\boldsymbol{y}^T K^{-1}\boldsymbol{y} - \frac{1}{2}\log|K| - \frac{n}{2}\log 2\pi \tag{3}$$

where K is the kernel matrix with entries $K_{ij} = k(\tilde{\boldsymbol{x}}_i,\ \tilde{\boldsymbol{x}}_j)$. Then the predicted state $\boldsymbol{x}_{t+1}$ is Gaussian distributed given by

$$\begin{aligned} p(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t,\ \boldsymbol{u}_t) &= \mathcal{N}(\boldsymbol{x}_{t+1}|\boldsymbol{\mu}_{t+1},\ \boldsymbol{\Sigma}_{t+1}) \\ \boldsymbol{\mu}_{t+1} &= \boldsymbol{x}_t + \mathbb{E}_f[\boldsymbol{\Delta}_t], \quad \boldsymbol{\Sigma}_{t+1} = \boldsymbol{\Sigma}_f[\boldsymbol{\Delta}_t] \end{aligned} \tag{4}$$

where the mean and variance of the Gaussian process prediction are

$$\begin{aligned} \mathbb{E}_f[\boldsymbol{\Delta}_t] &= k_*^T(K + \sigma_w^2\boldsymbol{I})^{-1}\boldsymbol{y} \\ \boldsymbol{\Sigma}_f[\boldsymbol{\Delta}_t] &= k_{**} - k_*^T(K + \sigma_w^2\boldsymbol{I})^{-1}k_* \end{aligned} \tag{5}$$

respectively, with $k_* = k(\tilde{\boldsymbol{X}},\ \tilde{\boldsymbol{x}}_t)$, $k_{**} = k(\tilde{\boldsymbol{x}}_t,\ \tilde{\boldsymbol{x}}_t)$.

The Gaussian process is continuously updated online by incorporating real-time sensory data. All hyper-parameters of the Gaussian process will be re-learned when new data is recorded. The online learning helps keep model accuracy.

III. Optimal Control

Based on the learned probabilistic model in Section II, we acquire optimal control actions $\boldsymbol{u}_{opt}$ by minimizing a cost function $C(\boldsymbol{x})$. The cost function penalizes states which are away from target positions. A saturating cost function is better than a quadratic cost function in terms of dealing with state uncertainty and system noise [15]. Hence, we choose the saturating cost function here which is given as follows:

$$C(\boldsymbol{x}) = 1 - \exp\big(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{r})^T\boldsymbol{T}^{-1}(\boldsymbol{x}-\boldsymbol{r})\big) \tag{6}$$

where $\boldsymbol{r}$ is the reference point and $\boldsymbol{T}^{-1}$ is a diagonal matrix with entries either unity or zero, scaled by $1/\sigma_c^2$. The parameter σ_c controls the width of the cost function. The expected value of the cost function (6) is:

$$\begin{aligned}\mathbb{E}_{\boldsymbol{x}}[C(\boldsymbol{x})] &= \int C(\boldsymbol{x})p(\boldsymbol{x})d\boldsymbol{x} \\ &= 1 - \int \exp\big(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{r})^T\boldsymbol{T}^{-1}(\boldsymbol{x}-\boldsymbol{r})\big)p(\boldsymbol{x})d\boldsymbol{x}\end{aligned} \tag{7}$$

From equation (4) we know that $\boldsymbol{x}_{t+1} \sim \mathcal{N}(\boldsymbol{x}_t + \mathbb{E}_f[\boldsymbol{\Delta}_t],\ \boldsymbol{\Sigma}_f[\boldsymbol{\Delta}_t])$. Thus, the expected cost value at state $\boldsymbol{x}_{t+1}$ is:

$$\begin{aligned}\mathbb{E}_{\boldsymbol{x}_{t+1}}[C(\boldsymbol{x}_{t+1})] =& 1 - |\boldsymbol{H}|^{-\frac{1}{2}} \times \exp\big(-\frac{1}{2}\boldsymbol{Z}^T\boldsymbol{S}\ \boldsymbol{Z}\big) \\ \boldsymbol{H} &= \boldsymbol{I} + \boldsymbol{\Sigma}_f[\boldsymbol{\Delta}_t]\ \boldsymbol{T}^{-1} \\ \boldsymbol{Z} &= \boldsymbol{x}_t + \mathbb{E}_f[\boldsymbol{\Delta}_t] - \boldsymbol{r} \\ \boldsymbol{S} &= (\boldsymbol{H}\ \boldsymbol{T})^{-1}\end{aligned} \tag{8}$$

Different control actions would affect the expected cost value. The optimal control action is the one that minimizes the expected cost value:

$$\begin{aligned}\boldsymbol{u}_{opt} &= \arg\min_{\boldsymbol{u}_t} \mathbb{E}_{\boldsymbol{x}_{t+1}}[C(\boldsymbol{x}_{t+1})] \\ \text{s.t.}\ \ \boldsymbol{x}_{t+1} &= f(\boldsymbol{x}_t, \boldsymbol{u}_t) + \boldsymbol{w} \\ \boldsymbol{u}_{lb} &\le \boldsymbol{u}_t \le \boldsymbol{u}_{ub} \\ \boldsymbol{x}_{lb} &\le \boldsymbol{x}_t \le \boldsymbol{x}_{ub}.\end{aligned} \tag{9}$$

where $\boldsymbol{u}_{lb}$ and $\boldsymbol{u}_{ub}$ are the lower and upper bounds of control actions, respectively. Accordingly, $\boldsymbol{x}_{lb}$ and $\boldsymbol{x}_{ub}$ are the lower and upper bounds of states. These boundary conditions are set to consider system safety issues.

Obtaining the exact optimal solution of equation (9) is difficult as the expected cost function and transition dynamics are nonlinear. Although analytical solutions are hard to achieve, there exist many numerical approximation techniques that are sufficient for succeeding at the desired solution. Here we adopt a simple random-sampling shooting method [11], [16] in which M candidate actions are randomly generated and the corresponding states are predicted using the learned dynamics model. Then the expected cost for all candidate actions are calculated. Finally, the candidate action with the minimum expected cost value is chosen as the optimal control action.

Furthermore, to improve control performance, we incorporate the idea of iterative learning control [17] which learns from past experience. Thus, in our control framework we purposely design an additional virtual control action $\boldsymbol{v}_t$ to implement such an ability of learning from previous executions. The virtual action is given as follows:

$$\boldsymbol{v}_{t+1} = \boldsymbol{v}_t + \lambda(\boldsymbol{r} - \boldsymbol{x}_t) \tag{10}$$

where λ is the error learning gain which adjusts amplitude of the virtual action. Then the final control action $\boldsymbol{u}_t$ is an integration of the optimal action $\boldsymbol{u}_{opt}$ in equation (9) and the virtual action $\boldsymbol{v}_t$, given as follows:

$$\boldsymbol{u}_t = \boldsymbol{u}_{opt} + \boldsymbol{v}_t \tag{11}$$

Algorithm 1 PMOLOC

1: **initialize:**Apply N random control actions and record data. Set virtual action $\boldsymbol{v}_0 = \boldsymbol{0}$
2: **repeat**
3: Learn probabilistic model f using all data.
4: Generate reference point $\boldsymbol{r}$.
5: Approximate optimal actions in Eq. (9).
6: Compute virtual actions in Eq. (10).
7: Apply final control actions and record data
8: **until** Desired performance achieved

The final control action $\boldsymbol{u}_t$ can be seen as a combination of feedback and feedforward controllers. Previous research [18] demonstrated that such combination could achieve better reference tracking performance than single feedback or feedforward controllers.

IV. Algorithm Simulation

After introducing the key ideas of our proposed control algorithm, we summarize the probabilistic model-based online learning optimal control (PMOLOC) framework in Algorithm 1. The simulation test and parameter analysis are described in the following subsections.

A. Simulation Test

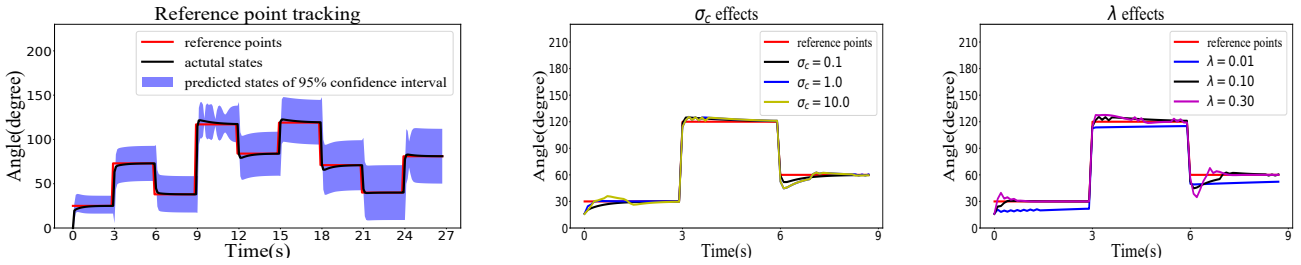
The simulation was built on Python 3.7 platform. Two datasets (D_1 and D_2) were collected from the real world system (seen in Fig. 4a), which contained 1200 and 120 sampling points respectively. The dataset D_1 was used for constructing the true model in simulation environment while D_2 was to initialize the probabilistic model. Step trajectories were utilized to test tracking performance because the time response and overshoot to a set-point change could be observed. σ_c and λ were determined by the following parameter analysis results. Boundary values of u_{lb}, u_{ub}, x_{ub} and x_{ub} were set based on safety issues of the real world system. The reference tracking error is defined as the root mean square error between the reference points and the actual states, given as follows:

$$e_i = \sqrt{\frac{1}{N_i}\|\boldsymbol{r}_i - \boldsymbol{x}\|_2^2} \tag{12}$$

where i is the iteration number and N_i is the data length of step command $\boldsymbol{r}_i$. Here, one step command change is treated as one iteration. The desired tracking performance is achieved when the relative tracking error $e_i/|\boldsymbol{r}_i|_\infty$ is lower than a required tracking accuracy ϵ. All the control parameters for the simulation are listed in Table I. The simulation results are shown in Fig. 3. From Fig. 3a we can see that step trajectories could be well tracked by our proposed control algorithm. At the final step command (24s$\sim$27s) when desired tracking performance was achieved, the settling time was 1 second and overshoot was 2.4%.

B. Parameter Analysis

Here we discuss how σ_c and λ affect the performance of our control algorithm.



(a) (b) (c)

Fig. 3. Algorithm simulation results. (a) Tracking process of step reference commands. (b) and (c) Effects of different σ_c and λ values on reference tracking performance, respectively.

TABLE I: SIMULATION CONTROL PARAMETERS.

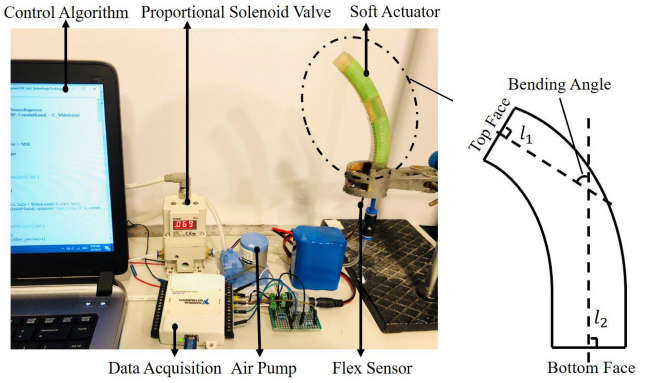
$\boldsymbol{r}$	σ_c	λ	u_{lb}	u_{ub}	x_{lb}	x_{ub}	M	N	ϵ
Step Commands	0.1	0.1	0 kPa	120 kPa	0°	130°	120	120	1%

1) σ_c effects: σ_c controls the width of the cost function. We try different values of σ_c to see its effects on tracking performance, as shown in Fig. 3b. We can observe that larger values of σ_c lead to larger overshoot and oscillations while smaller values tend to be more stable. The value of σ_c is suggested to be set in a way that the cost function can easily distinguish between a "good" state (close to reference point) and a "bad" state (away from reference point). Based on the results of Fig. 3b, we chose $\sigma_c = 0.1$ in our simulation.

2) λ effects: Parameter λ affects the magnitude of virtual actions. Different λ values are applied to test its performance. From Fig. 3c we can observe that actual states have no overshoot but cannot achieve reference points at $\lambda = 0.01$. Meanwhile, fast response but obvious overshoot and oscillation occur at $\lambda = 0.3$. Equation 10 indicates that smaller λ value may not utilize previous error efficiently while larger λ may exaggerate error effects. An appropriate λ value should be selected to balance tracking speed and stability. As the results in Fig. 3c suggest, $\lambda = 0.1$ was selected in our simulation.

V. Experiment

In this section, real-world experiments were conducted to demonstrate the feasibility of our proposed control algorithm. The experimental setup is shown in Fig. 4a. The control algorithm was implemented on a personal computer (64-bit operating system, i5-5200 CPU, 2.2GHz, 2 cores). The data acquisition device (DAQ) (USB-6009, National Instruments, Corp., USA) transmitted control commands to a proportional solenoid valve (ITV2050-212L, SMC, Tokyo, Japan) and received sensor readings from the solenoid valve and the flex sensor (Spectra Symbol, USA). The air pump (BTC Diaphragm Pump, Parker Hannifin Corporation, OH, USA) supplied air pressure for soft actuators. The proportional solenoid valve could regulate the air pressure based on command signals from the DAQ. The flex sensor measured the bending angle of soft actuator. Both controller frequency and data sampling rate were 10 Hz. Furthermore, the bending



(a) (b)

Fig. 4. Experimental setup. (a) The control algorithm is implemented on a personal computer. The air pump supplies air pressure for the soft actuator and the proportional solenoid valve regulates air pressure. The flex sensor measures bending angles. The data acquisition collects feedback pressure and bending angle information. (b) The bending angle definition of soft actuators.

angle of a soft actuator is defined as the angle between lines l_1 and l_2 , as shown in Fig. 4b. The line l_1 and line l_2 are perpendicular to the top face and bottom face, respectively. Additional media and information can be found in the supplementary video.

A. Test on Various Soft Pneumatic Actuators

Three types of soft pneumatic actuators were utilized to test the control performance of our proposed algorithm. All of tested soft actuators were directly 3D-printed from silicones (ACEO, Burghausen, Germany). Design characteristics and experimental control parameters of the three soft actuators are listed in Table II. The parameters u_{lb}, u_{ub} , x_{lb} and x_{ub} were determined by the safe operation range of each soft actuator. σ_c and λ were adjusted based on the parameter analysis in section IV. As the resolution of the proportional solenoid valve was 1kPa, M was determined to cover all feasible pressure values within the safe pressure range for each soft actuator. The effects of different M values on control performance would be discussed in section VI. N

TABLE II: DESIGN CHARACTERISTICS AND EXPERIMENTAL CONTROL PARAMETERS OF SOFT ACTUATORS.

Soft Actuators	Material	Shape	Structure	Dimension(mm)	σ_c	λ	u_{lb}	u_{ub}	x_{lb}	x_{ub}	M	N	ϵ
A	Shore A 30	Semi-cylindrical	Fiber+Limiting Layer	ϕ18×120	0.01	0.05	0 kPa	120 kPa	0°	130°	120	120	5%
B	Shore A 60	Bellow-wave	With Limiting Layer	79×18×17	0.01	0.06	0 kPa	120 kPa	0°	160°	120	120	5%
C	Shore A 20	Bellow-wave	Purely Soft Material	69×11×10	0.01	0.01	0 kPa	90 kPa	0°	160°	90	90	5%

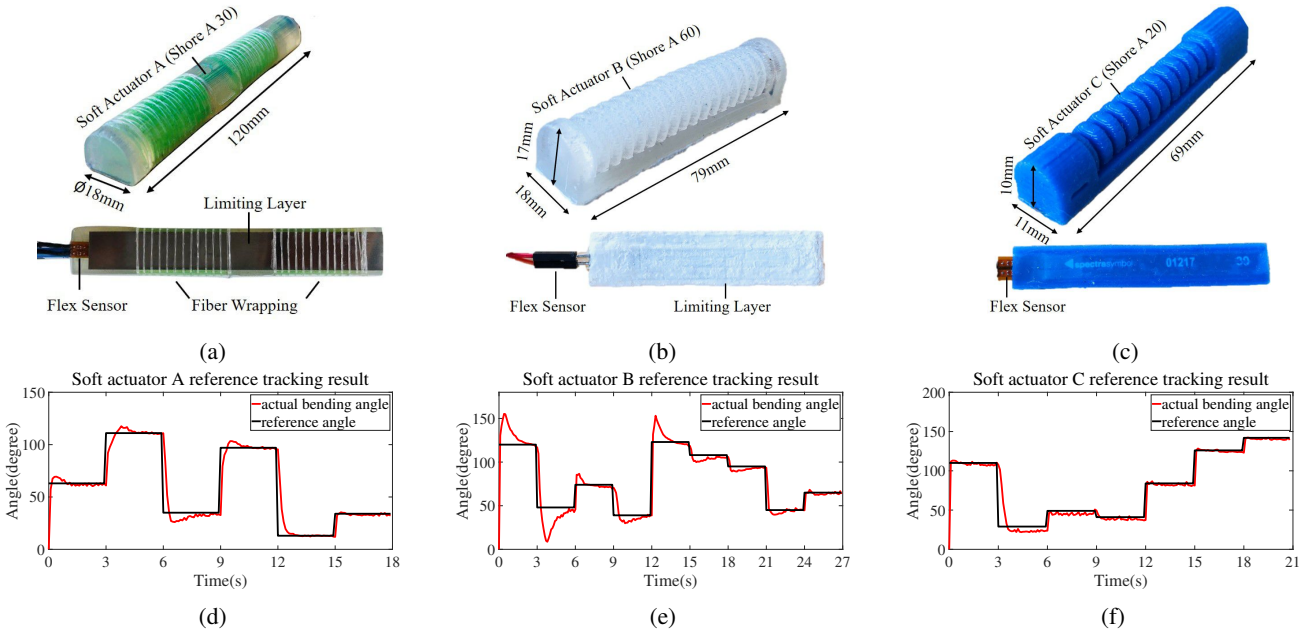


Fig. 5. Physical demonstrations and experimental results of tested soft pneumatic actuators. (a), (b) and (c): Basic descriptions about the design of soft actuator A, B and C, respectively; (d), (e) and (f): Experimental reference tracking results of soft actuator A, B and C, respectively.

was chosen in a similar way as M for the model initialization and ϵ was set to terminate the algorithm.

The selected silicone material of soft actuator A could offer sufficient elongation (450%) and tensile strength (6 MPa) with hardness of Shore A 30. Soft actuator A was designed as the semi-cylindrical shape which was also used in previous research articles [19], [20], as shown in Fig.5a. An A2 stainless steel plate of thickness 0.1 mm was put on the bottom of the soft actuator body as the strain limiting layer and transparent fiber reinforcements were wrapped around the soft actuator body. A flex sensor was positioned between the soft actuator body and steel plate. The overall dimension of soft actuator A was 120mm length with 18mm diameter and its mass was 24 grams. During the experiments, reference points were generated with a random amplitude ranging from 0° to 130°, changing the command at time increments of 3 seconds. The reference tracking result of soft actuator A is shown in Fig. 5d. From Fig. 5d we could observe that the tracking performance improved iteratively. At the final step command (15s∼18s) when desired tracking performance was achieved, the rising time was 0.1 seconds and the settling time was 1.5 seconds and overshoot was 4.5%.

The selected silicone material of soft actuator B could offer 200% elongation with 7MPa tensile strength and hardness of Shore A 60. Soft actuator B was designed as a bellow-wave shape which was similar to PneuNets soft actuators [21], [22], as shown in Fig. 5b. An A2 stainless steel plate of thickness 0.1 mm was put on the bottom of soft actuator body but without fiber reinforcements. A flex sensor was positioned between the soft actuator body and the steel plate. The overall dimension of soft actuator B was 79mm×18mm×17mm (length×width×height) and its mass was 21 grams. Reference points were step commands with a random amplitude ranging from 0° to 160°, changing the command at time increments of 3 seconds. Figure 5e shows the reference tracking result. Obvious overshoots occurred in the early stage (0∼15s). When the desired tracking performance was achieved at time 24s∼27s, the rising time was 0.1 seconds and the settling time was 2 seconds and overshoot was 5.0%.

The selected silicone material of soft actuator C could offer 800% elongation with 7MPa tensile strength and hardness of Shore A 20. It was also designed as a bellow-wave shape but without fiber reinforcements and strain limiting layer, as shown in Fig. 5c. A flex sensor was positioned on the bottom of the soft actuator body. The overall dimension of soft actuator C was 69mm×11mm×10mm and its mass was 6 grams. Reference points were step commands with a random amplitude ranging from 0° to 160°, changing the

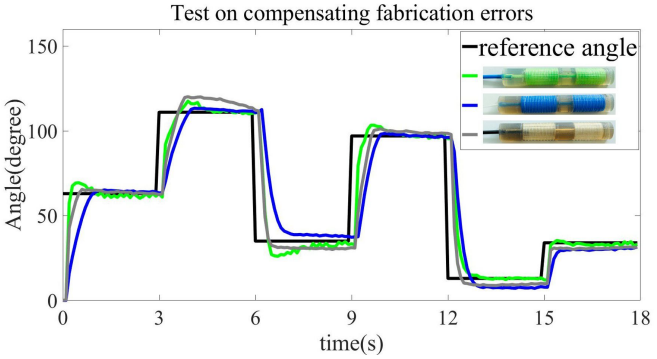


Fig. 6. Online learning compensates fabrication errors of the same design without parameter tuning.

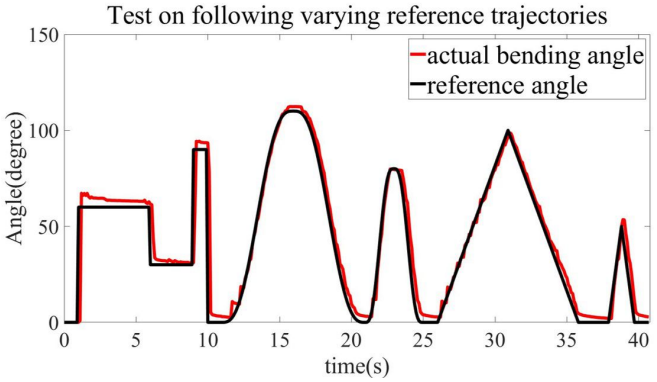


Fig. 7. The proposed algorithm can control the soft actuator to track varying reference trajectories.

command at time increments of 3 seconds. The reference tracking result is shown in Fig.5f from which we can see that the tracking performance improves gradually. When the desired tracking performance was achieved at time 18s∼21s, the rising time was 0.1 seconds and the settling time was 1.5 seconds and overshoot was 0.2%.

B. Test Under Different Situations

The following experiments tested the tracking performance of the proposed approach in different situations.

Soft actuators with the same design may have different bending angles with the same pressure inputs due to fabrication errors. To test whether our proposed method could be adaptive to such fabrication errors, we experimented with multiple soft actuators of the same design. During the experimentation, three soft actuators of design A were controlled to follow the same reference trajectory. Control parameters were the same as the previous soft actuator A test. The experimental results are shown in Fig.6. The results indicated that all three soft actuators could gradually achieve satisfactory tracking performance as online learning progressed. Thus, the online updated dynamic model can automatically compensate fabrication errors of soft actuators without parameter tuning.

Besides step trajectories, more time-variant reference trajectories can be tested. Fig.7 showed a varying reference

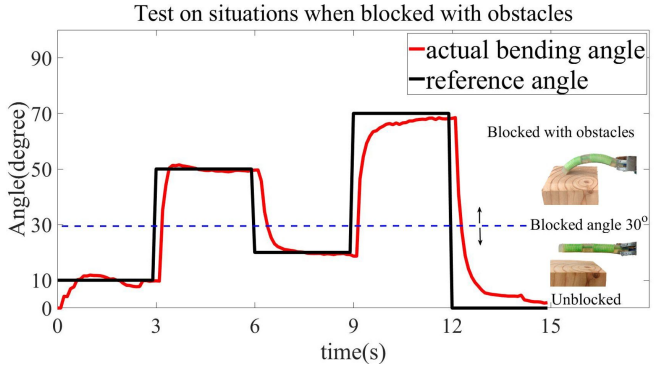


Fig. 8. Reference points can still be achieved even though the soft actuator is blocked by environmental constraints.

trajectory which was designed as a combination of complex step trajectories (step length and amplitude changed), varying sine trajectories (amplitude and frequency changed) and varying ramp trajectories (steepness and amplitude changed). The soft actuator C was controlled to track the varying reference trajectory by the same control parameters as the previous soft actuator C test. The results in Fig.7 showed that our proposed method could control the soft actuator to track varying reference trajectories with satisfactory performance.

All the previous tests were free bending situations. We further conducted an experiment in constrained environment. During the experimentation, the soft actuator A was placed horizontally above a fixed woodblock. When the bending angle of the soft actuator reached 30°, the soft actuator would be blocked by the woodblock, as shown in Fig.8. The experimental results in Fig.8 demonstrated that the soft actuator could still reach the reference points (50°and 70°) and keep tracking performance even though it was blocked by the woodblock. Thus, the proposed algorithm can deal with unexpected environmental situations.

VI. Discussion

We further compare the proposed approach with other model-based controllers. Limitations and future work are also discussed in this section.

A. Comparison with Other Methods

The PILCO algorithm is one of the state-of-the-art algorithms for data-efficient model-based policy search. We conducted a comparison experiment to show the performance between our method (i.e. PMOLOC) and PILCO. We utilized PILCO open-source code [23] to execute the PILCO algorithm. The PILCO controlled the soft actuator A to follow the same reference angle as in Fig.5d. Main parameters of PILCO were as follows: the control period was 0.1s and the prediction horizon was 3s; the initial dataset consisted of 120 sampling points; the cost width was 0.01; numbers of basis functions and controller optimizations were 100 and 6, respectively.

The comparison results were shown in Fig.9a, which demonstrated that our proposed method achieved better

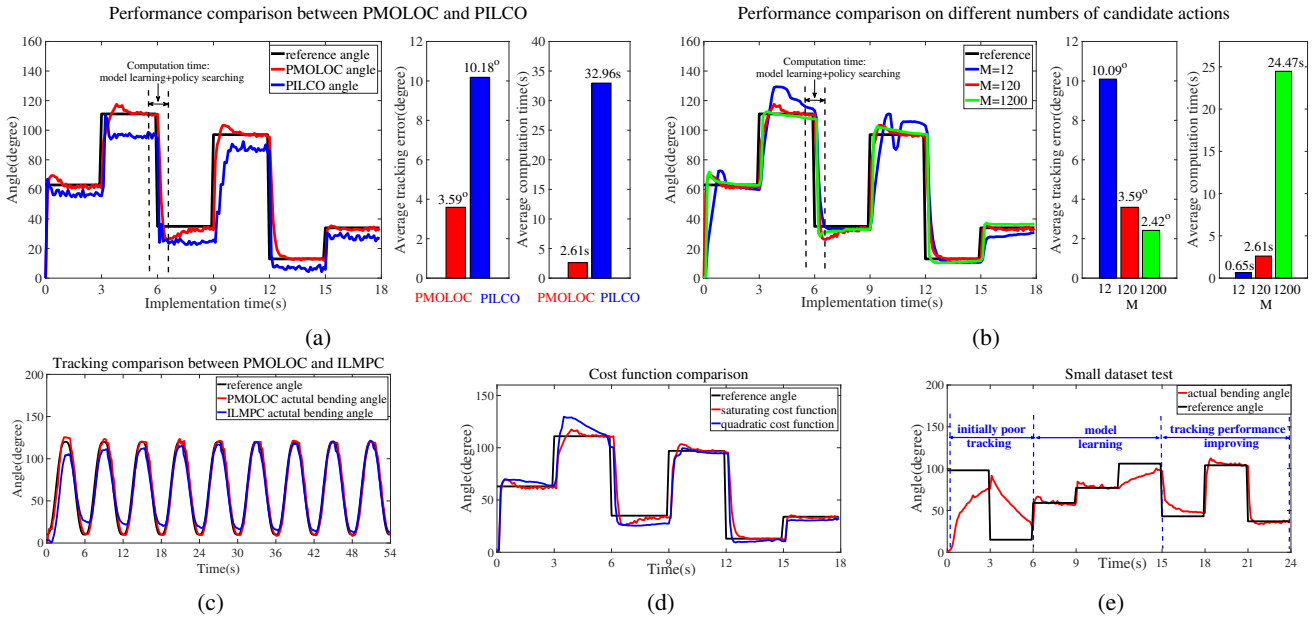


Fig. 9. Algorithm comparison results. (a) Control performance comparison between PMOLOC and PILCO. (b) Performance comparison among different numbers of candidate actions (c) Tracking comparison between PMOLOC and ILMPC. (d) Tracking comparison of different cost functions. (e) Online learning ability to perform new tasks.

tracking performance than PILCO in terms of less tracking error (PMOLOC: 3.59°error; PILCO: 10.18°error) and reduced computation time (PMOLOC: 2.61 seconds ; PILCO: 32.96 seconds). The average tracking error was an average value of all iteration errors defined in equation (12). Total running time included model learning time, policy searching time and policy implementation time. Policy implementation was executed in real time. Model learning and policy searching were performed during each iteration interval. The computation time here was the CPU running time of model learning and policy searching.

The PILCO required a longer computation time because it relied on computationally expensive analytical methods to do approximate inference for each step of the policy evaluation. While the analytic approach in PILCO is efficient on a sequential computer, it cannot take advantage of the multi-core architectures now present in most computers [24]. By contrast, random sampling approaches could be computed parallelly by multi-core and multi-threads architectures on the computer. Moreover, the computational complexity of PILCO during policy searching was at least $\mathcal{O}(N^2)$ [12] while ours was $\mathcal{O}(M)$. The computational complexity of PILCO was obviously larger than PMOLOC. Furthermore, PILCO might be stuck into local optima [25] due to the highly nonlinear characteristics of soft material. Our approach could find global optima, because the optimal action in our approach was chosen among an action space that covered all feasible pressure values.

The large computation time required by PILCO restricted its real-time applications. By contrast, the proposed PMOLOC approach required sufficiently less computation time while keeping a similar level of performance, which had advantages over PILCO in our soft robotic applications.

In addition, we compared the PMOLOC with our previous work [26], in which an iterative learning model predictive control (ILMPC) method was proposed for soft bending actuators. Main parameters of ILMPC were as follows: learning step size was 0.1 and feed-forward input gain was 10. The comparison result was shown in Fig. 9c, from which we could see that ILMPC required more iterations to achieve a similar level of tracking performance than PMOLOC.

B. Self Comparison

Different numbers of candidate actions in our method will affect tracking performance and computation time. The soft actuator A was controlled to follow the reference angle in Fig.5d by different numbers of candidate actions. The results in Fig.9b indicated that larger M would help reduce tracking error ($M = 12$: 10.09°error; $M = 120$: 3.59°error; $M = 1200$: 2.42°error) but require more computation time ($M = 12$: 0.65 seconds; $M = 120$: 2.61 seconds; $M = 1200$: 24.47 seconds). 120 candidate actions could achieve satisfactory tracking performance while keeping low computation time in our experiments. Thus, M was 120 during the simulation and physical experiments for the soft actuator A.

Moreover, the choice of cost function will also affect control performance. We compared the tracking performance between a widely used quadratic cost function [10] and the saturating cost function. The soft actuator A was also used to conduct this comparison experiment. The comparison results were shown in Fig.9d, from which we could see that the saturating cost function achieved better tracking performance than the quadratic cost function. Thus, the saturating cost function was adopted in our method.

Furthermore, since Gaussian process regression is a machine learning method, its effectiveness is potentially restricted to the training dataset. To test the online learning ability of our proposed method, a small range dataset was used to initialize the model for soft actuator A, which contained 10 sampling points with angle range $[0^\circ, 60^\circ]$. As demonstrated in Fig.9e, the controller initially showed a poor tracking performance at starting step commands (0$\sim$6s) due to the inaccurate learned model from small offline dataset. After several iterations (6s$\sim$15s), the model was updated online by incorporating new data. Then the tracking performance was significantly improved in the last iterations (15s$\sim$24s). This result indicates that our proposed algorithm can perform new tasks with online learning.

C. Limitations and Future Work

In this work, we performed one-step prediction of the GP-based dynamics model. Since the soft material exhibited compliance and hysteresis characteristics [7] and reference trajectories were continuous, the states of soft actuators would not change abruptly. Thus, one-step prediction works in our present applications. To generalize the current method, multi-step predictions will be investigated in the future work. Random sampling may be insufficient for long-term predictions and other methods [10], [12] will be explored. Moreover, input uncertainty will be considered in our future work to improve control performance.

VII. Conclusion

In this work a probabilistic model-based online learning optimal control (PMOLOC) algorithm was proposed for various soft pneumatic actuators. A Gaussian process model was employed to describe the system dynamics and updated online to keep model accuracy. Then an optimal control policy was derived based on the probabilistic model which minimized a saturating cost function. Through real robot experiments involving three different designs of soft pneumatic actuators under different situations, we demonstrated the effectiveness of our proposed method. Besides, comparisons between our method with the PILCO algorithm showed advantages of the proposed approach in real-time soft robotic applications. We also showed the online learning ability to perform new tasks starting from a small range dataset. In a nutshell, this work provides a promising design-independent control approach for the soft robotics community.

References

[1] A. Villoslada, C. Rivera, N. Escudero, F. Martín, D. Blanco, and L. Moreno, "Hand exo-muscular system for assisting astronauts during extravehicular activities," *Soft robotics*, vol. 6, no. 1, pp. 21–37, 2019.

[2] K. C. Galloway, K. P. Becker, B. Phillips, J. Kirby, S. Licht, D. Tchernov, R. J. Wood, and D. F. Gruber, "Soft robotic grippers for biological sampling on deep reefs," *Soft robotics*, vol. 3, no. 1, pp. 23–33, 2016.

[3] P. Polygerinos, Z. Wang, K. C. Galloway, R. J. Wood, and C. J. Walsh, "Soft robotic glove for combined assistance and at-home rehabilitation," *Robotics and Autonomous Systems*, vol. 73, pp. 135–143, 2015.

[4] Z. Q. Tang, H. L. Heung, K. Y. Tong, and Z. Li, "Model-based online learning and adaptive control for a "human-wearable soft robot" integrated system," *The International Journal of Robotics Research*, 2019.

[5] M. Runciman, A. Darzi, and G. P. Mylonas, "Soft robotics in minimally invasive surgery," *Soft robotics*, 2019.

[6] H. L. Heung, P. W. Chiu, and Z. Li, "Design and prototyping of a soft earthworm-like robot targeted for gi tract inspection," in *2016 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2016, pp. 497–502.

[7] T. George Thuruthel, Y. Ansari, E. Falotico, and C. Laschi, "Control strategies for soft robotic manipulators: A survey," *Soft robotics*, vol. 5, no. 2, pp. 149–163, 2018.

[8] K. Elgeneidy, N. Lohse, and M. Jackson, "Bending angle prediction and control of soft pneumatic actuators with embedded flex sensors–a data-driven approach," *Mechatronics*, vol. 50, pp. 234–247, 2018.

[9] B. Yu, J. d. G. Fernández, and T. Tan, "Probabilistic kinematic model of a robotic catheter for 3d position control," *Soft robotics*, vol. 6, no. 2, pp. 184–194, 2019.

[10] Y. P. Pan, K. Saigol, and E. A. Theodorou, "Belief space stochastic control under unknown dynamics," in *2017 American Control Conference (ACC)*. IEEE, 2017, pp. 3764–3770.

[11] A. Nagabandi, G. Yang, T. Asmar, R. Pandya, G. Kahn, S. Levine, and R. S. Fearing, "Learning image-conditioned dynamics models for control of underactuated legged millirobots," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 4606–4613.

[12] M. P. Deisenroth, D. Fox, and C. E. Rasmussen, "Gaussian processes for data-efficient learning in robotics and control," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 2, pp. 408–423, 2013.

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

[14] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*. MIT Press Cambridge, MA, 2006, vol. 2, no. 3.

[15] M. P. Deisenroth, *Efficient reinforcement learning using Gaussian processes*. KIT Scientific Publishing, 2010, vol. 9.

[16] A. V. Rao, "A survey of numerical methods for optimal control," *Advances in the Astronautical Sciences*, vol. 135, no. 1, pp. 497–528, 2009.

[17] D. A. Bristow, M. Tharayil, and A. G. Alleyne, "A survey of iterative learning control," *IEEE control systems magazine*, vol. 26, no. 3, pp. 96–114, 2006.

[18] N. Amann, D. H. Owens, and E. Rogers, "Iterative learning control using optimal feedback and feedforward actions," *International Journal of Control*, vol. 65, no. 2, pp. 277–293, 1996.

[19] K. C. Galloway, P. Polygerinos, C. J. Walsh, and R. J. Wood, "Mechanically programmable bend radius for fiber-reinforced soft actuators," in *2013 16th International Conference on Advanced Robotics (ICAR)*. IEEE, 2013, pp. 1–6.

[20] H. L. Heung, K. Y. Tong, T. H. Lau, and Z. Li, "Robotic glove with soft-elastic composite actuators for assisting activities of daily living," *Soft robotics*, vol. 6, no. 2, pp. 289–304, 2019.

[21] B. Mosadegh, P. Polygerinos, C. Keplinger, S. Wennstedt, R. F. Shepherd, U. Gupta, J. Shim, K. Bertoldi, C. J. Walsh, and G. M. Whitesides, "Pneumatic networks for soft robotics that actuate rapidly," *Advanced functional materials*, vol. 24, no. 15, pp. 2163–2170, 2014.

[22] H. K. Yap, J. H. Lim, F. Nasrallah, J. C. Goh, and R. C. Yeow, "A soft exoskeleton for hand assistive and rehabilitation application using pneumatic actuators with variable stiffness," in *2015 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2015, pp. 4967–4972.

[23] M. P. Deisenroth, A. McHutchon, J. Hall, and C. E. Rasmussen, "Pilco web site," 2013. [Online]. Available: http://mlg.eng.cam.ac.uk/pilco/

[24] K. Chatzilygeroudis, R. Rama, R. Kaushik, D. Goepp, V. Vassiliades, and J.-B. Mouret, "Black-box data-efficient policy search for robotics," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 51–58.

[25] Y. Gal, R. McAllister, and C. E. Rasmussen, "Improving pilco with bayesian neural network dynamics models," in *Data-Efficient Machine Learning workshop, ICML*, vol. 4, 2016.

[26] Z. Q. Tang, H. L. Heung, K. Y. Tong, and Z. Li, "A novel iterative learning model predictive control method for soft bending actuators," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 4004–4010.