Compensating for Material Deformation in Foldable Robots via Deep Learning – A Case Study

Mohammad Sharifzadeh[1,*], Yuhao Jiang[2,*], Amir Salimi Lafmejani[3,*], Daniel M. Aukes[1]

***Abstract*— Foldable, origami-inspired, and laminate mechanisms are highly susceptible to deformation under external loading, which can lead to position or orientation errors if idealized kinematic models are used. According to dimensional scaling laws, laminate devices can often be treated as rigid bodies at millimeter and smaller scale deformations. However, foldable mechanisms enter the territory of soft robots at larger scales. In this paper, we show the effect of external loads applied to a laminate, 2-DOF parallel robot and the corresponding errors during a pointing task. We then present two control methods, based on deep learning, that compensates for errors caused by the material deformation in foldable robots. For each proposed control method, a Deep Neural Network (DeepNN) is trained to learn the end-effector's deformation model in no-load and loaded conditions. A DeepNN called an updating network is trained and applied in real-time using measured sensor data, in order to transfer updated weights into another DeepNN called the target network. The target network generates control signals with the aim of compensating for the end-effector's error in tracking a desired trajectory. We evaluate our proposed control methods when applied to a laminate robotic end-effector under different external loading conditions in tracking spiral paths. The experimental results show the effectiveness of our proposed control methods in compensating for material deformation in foldable robots.**

I. Introduction

Laminate fabrication techniques provide an affordable and rapid alternative to traditional rigid robot prototyping, and have been applied in a variety of micro- and millimeter-scale robotic mechanisms. However, the use of long and slender beams within laminate systems can produce large deformation in the robot's end-effector even when stiffer materials are used in fabrication. This large deformation of the end-effector causes significant errors in accomplishment of tasks such as position stabilization and path tracking. To address this issue, parallel mechanisms are often used in laminate designs in order to achieve higher stiffness of the robot. Unlike serial robots, parallel robots allow the actuators to be mounted proximal to the end-effector, which in turn lowers the robot's structure load on its end-effector. Despite the improvements in stiffness, using traditional kinematic formulations that neglect link flexibility would be inaccurate to describe robot's motion especially when the robot works under heavy loads or when its links are fabricated in large scales. The same phenomenon is observed in the field of soft robotics, where computational solutions to control the position of robot's end-effector under load are based on dividing the material into finite number of small elements. Due to its high computational complexity, this approach is intractable in real-time implementations. This issue can be solved by creating reduced-order Finite Element Analysis (FEA) models. These models have a limited number of cases that are used within controllers, yet it is an expert tool, the order of model reduction affects the precision and relies on domain expertise in order to successfully implement and optimize them on each new design [1]–[4].

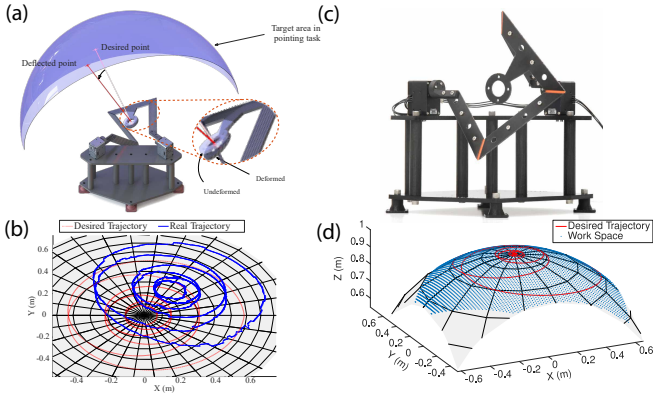

Fig. 1. **(a)** The effect of end-effector deviation caused by a load in a pointing task. **(b)** Illustration of the tracked path in tracking of a desired spiral trajectory caused by a load on the robot's end-effector. **(c)** The 2-DoF laminated foldable parallel robot. **(d)** The robot's workspace (blue dots) and the periodical desired spiral trajectory (red line) that is used in the rest of the paper.

Soft robots offer the possibility to extend the benefits of robotics to applications that have not previously been approachable with rigid robots [5]. However, further progress of soft robots will also increasingly depend on advancements in feedback control, machine intelligence, and computational modeling since conventional model-based control methods are not applicable to soft robots with infinite number of degrees-of-freedom [6]. Various control approaches for control of soft robots have been proposed and validated in the literature. One of the most precise approaches is to employ FEA methods. In [7], model order reduction of FEM using

[1]Mohammad Sharifzadeh and Daniel Aukes are with the Polytechnic School, Fulton Schools of Engineering, Arizona State University, Mesa, AZ, 85212, USA sharifzadeh@asu.edu; danaukes@asu.edu

[2]Yuhao Jiang is with School for Engineering of Matter, Transport and Energy, Fulton Schools of Engineering, Arizona State University, Tempe, AZ, 85281, USA yuhao92@asu.edu

[2]Amir Salimi Lafmejani is with School of Electrical, Computer and Energy Engineering, Fulton Schools of Engineering, Arizona State University, Tempe, AZ, 85281, USA asalimil@asu.edu

*These authors contributed equally to the paper.

(Corresponding author: Daniel M. Aukes)

This work is partially supported by the National Science Foundation Grant No. 1935324.

Supplementary video: https://youtu.be/32UFB4Ziq0I

snapshot proper orthogonal decomposition is proposed in order to achieve a reduced-order model with lower calculation effort. The control methods based on Piecewise Constant Curvatures (PCC) [8] for modeling configuration of soft robots are widely used in numerous studies. A linear time-varying Gaussian model is proposed in [9] for dynamic modeling. This model alongside a Linear Quadratic Regulator (LQR)-based Gaussian controller and a Kullback-Leibler divergence policy is used to stabilize the robot's end-effector to a desired position after several iterations while accounting for both dynamics control and path planning. The control of a soft robotic arm driven by a Shape-Memory Alloy (SMA) coil has been studied in [10] where they employ Proportional Integral Derivative (PID) controller when the curvature is measured by Hall sensors under assumption of constant curvature of each segment. In [11] dynamic control of a planar soft robotic arm interacting with the environment has been presented by the assumption of PCC and modeling each segment by a rigid limb. This method was extended to the control of 3D robots in [12] by modeling each segment as a rigid limb. In both studies the mass, stiffness, and damping coefficient of each link of the robot must be experimentally identified. Dynamics of a variable-length, multi-section, continuum robot has been introduced and experimentally validated in [13], [14]. A combination of PCC with FEA model-reduction policy is proposed in [15] in which the results show that this combination provides more precise models than using PCC models whereas less accurate models when using FEA methods.

In order to deal with computational complexity of existing methods for modeling and control of foldable robots, many studies investigated to use neural networks and specifically DeepNNs into their control approach to learn the highly-nonlinear behavior of robots. Several successful deployment of neural networks into control approaches are in applications such as underwater vehicles [16], ships [17], and robotic manipulators [18]–[21]. Given the complexity of these nonlinear robotic systems, traditional linear models results in inaccurate models. Neural networks offers characterization of these nonlinearities directly through data sampling during the learning process. While neural networks have been studied in depth for analyzing both forward kinematic and inverse kinematic problems of rigid robots [22]–[24], to the knowledge of the authors, a closed-loop controller based on DeepNNs has not been introduced for analysis and compensation of the material deformation in foldable and soft robots.

Figure 1 shows an example of the effect of external loading on the end-effector of a 2-DoF parallel robot with foldable joints and flexible links in a potential pointing task. Figure 1(a) illustrates the end-effector of the robot in two static conditions, namely no-load and loaded. We can see the deflection of the loaded end-effector from its no-load position. In Figure. 1(b), we show the effect of the end-effector deflection in tracking of a desired spiral trajectory, which results in large amount of errors between the desired trajectory (red) and the tracked trajectory (blue).

In contrast with computational approaches that are used to solve soft robotic control problems, the objective of this paper is to investigate alternative strategies that simultaneously allow us to represent and solve for the complex inverse kinematics of parallel mechanisms and the position error due to deformation under load for moderately-compliant laminates. This is possible because the motion of laminate robots is still primarily dictated by the geometric relationships of flexure hinges rather than the deformation of stiffer links, making traditional, rigid-link models a good starting point for representing these mechanisms. This paper presents two different methods based on DeepNNs for data-driven control of foldable robots. The main contributions of this paper are as follows:

- We propose two control methods, called model-free and model-based methods, based on DeepNNs for compensating the deformation of foldable robot's end-effector.
- Our proposed control methods are computationally tractable which enables the real-time training of the DeepNN and implementation of the controllers.
- Our proposed model-free control method does not require any priori model of the robot, which allows to use this method when the derivation of an accurate kinematic model the foldable robot is a challenging task.

In Section II, we first introduce the 2-DoF foldable robot shown in Figure. 1(c,d) and derive analytical formulations for the robot's inverse and forward kinematics. In Section III, we provide a workflow in three main stages which describe data sampling to create datasets required for training of DeepNNs, tuning the hyper-parameters of DeepNNs, and implementation of closed-loop controllers. In Sections IV and V, we respectively present our proposed model-based and model-free control methods based on DeepNNs and provide experimental implementation results of using these two control methods in trajectory tracking by the 2-DoF foldable robot.

II. 2-DoF Laminated Parallel Robot

Figure 1(d) demonstrates the 2-DoF laminated parallel robot that we used as a case study to evaluate our proposed control approaches in tracking of a desired spiral trajectory shown in Figure. 1(c). This foldable parallel robot was originally introduced as a 2-DoF spherical orienting mechanism in [25], [26]. The design of robot is based on the kinematic synthesis analysed by Ouerfelli et al. that maximizes its workspace [27]. In [28], we have developed a small-scale version of this robot as a camera stabilizer while taking advantage of laminated techniques in its design procedure. In this paper, we extend our previous study by introduction of a larger-scale version of the foldable robot that also permits deformation under external load on its end-effector. This robot has been constructed via a laminate fabrication process similar to the procedure described in [29], [30]. The final prototype of the robot was made of $0.762\,\text{mm}$ fiberglass

sheets as the rigid layer and 0.127 mm polyester sheet as the flexible layer. A heat-activated acrylic adhesive from Drytac[1] has been used to bond layers. The actuators of the robot are two XM430 Dynamixel DC servo motors. The two custom-made Nylon 3D-printed horns are responsible for attaching and aligning mechanism hinges to the servo motors. The horns essentially act as a safety coupling in the mechanism. Moreover, the chassis has been built from acrylic and 3D-printed parts.

A. Inverse Kinematics

The inverse kinematics of the robot solves for the angular position of the robot's actuators given the orientation of the robot's end-effector. Considering the 2-DoF foldable robot shown in Figure. 1(d) and based on the global axes' alignment with servo actuators, the inverse kinematic equations of the robot can be written as [27]:

$$\theta_1 = \arctan(\frac{n_y n_z}{n_x^2 + n_z^2}) \quad \theta_2 = \arctan(\frac{n_x}{n_z}) \tag{1}$$

where θ_i are the analytical angular position of i-th actuator and n_x, n_y, and n_z are the components of the orthonormal unit vectors on the robot's end-effector. In a special case when $n_z = 0$ (a representational singularity), the actuators' angular positions are assumed to be $\theta_1 = 0$ and $\theta_2 = \pi/2$.

B. Forward Kinematics

The mechanism's forward kinematics provides the orientation of the robot's end-effector given the angular position of the robot's actuators. Accordingly, the forward kinematics of the 2-DoF foldable robot can be formulated as [27]:

$$\begin{aligned} n_z &= \left((t_1 + t_1 t_2)^2 + t_2^2 + 1\right)^{-\frac{1}{2}} \\ n_x &= t_2 n_z \\ n_y &= (t_1 + t_1 t_2) n_z, \end{aligned} \tag{2}$$

where $t_i = \tan(\theta_i)_{\{i=1,2\}}$. Considering Eqs. (2), we can span the 2D space of the robot's actuators, i.e. θ_1 and θ_2, in order to obtain the end-effector workspace as shown by blue dots in Figure. 1(c). Furthermore, we specify the desired spiral trajectory such that it lies on workspace of the robot. This spiral trajectory is used throughout this paper as the desired path for tracking by the robot and is basically selected as to entirely traverse the workspace by projection of the normal vectors of the end-effector on a sphere with the radius of 1 m.

III. METHODOLOGY

In this section, we describe the step-by-step implementation procedure of our proposed control methods in a workflow as: 1) Defining the inputs and ouputs; 2) Sampling the data under no load condition; 3) Sampling the data under loaded condition; 4) Designing the DeepNNs; 5) Training the DeepNNs; 6) Tuning the DeepNNs with hyper-parameters; 7) Implementing control methods; 8) Evaluating control methods; 9)Parameter tuning of updating DeepNN.

[1]Drytac Multi-Heat Adhesive, Drytac, USA

We explain these control methods in detail in the following sections on model-based control in Section IV and model-free control in Section V. This procedure is valid for each control method and is implemented in three main stages including data sampling, modeling, and control. We explain each stage in details as follows.

A. Data Sampling

[Step 1: Define input and outputs] The procedure starts by defining the inputs and outputs (ground-truth data) for each DeepNN in the control method. Then, we create two datasets of inputs-outputs. The first dataset consists of a set of inputs-outputs, where the inputs are different orientations of the end-effector, $\mathbf{q}$ and the outputs are the errors of the actuators' angular positions, i.e. $\Delta\theta_1^m = \theta_1^m - \theta_1$ and $\Delta\theta_2^m = \theta_2^m - \theta_2$ where θ_1^m and θ_2^m are the measured angular position of the actuators recorded by the actuators' encoders. Additionally, we create another set of inputs-outputs. In the second dataset, the inputs are the same as the inputs of the first dataset, $\mathbf{q}$, whereas the outputs are the measured angular positions of the actuators, i.e. θ_1^m and θ_2^m. **[Step 2: Sampling data (no-load)]** When the inputs-outputs of DeepNNs are determined, we start sampling the required data when the robot's end-effector is in undeformed (no-load) condition. **[Step 3: Sampling data (loaded)]** Then, we also sample data when the robot is in deformed (loaded) with different loading condition. We note that the sample data of undeformed and deformed end-effector should span a large portion of the robot's workspace. In addition, the dataset is split into three sets of data, namely train, validation, and test sets.

B. Modeling

[Step 4: DeepNN design] In the modeling stage, we aim to train two types of DeepNNs. The first type of the DeepNN learns a mapping between the orientation of the end-effector represented in quaternions as the inputs and the the angular position error of the actuators as the ground-truth outputs. In the second type of DeepNN that learns the inverse kinematic (IK) model of the robot, the inputs are the orientation of the robot's end-effector in quaternions and the ground-truth outputs are the angular position of the actuators. We first set the hyper-parameters for each DeepNN. The hyper-parameters are the number of hidden layers, number of neurons at each layer, activation functions, learning rate, and batch size. Moreover, different optimization algorithms are used for training. **[Step 5: Training DeepNNs]** Then, we use the train dataset including shuffled data corresponding to both undeformed and deformed conditions of the robots to train the two DeepNNs with different hyper-parameters. We examined different activation functions such as identity, logistic regression, tanh, and ReLU. Moreover, we tried different number of layers from a simple network with one hidden layer to an extremely DeepNN with 3000 hidden layers with two powerful optimization algorithms called ADAM (Adaptive Moment Estimation) and LBFGS (Broyden, Fletcher, Goldfarb, and Shanno) in training of DeepNNs. **[Step 6: DeepNN hyper-parameters tuning]** By trying

different hyper-parameters and optimization methods to train DeepNNs, one can find the best parameters by which the DeepNN provides an input-output mapping with relatively less training error, training time, and variance. The identity as the activation function with LBFGS as the optimization algorithm results in the best performance of the first type of DeepNN used in the model-based control method. On the other hand, this can be achieved when we use ReLU as the activation function and LBFGS as the optimization algorithm in the second type of DeepNN in the model-free controller.

C. Control

[Step 7: Control methods implementation] In the control stage, we can use either model-based or model-free control methods. In the model-based method, we use the analytical IK model of the robot described in Eq. (1) to find the desired actuators' angular position given the desired orientation of the robot's end-effector. However, the model-free control method does not require the analytical kinematic model of the robot. As a *target* network, we employ the first type of DeepNN in model-based method and second type of DeepNN in the model-free method. We also use a second DeepNN namely *updating* network that is trained in real-time with measured sensor data in order to back up new weights. The updating network periodically updates the target network with these new weights in real-time to prevent the divergence of the target network's weights. **[Step 8: Control methods evaluation]** Then, we study the performance of control methods in compensation of the robot's deformation on the end-effector using the validation dataset. If the trained target DeepNN suffers from high variance, the hyper-parameters must be tuned to avoid over-fitting the training dataset. **[Step 9: Parameter tuning of updating DeepNN]** We also need to find the optimal batch size that results in relatively minimum training error and training time with the lowest variance of the model difference between the training error and validation error. We found 1000 as the best batch size in our experiments.

IV. Model-Based Control Method

In this section, we describe our proposed model-based control method as well as the experimental implementation results of tracking the desired spiral trajectory by the robot.

A. Approach

Figure 2(a) shows the block diagram of our proposed model-based control methods. We note that the target network is pre-trained offline with the train dataset. In online implementation of the control method, the pre-trained target DeepNN outputs approximations for the desired errors of the actuators' angular position, $\Delta\tilde{\theta}_1^d$ and $\Delta\tilde{\theta}_2^d$, given the desired orientation of the robot's end-effector, $\mathbf{q}^d$. The analytical desired angular position of the actuators, θ_1^d and θ_2^d, are calculated in parallel using the IK model of the robot. We sum the outputs of the target DeepNN and the IK model to compute the compensated desired angular position as the actuators' setpoints, $\tilde{\theta}_1^d$ and $\tilde{\theta}_2^d$. In turn, the orientation of the

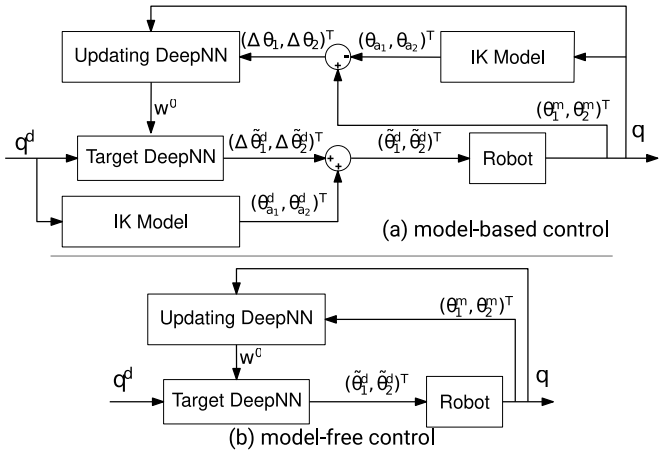

Fig. 2. **System configurations.** (a) Block diagram of model-based control method. (b) Block diagram of model-free control method.

robot's end-effector, $\mathbf{q}_m$, is measured by overhead cameras. There is always an error between the measured orientation and the desired orientation while the robot is tracking the desired trajectory. Thus, we have to close the control loop in order to compensate this error. For this purpose, we train the updating DeepNN in real-time with the measured data whenever 300 set of measurements are available. The inputs to the updating network is the measured orientation of the end-effector and its ground-truth outputs are the errors of the angular position of the actuators calculated based on the measured angular positions of the actuators, i.e. $\Delta\theta_1^m$ and $\Delta\theta_2^m$. We update the weights of the target network, $\mathbf{w}$, with the new weights, $\mathbf{w}'$, when the training of updating network is completed.

B. Implementation Results

In our experimental setup, we have two processes which are executed in parallel to receive feedback from the camera and to send control commands to the actuators of the robot. This parallelization allows each process to have its own refresh rate. The closed-loop control process has an average frequency of 160 Hz. It is worthwhile to note that the robot completes one spiral every 30 s, a rate that is limited by the closed-loop control frequency. In this time range, 4800 data points are sampled in the control loop. The performance of the controller is evaluated under different loading conditions while tracking the desired spiral trajectory. We start with the no-load condition. Next, we add a 100 g fixed weight to the robot's end-effector. Then, an 80 g "variable" load is added to the end-effector of the robot as well. This load is variable since it consists of small freely moving masses that are packed in a container that is attached to the robot's end-effector. By this experiment, we essentially evaluate the performance of the model-based controller in three different conditions, namely no-load, fixed-load, and variable-load conditions. The transition between these loading conditions are applied when we let the robot to complete three turns of the spiral. Figure 3(b) shows the Mean Absolute Error (MAE) of the controller's performance in tracking of the

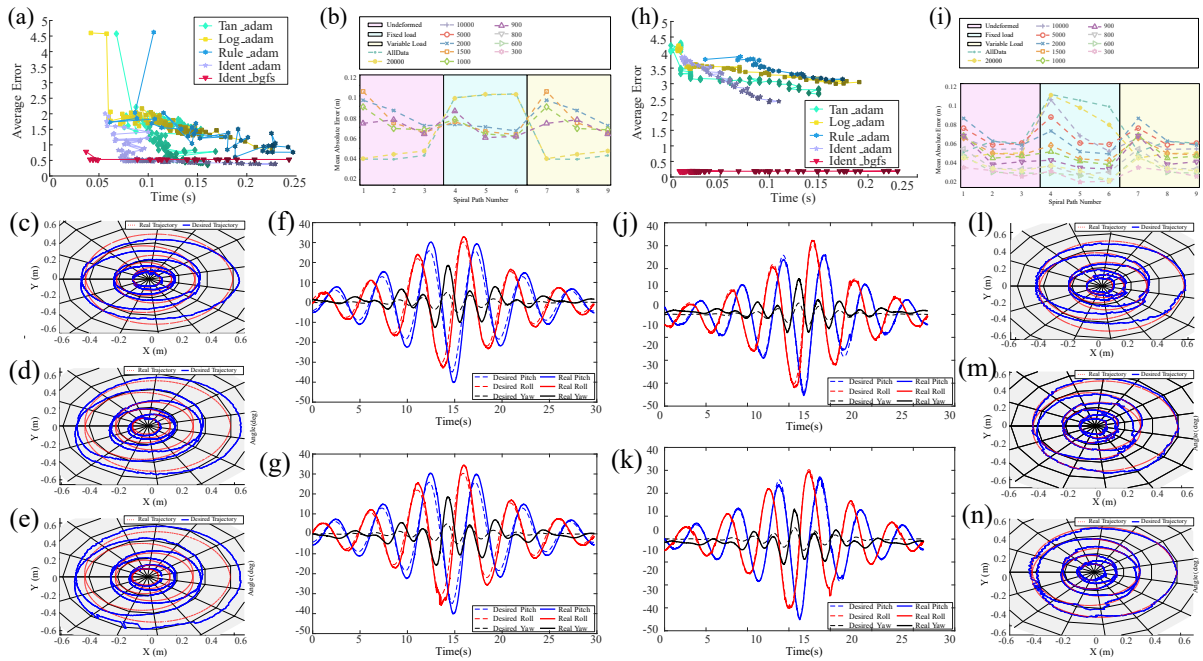

Fig. 3. **(a)-(g):Trajectory tracking using model-based control method. (h)-(n):Trajectory tracking using model-free control method.** (a) DeepNN tuning for deformation error. (b) Tuning the retraining history in the control. (c) Trajectory tracking by pre-trained DeepNN. (d) Path tracking under fixed load in top view. (e) Path tracking under variable load in top view. (f) Path tracking under fixed load in Euler angles. (g) Path tracking under variable load in Euler angles. (h) DeepNN tuning for inverse kinematics. (i) Tuning the retraining history in control. (j) Path tracking under fixed load in Euler angles. (k) Path tracking under variable load in Euler angles. (l) Trajectory tracking by Pre-trained DeepNN. (m) Path tracking under fixed load in top view. (n) Path tracking under variable load in top view.

desired spiral trajectory in different loading conditions. For better illustration, we show the graphs of errors separately for each loading condition. The experimental results show that the batch size of data that we used to train the updating network has significant impact on the closed-loop controller speed. We initially set this batch size to 300, but the experiments show that the best performance is achieved when this batch size is set to 1000 with our computational system configuration. The smaller batch sizes, e.g. 900, provide smaller training error while it suffers from over-fitting and the controller will not be able to compensate the deformation errors properly. Figures 3(e), and 3(g) illustrate the desired spiral trajectory and the tracked path by the robot in different scenarios of constant and variable loading conditions. The Euler angles including the roll, pitch, and yaw of the end-effector are shown in Figures. 3(f) and 3(g) for the same paths and loading conditions.

V. Model-Free Control Method

In this section, we describe our proposed model-free control method as well as the experimental implementation results of tracking the desired spiral trajectory by the robot.

A. Approach

Figure 2(b) shows the block diagram of our proposed model-free control method. As its name would suggest, a model-free controller does not depend on the analytical kinematic model of the robot while this model is directly learned by the second type of DeepNN. We should note that the target network is pre-trained offline with the training dataset. In online implementation, the desired orientations of the robot's end-effector, $\mathbf{q}^d$, are given as the inputs of the DeepNN and the outputs are approximation of desired angular positions of the actuators, i.e. $\tilde{\theta}_1^d$ and $\tilde{\theta}_2^d$. These approximated angular positions are directly fed to the robot's actuators. Then, we store the measured orientation of the end-effector, $\mathbf{q}_m$, using overhead cameras as well as the angular positions measured by the actuator encoders. Similar to the model-based control method, we employ an updating network to update the weights of the target network when 300 set of measured data are available. In fact, the updating network learns the IK of the robot in the presence of material deformation caused by different loading conditions.

B. Implementation Results

Figure 3(h) shows the error of training the IK model of the robot under deformation by the DeepNN with different optimization algorithms and activation functions. Based on experimental results, the best performance of DeepNN is achieved with two hidden layers with identity activation function and LBFGS as the optimization algorithm. Similar to the previous case, the performance of the model-free controller is evaluated under different loading conditions including no-load, constant loading (100 g), and variable loading (80 g). Figure 3(i) shows the MAE of tracking the desired spiral trajectory by the robot at each loading condition. The experimental results show that the best batch size for the updating network is 500 data points. Figures 3(l), 3(m),

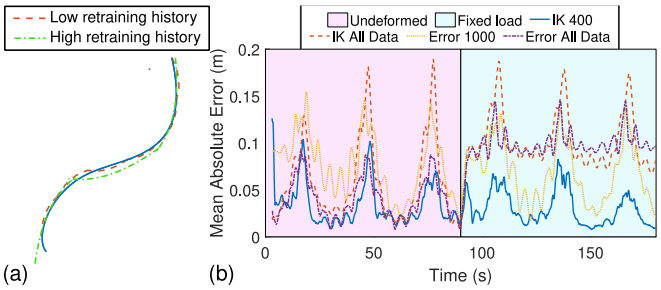

Fig. 4. (a) Effects of retraining history. (b) Path tracking errors under the optimal scenario.

and 3(n) illustrate the desired spiral trajectory and the tracked path by the robot's end-effector in different loading conditions. We also show the corresponding end-effector's Euler angles in Figures. 3(j) and 3(k).

VI. Discussion and Analysis

The implementation of model-free controller is straightforward compared to the model-based controller implementation since there is no need to derive the analytical model of the robot. This independence from the model allows us to utilize the model-free method in the control of various foldable robots, because the kinematic model of the robot can be learned by DeepNNs in the loop. In the case that we do not employ the updating network, we have an open-loop control system in which the target network is able to to control the robot to track the desired trajectory with small tracking error in no-load condition. However, this open-loop controller does not adapt to loaded conditions properly, especially when the load is variable. Thus, the updating network plays an important role in compensation of the tracking error via online training of new loading conditions. In this way, the adaptation to unseen loading conditions, which are applied by the test set, will be performed properly in real-time by the closed-loop controller.

Tuning of hyper-parameters and the optimal design of DeepNNs are the most crucial steps in implementation of our proposed control methods. Accordingly, it is highly recommended to have a validation set to examine different hyper-parameters in order to find optimal parameters. The best hyper-parameters should minimize training error and training time and minimize the variance of the model. The issue of high variance results in over-fitting of the model, which in turn causes large errors of tracking under unseen loading conditions. It is of significant importance to tune the batch size since it can directly affect the stability of training and limit the training speed of updating DeepNN in the closed-loop controller. Figure 4 shows the effect of choosing different batch size for online training of the updating DeepNN. As shown in Figure. 4, we also plots the error graphs for training of IK model of the robot by the updating DeepNN with different batch size in no-load and loaded conditions.

The model-based control method uses DeepNN to accommodate for the error found between the IK solution and reality. Its limited performance led us to develop the model-free approach, which exhibits better performance. We believe we can attribute this improvement to the higher computational cost of solving the IK Model, which impacts controller frequency, vs the increased closed-loop frequency of the model-free controller.

VII. Conclusion

In this paper, we proposed two closed-loop model-based and model-free control methods based on DeepNNs for compensating the material deformation of laminated robots. We described a workflow for implementation of our proposed control methods in three stages that describe sampling the data, design of the DeepNNs, tuning the hyper-parameters of DeepNNs, and implementation of the control methods. Furthermore, two different DeepNNs were introduced, one to train directly the IK model of the robot and another to train the given the actuation space errors given the robot's task space information. The performance of our proposed control methods were evaluated in tracking of a desired spiral trajectory by a 2-DoF laminated foldable robot. The experimental results show the effectiveness of the control methods while the model-free controller outperforms the model-based control method in that it results in smaller tracking errors under different loading conditions.

Future work includes comparison between the proposed control approach with the traditional closed-loop controller and other machine learning techniques. There are many potential future directions to improve our proposed control methods. We aim to include system dynamics in the controllers' design and compensate the deformation of foldable robots caused by dynamic loads. We also want to test our proposed control methods on different foldable platforms to evaluate the effectiveness of the methods on other robots.

References

[1] P. Moseley, J. M. Florez, H. A. Sonar, G. Agarwal, W. Curtin, and J. Paik, "Modeling, Design, and Development of Soft Pneumatic Actuators with Finite Element Method," *Advanced Engineering Materials*, no. 6, 2015.

[2] C. Duriez, "Control of elastic soft robots based on real-time finite element method," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 3982–3987, 2013.

[3] K. Suzumori, S. Endo, T. Kanda, N. Kato, and H. Suzuki, "A Bending Pneumatic Rubber Actuator Realizing Soft-bodied Manta Swimming Robot," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE, apr 2007, pp. 4975–4980. [Online]. Available: http://ieeexplore.ieee.org/document/4209864/

[4] J. Hiller and H. Lipson, "Dynamic simulation of soft heterogeneous objects," *arXiv preprint arXiv:1212.2845*, 2012.

[5] R. J. Webster and B. A. Jones, "Design and kinematic modeling of constant curvature continuum robots: A review," *International Journal of Robotics Research*, vol. 29, no. 13, pp. 1661–1683, 2010.

[6] S. I. Rich, R. J. Wood, and C. Majidi, "Untethered soft robotics," *Nature Electronics*, vol. 1, no. 2, pp. 102–112, 2018.

[7] O. Goury and C. Duriez, "Fast, Generic, and Reliable Control and Simulation of Soft Robots Using Model Order Reduction," *IEEE Transactions on Robotics*, vol. 34, no. 6, pp. 1565–1576, 2018.

[8] R. J. Webster III 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.

[9] S. Huang, Q. Zhang, Z. Liu, X. Wang, and B. Liang, "Control of a piecewise constant curvature continuum manipulator via policy search method," *2018 IEEE International Conference on Robotics and Biomimetics, ROBIO 2018*, no. 61673239, pp. 1777–1782, 2019.

[10] H. Yang, M. Xu, W. Li, and S. Zhang, "Design and Implementation of a Soft Robotic Arm Driven by SMA Coils," *IEEE Transactions on Industrial Electronics*, vol. 66, no. 8, pp. 6108–6116, 2019.

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

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

[13] I. S. Godage, Y. Chen, K. C. Galloway, E. Templeton, B. Rife, and I. D. Walker, "Real-time Dynamic Models for Soft Bending Actuators," *2018 IEEE International Conference on Robotics and Biomimetics, ROBIO 2018*, pp. 1310–1315, 2019.

[14] I. S. Godage, G. A. Medrano-Cerda, D. T. Branson, E. Guglielmino, and D. G. Caldwell, "Dynamics for variable length multisection continuum arms," *International Journal of Robotics Research*, vol. 35, no. 6, pp. 695–722, 2016.

[15] T. M. Bieze, F. Largilliere, A. Kruszewski, Z. Zhang, R. Merzouki, and C. Duriez, "Finite element method-based kinematics and closed-loop control of soft, continuum manipulators," *Soft Robotics*, vol. 5, no. 3, pp. 348–364, 2018.

[16] B. Miao, T. Li, and W. Luo, "A DSC and MLP based robust adaptive NN tracking control for underwater vehicle," *Neurocomputing*, vol. 111, pp. 184–189, 2013. [Online]. Available: http://dx.doi.org/10.1016/j.neucom.2012.12.026

[17] G. Zhang and X. Zhang, "Concise robust adaptive path-following control of underactuated ships using DSC and MLP," *IEEE Journal of Oceanic Engineering*, vol. 39, no. 4, pp. 685–694, 2014.

[18] A. T. Hasan, N. Ismail, A. M. Hamouda, I. Aris, M. H. Marhaban, and H. M. Al-Assadi, "Artificial neural network-based kinematics Jacobian solution for serial manipulator passing through singular configurations," *Advances in Engineering Software*, vol. 41, no. 2, pp. 359–367, 2010. [Online]. Available: http://dx.doi.org/10.1016/j.advengsoft.2009.06.006

[19] P. Jha and B. B. Biswal, "A Neural Network Approach for Inverse Kinematic of a SCARA Manipulator," *IAES International Journal of Robotics and Automation (IJRA)*, vol. 3, no. 1, pp. 52–61, 2014.

[20] J. W. Park, R. G. Harley, and G. K. Venayagamoorthy, "Indirect adaptive control for synchronous generator: Comparison of MLP/RBF neural networks approach with Lyapunov stability analysis," *IEEE Transactions on Neural Networks*, vol. 15, no. 2, pp. 460–464, 2004.

[21] S. Alavandar and M. J. Nigam, "Neuro-fuzzy based approach for inverse kinematics solution of industrial robot manipulators," *International Journal of Computers, Communications and Control*, vol. 3, no. 3, pp. 224–234, 2008.

[22] A. H. E. Ezzat A Showaib, "Artificial Neural Network Based Forward Kinematics Solution for Planar Parallel Manipulators Passing through Singular Configuration," *Advances in Robotics & Automation*, vol. 02, no. 02, 2013.

[23] B. Daya, S. Khawandi, and M. Akoum, "Applying Neural Network Architecture for Inverse Kinematics Problem in Robotics," *Journal of Software Engineering and Applications*, vol. 03, no. 03, pp. 230–239, 2010.

[24] D. T. Pham, M. Castellani, and A. A. Fahmy, "Learning the inverse kinematics of a robot manipulator using the Bees Algorithm," *IEEE International Conference on Industrial Informatics (INDIN)*, no. Indin, pp. 493–498, 2008.

[25] C. M. Gosselin and F. Caron, "Two degree-of-freedom spherical orienting device," Oct. 19 1999, uS Patent 5,966,991.

[26] E. Samson, D. Laurendeau, M. Parizeau, S. Comtois, J.-F. Allan, and C. Gosselin, "The agile stereo pair for active vision," *Machine Vision and Applications*, vol. 17, no. 1, pp. 32–50, 2006.

[27] M. Ouerfelli and V. Kumar, "Optimization of a spherical five-bar parallel drive linkage," *Journal of mechanical design*, vol. 116, no. 1, pp. 166–173, 1994.

[28] M. Sharifzadeh, Y. Jiang, R. Khodambashi, and D. M. Aukes, "Increasing the life span of foldable manipulators with fabric," in *ASME 2020 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. American Society of Mechanical Engineers Digital Collection, 2020.

[29] J. P. Whitney, P. S. Sreetharan, K. Y. Ma, and R. J. Wood, "Pop-up book MEMS," *Journal of Micromechanics and Microengineering*, vol. 21, no. 11, p. 115021, nov 2011. [Online]. Available: http://stacks.iop.org/0960-1317/21/i=11/a=115021?key=crossref.4ebe6c2c4c4804ab44e0dfb88e1b355e

[30] P. S. Sreetharan, J. P. Whitney, M. D. Strauss, and R. J. Wood, "Monolithic fabrication of millimeter-scale machines," *Journal of Micromechanics and Microengineering*, vol. 22, no. 5, p. 055027, may 2012. [Online]. Available: http://stacks.iop.org/0960-1317/22/i=5/a=055027?key=crossref.491915c123069b686e444a77780882a9