Localized Motion Dynamics Modeling of A Soft Robot: A Data-Driven Adaptive Learning Approach

Xiaotian Chen, Paolo Stegagno, Wei Zeng, Chengzhi Yuan

***Abstract*— Soft robots have recently drawn extensive attention thanks to their unique ability of adapting to complicated environments. Soft robots are designed in a variety of shapes of aiming for many different applications. However, accurate modelling and control of soft robots is still an open problem due to the complex robot structure and uncertain interaction with the environment. In fact, there is no unified framework for the modeling and control of generic soft robots. In this paper, we present a novel data-driven machine learning method for modeling a cable-driven soft robot. This machine learning algorithm, named deterministic learning (DL), uses soft robot motion data to train a radial basis function neural network (RBFNN). The soft robot motion dynamics are then guaranteed to be accurately identified, represented, and stored as an RBFNN model with converged constant neural network weights. To validate our method, We have built a simulated soft robot almost identical to our real inchworm soft robot, and we have tested the DL algorithm in simulation. Furthermore, a neural network weight combining technique is used which can extract and combine useful dynamics information from multiple robot motion trajectories.**

I. INTRODUCTION

Soft robotics takes inspiration from living organisms that use compliant materials to make robots highly adaptive to their surroundings. Thanks to this characteristic, a large number of soft robots have been designed and researched during the last decade. Some well-known examples of soft manipulators include octopus inspired underwater manipulators [1], flexible sac shaped grippers [2], pneumatic actuated soft grippers [3], and many others [4], [5], [6]. As for soft mobile robots, starfish shaped [7], meshworms [8], caterpillars [9], four-leg inchworm soft robots [10], and so on [11], [12] have been proposed.

Compared to conventional rigid body robots, soft robots have more potential for safe human-robot interaction, resilience to perturbation, and flexibility to multi-task setup. However, the challenge that remains for soft robotics is how to accurately model the robot dynamics. Unlike rigid-body robots with known finite dimensionality, soft robots are typically of infinite-dimensional structures. Also, soft robots could have oscillation during the interaction with the environments, and cause unexpected error during the modeling. Additionally, soft robots come in many different types and shapes making it almost impossible to find a generic model (with unknown parameters) suitable for all soft robots. Currently, most soft robotics modeling techniques use continuous mathematics to describe the robot dynamics and further lead to Piecewise Constant Curvature (PCC) models. However, this leads to a simplified approximate model with respect to the real soft robot dynamics.

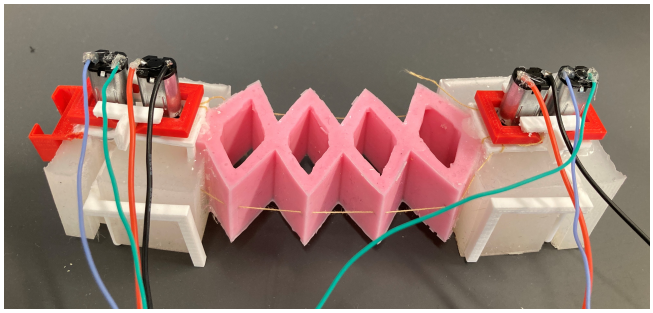

Fig. 1: An improved prototype inchworm soft robot from [18].

Recently, data-driven machine learning methodologies, e.g., [13], have become popular in robotics. As robots keep growing in complexity (especially soft robot), accurate soft robot dynamics modeling becomes extremely difficult. By using machine learning to analyze data from robot motion, the modeling can be achieved with little pre-knowledge of the robot. [14] and [15] present the ideas of using machine learning to model and predict the external environment onto a soft finger, proving that machine learning can deal with complex soft sensor information. In [16] and [17], a well designed and fabricated soft manipulator, the Bionic Handling assistant (BHA), is presented to be used for modelling and control. The authors combined the data-driven approach with continuum kinematics model to improve the modelling accuracy.

In this paper, we propose to use a novel machine learning approach, named deterministic learning (DL), to achieve locally-accurate identification of the motion dynamics of a prototype soft robot system using radial basis function NNs ([19], [20]). This approach is developed from the conventional adaptive learning control theory using Lyapunov function method to ensure provable convergence of the trained NN model. In our previous research, we have designed and developed an inchworm soft robot [18] which uses a cable-motor actuator and friction switching structures

This work was partially supported by the National Science Foundation under grant: CMMI-1929729.

X. Chen and C. Yuan are with the Department of Mechanical, Industrial and Systems Engineering, University of Rhode Island, Kingston, RI 02881, USA xiaotian_chen@my.uri.edu; cyuan@uri.edu

P. Stegagno is with the Department of Electrical, Computer and Biomedical Engineering, University of Rhode Island, Kingston, RI 02881, USA pstegagno@uri.edu

W. Zeng is with the School of Physics, Mechanical and Electrical Engineering, Longyan University, Longyan 364012, China zw0597@126.com

to enable locomotion of the soft robot. In this paper, we have improved the design of the main deformable body (Fig. 1) which will let the robot move more stably and efficiently. Based on this robot, we have constructed a virtual robot in a soft material simulation software, SOFA. Then we have studied the robot's motion and tested the modeling algorithm in the simulation. Specifically, a set of robot motion data have been collected in simulation, and we have used an RBFNN to locally-accurately approximate the robot's nonlinear uncertain motion dynamics along each collected data trajectory. Finally, we have computed the overall robot motion dynamics by merging the information from each resulting individual localized RBFNN model.

The rest of this paper is organized as follow. Section II presents some preliminaries on the deterministic learning theory. Section III discusses the localized dynamics learning algorithm. Section IV introduces the detail of our soft robotic system. Section V describes the simulation setup. Section VI gives the experimental setup and results. Finally, Section VII concludes the paper.

II. PRELIMINARIES

In deterministic learning theory [21], [22], identification of system dynamics of general nonlinear systems is achieved through the following elements: (i) employment of localized RBFNNs; (ii) satisfaction of a partial persistence of excitation (PE) condition; (iii) exponential stability of the adaptive system along the periodic or recurrent orbit; (iv) locally accurate neural network (NN) approximation of the unknown system dynamics. An RBFNN can be described by $f_{nn}(Z) = \sum_{i=1}^{N} w_i s_i(Z) = W^T S(Z)$, where $\mathbf{Z} \in \Omega_{\mathbf{Z}} \subseteq \mathbb{R}^q$ is the input vector, with $\Omega_{\mathbf{Z}}$ is a compact set, $W = \left[w_1, ..., w_N\right]^T \in \mathbb{R}^N$ is the weight vector, N is the NN node number, and $S(Z) = [s_1(||\mathbf{Z} - \mu_i||), ..., s_N(||\mathbf{Z} - \mu_i||)]^T$, with $s_i(\cdot)$ being a radial basis function, and $\mu_i (i = 1, ..., N)$ being distinct points in the state space. The Gaussian function $s_i(||\mathbf{Z} - \mu_i||) = \exp\left[-\frac{(\mathbf{Z}-\mu_i)^T(\mathbf{Z}-\mu_i)}{\eta_i^2}\right]$ is one of the most commonly used radial basis functions, where $\mu_i = [\mu_{i1}, \mu_{i2}, ..., \mu_{iN}]^T$ is the center of the receptive field and η_i is the width of the receptive field. The Gaussian function belongs to the class of localized radial basis function s in the sense that $s_i(||\mathbf{Z} - \mu_i||) \to 0$ as $||Z|| \to \infty$.

It has been shown in [23], [24] that for any continuous function $f(Z) : \Omega_{\mathbf{Z}} \to R$ where $\Omega_{\mathbf{Z}} \subset R^p$ is a compact set, and for the NN approximator, where the node number N is sufficiently large, there exists an ideal constant weight vector W^*, such that for each $\epsilon^* > 0$, $f(Z) = W^{*T} S(Z) + \epsilon(Z)$, $\forall Z \in \Omega_{\mathbf{Z}}$, where $\epsilon(Z)$ is the approximation error and is smaller than ϵ^*. Moreover, for any bounded trajectory $Z_\zeta(t)$ within the compact set $\Omega_{\mathbf{Z}}$, $f(Z)$ can be approximated by using a limited number of neurons located in a local region along the trajectory: $f(Z) = W_\zeta^{*T} S_\zeta(Z) + \epsilon_\zeta$, where $S_\zeta(Z) = [S_{j1}(Z), ..., S_{j\zeta}(Z)]^T \in R^{N_\zeta}$, with $N_\zeta < N$, $|s_{ji}| > \iota$, $(ji = j1, ..., j\zeta)$, ι is a small positive constant, $W_\zeta^* = [w_{j1}^*, ..., w_{j\zeta}^*]^T$, and ϵ_ζ is the approximation error, with $\Big||\epsilon_\zeta| - |\epsilon|\Big|$ being small.

Based on previous results on the PE property of RBFNNs [25], it is shown in [26] that for a localized RBF network $W^T S(Z)$ whose centers are placed on a regular lattice, almost any recurrent trajectory $Z(t)$ can lead to the satisfaction of the PE condition of the regressor sub-vector $S_\zeta(Z)$.

III. THE ADAPTIVE LEARNING ALGORITHM

To achieve accurate motion dynamics modeling of soft robots, we present a new data-driven machine learning algorithm by extending the continuous-time deterministic learning algorithm to discrete-time systems.

We consider the following nonlinear dynamical system described in a general form:

$$\dot{x} = f(x; p) \tag{1}$$

where x is the state of the system with initial condition $x(t_0) = x_0$, the vector p is a constant parameter of the system, and $f(x; p)$ is a smooth but unknown nonlinear vector field. According to the methodology of [19], by using the Euler approximation, a discretized system representation of the Equation (1) can be written as

$$x[k+1] = x[k] + T_s f(x[k]; p), \qquad x[0] = x_0. \tag{2}$$

where $x[k]$ is the system state, $f(x[k]; p)$ is the unknown dynamics representing the knowledge that we are trying to obtain from the system states/trajectories, and T_s is the duration of a sampling timestep. Based on the discrete-time deterministic learning theory [19], the dynamical RBF neural network can be employed to identify this unknown dynamics through the following algorithm:

$$\hat{x}[k+1] = x[k] + a(\hat{x}[k] - x[k]) + T_s \hat{W}^T[k+1] S(x[k]) \tag{3}$$

where $\hat{x} = [\hat{x}_1, \hat{x}_2, ..., \hat{x}_n]$ is the state vector, $0 < |a| < 1$ is a constant number, $\hat{W}^T S(x[k])$ is the RBF neural network with $\hat{W}$ as the neural network weight. This weight will be updated through the following learning law:

$$\hat{W}[k+1] = \hat{W}[k] - \frac{\alpha P(e[k] - ae[k-1]) S(x[k-1])}{1 + \lambda_{max}(P) S^T(x[k-1]) S(x[k-1])} \tag{4}$$

where $P = P^T > 0$, and $\lambda_{max}(P)$ denotes the largest eigenvalue of the matrix P, $\alpha \in (0, 2)$ is the learning gain for design, and $e[k] = \hat{x}[k] - x[k]$.

According to [19], with Equations (3) and (4), we are able to accurately identify the unknown dynamics $f(x[k]; p)$ in (2) by ensuring the overall system stability and NN weights converging to their true values. To show this, we first derive the following error dynamics:

$$e[k+1] = ae[k] + T_s \tilde{W}^T[k+1] S(x[k]) - T_s \epsilon \tag{5}$$

where $\epsilon = f(x; p) - W^{*T} S(x)$ is the ideal NN approximation error, with W^* defined in preliminary section. The RBF network weight error equation can be written as

$$\begin{aligned} \tilde{W}[k+1] &= \hat{W}[k+1] - W^* \\ &= A[k]\tilde{W}[k] + B[k]. \end{aligned} \tag{6}$$

where $A[k] = I - \dfrac{\alpha T_s P(S^T x[k-1]S(x[k-1]))}{1+\lambda_{max}(P)S^T(x[k-1])S(x[k-1])}$ and $B[k] = \dfrac{\alpha T_s P S(x[k-1])\epsilon}{1+\lambda_{max}(P)S^T(x[k-1])S(x[k-1])}$.

From Equations (5) and (6), we can derive the perturbed error dynamics as:

$$\begin{bmatrix} e[k+1] \\ \tilde{W}[k+1] \end{bmatrix} = \begin{bmatrix} a & T_s S^T(x[k])A[k] \\ 0 & A[k] \end{bmatrix} \begin{bmatrix} e[k] \\ \tilde{W}[k] \end{bmatrix} + \begin{bmatrix} T_s\epsilon(\dfrac{A[k]x[k]}{x[k-1]} - 1) \\ B[k] \end{bmatrix} \tag{7}$$

According to the Theorem 1 in [19], if the sampling period T_s satisfies $0 < T_s < 2/\alpha$, we can get that the state estimation error in Equation (5) will converge to a small neighborhood of zero and also the $\hat{W}_\zeta$ will converge to its optimal value W_ζ^*. From these two results, we can obtain that the RBF network $\hat{W}^T S(\varphi_\zeta)$ can be used to locally-accurately approximate the unknown internal dynamics $f(x;p)$, i.e.,

$$f(x[k];p) = \bar{W}^T S(x[k]) + \epsilon_2, \tag{8}$$

where

$$\bar{W} = \frac{1}{k_b - k_a + 1} \sum_{k=k_a}^{k_b} \hat{W}[k], \tag{9}$$

and $[k_a, ..., k_b]$ represents a time segment after the transient process.

IV. THE INCHWORM SOFT ROBOT

The robot presented in this paper (shown in Fig 1) is a cable-driven switching legged inchworm soft robot, which is an updated/improved version from our previous work [18]. The main body of the robot is formed by a highly deformable material, silicone rubber, combined with a switching legged mechanism to enable locomotion of the soft robot.

As shown in the picture, the silicone body is separated into two parts. The middle pink part is the main deformable component of the robot. It is shaped into a diamond grid in order to provide the desired recovering force. The two ends of the robot body are formed by the white silicone rubber which has lower stiffness compare to the middle part.

The four motors on top of the body control the two strings and two legs. The two strings link the two ends of the robot and are controlled by the two string motors, one for the left string and the other for the right string. The other two motors control the friction between the legs and the ground plane. This is achieved by using two 3d printed plastic legs that have low friction compared to the silicone rubber body. The leg motors can bring the legs into hidden or extended position. During the hidden position, the rubber body will directly contact with the ground causing high friction. While in extended position, the plastic leg will extend out and lift up the body, creating a low friction contact.

To achieve the locomotion of the robot, the motors need to be actuated in a coordinated manner. For example, to obtain the forward motion the following four steps are performed:

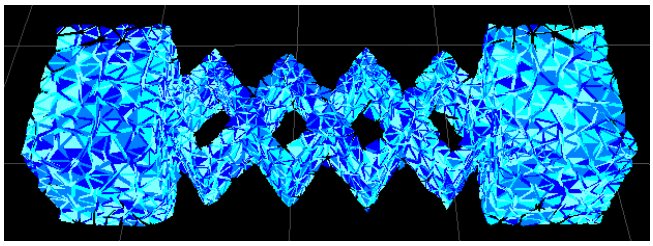

Fig. 2: Tetrahedral meshed 3D body.

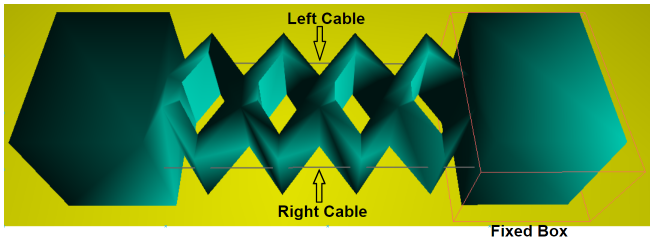

Fig. 3: Main robot body (green) with strings (the gray lines across the body) and the fixed box (red box, simulate the leg friction).

1) Set the front legs to hidden position and the rear legs to extended position.
2) Pull the bottom cable to shrink the body until the vertical bending angle reaches a value θ.
3) Set the rear legs to hidden position and the front legs to extended position.
4) Release the bottom cable to return the body to straight.

V. SIMULATION SETUP

The simulation software used during this research is called SOFA [27]. SOFA is a physics-based simulation software that mainly aims for soft material simulation, which can be used to study how soft materials act under different conditions.

In order to create a movable soft robot in SOFA, a 3d file of the soft body (with format .stl or .obj) needs to be imported to create a tetrahedral mesh model (Fig. 2). This mesh model determines the mesh quality and further affects the body action and computation speed.

Besides the main body, the robot also needs string actuators and fixed boxes for locomotion. The strings link the two ends of the robot (the gray line across the body in Fig. 3) and their length is variable. The fixed box is used to replace the legs in the real robot, and can fix part of the robot body inside the box to the space, which means, this part of the robot will not move from any force during the simulation.

To control the simulated robot, we can change the length of the strings and the position of fixed boxes. Specifically, the soft robot will be stretched along with the strings while the left and right strings distinguish the robot's stretch direction. The position of the fixed box can be switched between the front and backside of the robot, determining which side of the robot will be fixed to the ground.

For example, as for one complete forward motion: start with the fixed box at the front side of the robot and both strings are released. Next, reduce both string length to deform

the robot body. Then switch the fixed box position from front to back. Lastly, increase the string length to the original length. If both strings are shortened by the same amount, the robot will move forward by a distance that is proportional to the strings length reduction. If the strings are shortened by different amounts, the robot will perform a turning motion. The turning angle is also proportional to the difference of the two string length reduction.

The SOFA simulation can be commanded through Python scripts. In particular, we were able to command the desired string length and fixed box position, and extract the position and orientation information for further model learning purpose.

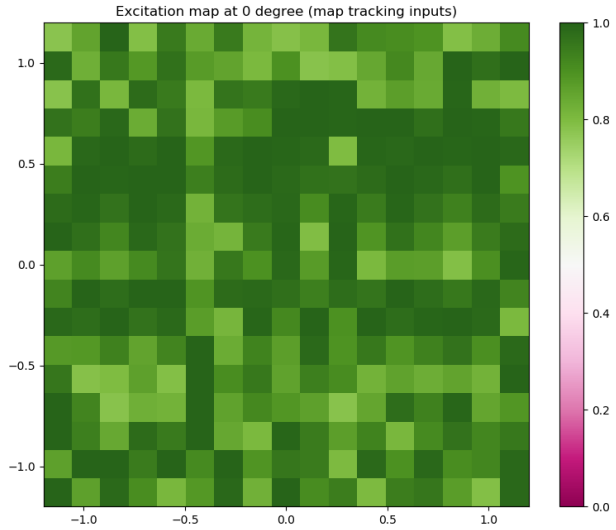

Fig. 4: Excitation levels for neural network nodes at orientation $\theta = 0$ degree for 3000 steps of map-tracking-input trajectory. x and y axis are first two dimensions of the neural network

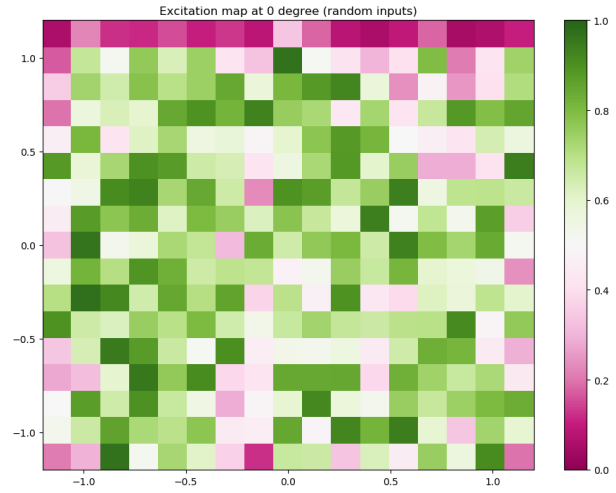

Fig. 5: Excitation levels for neural network nodes at orientation $\theta = 0$ degree for 3000 steps of random-input trajectory.

VI. ExPERIMENT

The goal of our experimental campaign is to learn the dynamic model of the simulated soft robot, and then use this learned model to predict the motion of robot and compare our results with other machine learning methods.

In this section, we will first explain the setup of the robot system, the learning algorithm, and the experiment. Then, the experimental results and comparison results will be presented.

A. Data Collection

As the robot is driven by two cables and the switching legs (represented by the fixed box in SOFA simulation), the motion would be chaotic and inefficient if these components do not cooperate. To this end, during our experiments, for every step the robot is set to move in the coordinated manner described below:

1) place the fixed box at the front side of the robot;
2) decrease the string length by l_1 and l_2 respectively;
3) place the fixed box at the rear side of the robot;
4) increasing the string length by l_1 and l_2 (bringing the robot back to its original shape);

where l_1 and l_2 are the controlled variables that evolve with the motion of the robot.

Considering that the robot is moving on a 2D plane, we use the central position of the robot in Cartesian coordinates (y_1, y_2), and the orientation (θ) of the robot as the system states. These three states will be collected at the beginning of each step. With the two inputs introduced previously (i.e., l_1, l_2) and these three position states (y_1, y_2, θ), the discretized motion dynamics of the robot are described as the following equations:

$$\begin{aligned} y_1[k+1] &= y_1[k] + T_s f_1(l_1, l_2, \theta) \\ y_2[k+1] &= y_2[k] + T_s f_2(l_1, l_2, \theta) \\ \theta[k+1] &= \theta[k] + T_s f_3(l_1, l_2, \theta) \end{aligned} \tag{10}$$

where k represents the steps of motion of the robot. In these equations, the robot position states (y_1, y_2) are not included in the unknown nonlinear function $f(l_1, l_2, \theta)$, because the robot's motion direction is independent of the current/past position.

During the data collection stage, to ensure that the collected data is sufficient, there should be enough samples to cover the whole neural network space. To this end, we can check the excitation level of the neural network nodes. During the DL learning process, at every step of the trajectory we will compute the excitation level of all the nodes by using the Gaussian radial basis function. After the whole trajectory has been studied, the excitation values will be compared to find out which nodes have not been well excited. By using this strategy, the program can track the excitation map during the robot motion and find the proper input for the next step, so that it can quickly fill the whole excitation map.

In addition, we also collected four random-input trajectories for testing purpose. Fig. 4 and 5 show the efficacy of the map-tracking-input trajectory with respect to the random-input trajectory. 3000 steps are collected for both types of trajectory and the computed excitation values are presented in

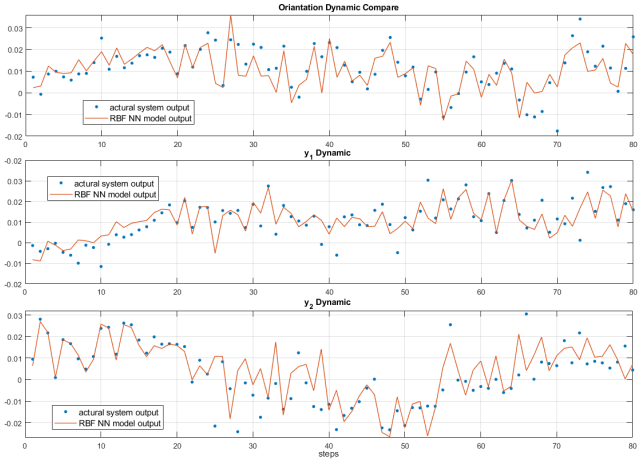

Fig. 6: The comparison between the actual system dynamics (blue dot) and the dynamics from the DL RBFNN model (red line), by using 80 steps random-input testing trajectory.

the figures. The results show that, the random-input trajectory fills 57% of the map while the map-tracking-input trajectory fills 75%, outperforming the random-input trajectory.

B. Neural Network Training Setup

To implement the discrete-time deterministic learning algorithm for training the neural network model of the robotic system, we collected 4 map-tracking-input trajectories and 4 random-input trajectories with an average of 4000 steps each. During the DL training, the RBF neural network has the number of the nodes $N = 4913$, the centers μ_i are evenly spaced on $[-1.2 : 0.15 : 1.2] \times [-1.2 : 0.15 : 1.2] \times [-1.2 : 0.15 : 1.2]$. The design constant a in Equations (3) and (4) is set to 0.5, α is set to 1.5, $P = diag\{10, 10, 10, 10\}$ which leads to $\lambda_{max}(P) = 10$. The T_s is set to 1, as the robot system is discretized based on the steps instead of the time. All the trajectories are repeated for 300 times during the learning process. All the data, for learning and testing, are normalized to values between -1.2 to 1.2 (match with the neural network space). The minimum value for the input l_1, l_2 is when the strings are at rest ($l_1 = l_2 = 0mm$), while the maximum input ($l_1 = l_2 = 30mm$) is when the string lengths are reduced by about $1/4$ of the robots body length (body length is about $120mm$). y_1, y_2 range about 10 times the robot's body length which is the robot's reachable area. The orientation θ ranges from $-180°$ to $180°$.

C. NN Weight Combination

After the learning process, every trajectory generated a set of weights. To combine all dynamic information together, we follow two steps from Equation (11) in [28]. First, we need to find out all the neural network nodes' highest excitation levels during each trajectory. The second step is to use these excitation values as the ratio to average the weight across the different trajectories. This leads to the following NN weights combination:

$$W_j = \begin{cases} \sum_{l=1}^{L} \frac{k_j^l}{k_j} \bar{W}_j^l, & \text{if } k_j \neq 0 \\ 0, & \text{if } k_j = 0 \end{cases} \tag{11}$$

where W_j is the final/combined weight value of the jth node. $\bar{W}_j^l$ is the weight value of the jth node for the lth trajectory ($l = \{1, ..., L\}$, where L =4 in this experiment) resulted from using Equation (9). $k_j^l = \max\{s(l_1, l_2, \theta)\}$ denotes the maximum excitation level and $S(x)$ is Gaussian radial basis function as describe in Sec. II, and $k_j = \sum_{l=1}^{L} k_j^l$.

With the fully estimated dynamic information, we can then produce the approximated dynamics from the neural network with combined weights. Based on Equations (2) and (3), the system dynamics can be computed by the neural network as $f(x[k]; p) = W^T S(x[k])$. While the states can be estimated as:

$$\begin{aligned} \bar{x}[k+1] = \bar{x}[k] + T_s B(\bar{x}[k] - x[k]) \\ + T_s (W)^T S(x[k]), \end{aligned} \tag{12}$$

where $\bar{x}[k]$ is the state of the estimator, $x[k]$ is the real state of the testing trajectory, $B = diag\{b_1, ..., b_n\}$ is a diagonal matrix which holds constant values for all estimators and satisfies $-1/T_s < b_i < 0$. In this experiment, the B is set to $diag[0.8, 0.8, 0.8]$. By combing all the weights learned through the map-tracking-input trajectories into one final-set of weights, we are able convey all the information into the data-driven model of the soft robot.

D. Results

Four random-input trajectories with 80, 141, 619, 7649 steps were used for testing and comparison purpose. We verified the performance of the modeling algorithm by comparing the real system data and the predicted outputs from the trained neural network model. The results for the first test trajectory (80 steps) are plotted in Fig. 6. The blue dots represents the real system data trajectory. The red line represents the predicted NN model output. The plot shows that the obtained NN model approximates well over the real robot dynamics.

We also compared learning performance of our algorithm with that of other existing machine learning algorithms. Five embedded functions from Matlab Machine Learning toolbox were used: 1) Regression Trees; 2) Regression Tree Ensembles; 3) Gaussian Process Regression; 4) Support vector Machine Regression (SVM) with Gaussian kernel; 5) Support vector Machine Regression (SVM) with linear kernel. TABLE I shows the overall results of the comparison between the six modeling algorithms. The first column indicates the four different testing trajectories with the number of steps. The second column indicates the three states of the system (orientation θ and position y_1, y_2). The error values in the table are computed as the mean root square error $E = \sqrt{\sum(\bar{f}(x) - (x_{k+1} - x_k))^2/k}$, where $\bar{f}(x)$ is the dynamics predicted by the learned models, while x_{k+1}, x_k are collected from SOFA.

Based on the results, we can see that our DL model and Model 4 have the best performance. In absolute values, the root mean square error on the orientation is about 5 deg, and 5 mm on the y_1 and y_2 coordinates. Normalizing these errors with respect to the range of the θ, y_1, y_2, the orientation error is more significant with respect to the position error.

TABLE I: Learning performance comparison among six different machine learning algorithms, including the proposed DL algorithms and five embedded Matlab Machine Learning Functions of 1) Regression Trees, 2) Regression Tree Ensembles, 3) Gaussian Process Regression, 4) Support vector Machine Regression (SVM) with Gaussian kernel, 5) Support vector Machine Regression (SVM) with linear kernel. The lowest value for each row is highlighted.

			Matlab Machine Learning Function				
Test Trajectory	State	DL Model	Model 1	Model 2	Model 3	Model 4	Model 5
Traj. 1 step = 80	θ	0.03464	0.0572	0.0823	0.0331	0.0322	0.0367
	y_1	0.00597	0.0142	0.0104	0.0057	0.0051	0.0109
	y_2	0.00976	0.0131	0.0154	0.0095	0.0087	0.0147
Traj. 2 step = 141	θ	0.06711	0.0754	0.0758	0.0617	0.0634	0.0636
	y_1	0.02069	0.0254	0.0259	0.0206	0.0206	0.0288
	y_2	0.02567	0.0277	0.0277	0.0247	0.0243	0.0343
Traj. 3 step = 619	θ	0.05867	0.0745	0.0874	0.0595	0.0592	0.0624
	y_1	0.02011	0.0283	0.0254	0.0207	0.0207	0.0311
	y_2	0.01861	0.0246	0.023	0.0189	0.0188	0.0326
Traj. 4 step = 7649	θ	0.05873	0.0793	0.09	0.0589	0.0588	0.0624
	y_1	0.02045	0.031	0.0248	0.0205	0.0205	0.0303
	y_2	0.01937	0.0254	0.0239	0.0194	0.0192	0.0336

However, those values are reasonable during the motion of a soft robot.

VII. CONCLUSIONS

This paper presented a novel model learning approach for motion dynamics modeling of a string-driven soft inchworm robot. Through a soft robot simulation software, we collected robot motion data that were used to train a radial basis function neural network. The network was then used to predict the soft robot locomotion. In the experimental section, we have shown that the resulting estimated NN model approximates well the dynamics of the real system.

REFERENCES

[1] 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.

[2] N. G. Cheng, M. B. Lobovsky, S. J. Keating, A. M. Setapen, K. I. Gero, A. E. Hosoi, and K. D. Iagnemma, "Design and analysis of a robust, low-cost, highly articulated manipulator enabled by jamming of granular media," in *2012 IEEE international conference on robotics and automation*. IEEE, 2012, pp. 4328–4333.

[3] Z. Wang, Y. Torigoe, and S. Hirai, "A prestressed soft gripper: design, modeling, fabrication, and tests for food handling," *IEEE Robotics and Automation Letters*, vol. 2, no. 4, pp. 1909–1916, 2017.

[4] C. Li, X. Gu, and H. Ren, "A cable-driven flexible robotic grasper with lego-like modular and reconfigurable joints," *IEEE/ASME Transactions on Mechatronics*, vol. 22, no. 6, pp. 2757–2767, 2017.

[5] V. Slesarenko, S. Engelkemier, P. I. Galich, D. Vladimirsky, G. Klein, and S. Rudykh, "Strategies to control performance of 3d-printed, cable-driven soft polymer actuators: From simple architectures to gripper prototype," *Polymers*, vol. 10, no. 8, p. 846, 2018.

[6] 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.

[7] H. Jin, E. Dong, G. Alici, S. Mao, X. Min, C. Liu, K. Low, and J. Yang, "A starfish robot based on soft and smart modular structure (sms) actuated by sma wires," *Bioinspiration & biomimetics*, vol. 11, no. 5, p. 056012, 2016.

[8] S. Seok, C. D. Onal, R. Wood, D. Rus, and S. Kim, "Peristaltic locomotion with antagonistic actuators in soft robotics," in *2010 IEEE international conference on robotics and automation*. IEEE, 2010, pp. 1228–1233.

[9] H.-T. Lin, G. G. Leisk, and B. Trimmer, "Goqbot: a caterpillar-inspired soft-bodied rolling robot," *Bioinspiration & biomimetics*, vol. 6, no. 2, p. 026007, 2011.

[10] R. F. Shepherd, F. Ilievski, W. Choi, S. A. Morin, A. A. Stokes, A. D. Mazzeo, X. Chen, M. Wang, and G. M. Whitesides, "Multigait soft robot," *Proceedings of the national academy of sciences*, vol. 108, no. 51, pp. 20 400–20 403, 2011.

[11] Y. Almubarak and Y. Tadesse, "Twisted and coiled polymer (tcp) muscles embedded in silicone elastomer for use in soft robot," *International Journal of Intelligent Robotics and Applications*, vol. 1, no. 3, pp. 352–368, 2017.

[12] Z. Deng, M. Stommel, and W. Xu, "A novel soft machine table for manipulation of delicate objects inspired by caterpillar locomotion," *IEEE/ASME Transactions on Mechatronics*, vol. 21, no. 3, pp. 1702–1710, 2016.

[13] D. Nguyen-Tuong and J. Peters, "Model learning for robot control: a survey," *Cognitive processing*, vol. 12, no. 4, pp. 319–340, 2011.

[14] Z. Y. Ding, J. Y. Loo, V. M. Baskaran, S. G. Nurzaman, and C. P. Tan, "Predictive uncertainty estimation using deep learning for soft robot multimodal sensing," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 951–957, 2021.

[15] T. G. Thuruthel, B. Shih, C. Laschi, and M. T. Tolley, "Soft robot perception using embedded soft sensors and recurrent neural networks," *Science Robotics*, vol. 4, no. 26, 2019.

[16] R. F. Reinhart and J. J. Steil, "Hybrid mechanical and data-driven modeling improves inverse kinematic control of a soft robot," *Procedia Technology*, vol. 26, pp. 12–19, 2016.

[17] J. F. Queißer, K. Neumann, M. Rolf, R. F. Reinhart, and J. J. Steil, "An active compliant control mode for interaction with a pneumatic soft robot," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 573–579.

[18] X. Chen, P. Stegagno, and C. Yuan, "A cable-driven switching-legged inchworm soft robot: design and testing," in *Proceedings of the American Control Conference*, 2021.

[19] C. Yuan and C. Wang, "Design and performance analysis of deterministic learning of sampled-data nonlinear systems," *Science China Information Sciences*, vol. 57, no. 3, pp. 1–18, 2014.

[20] ——, "Performance of deterministic learning in noisy environments," *Neurocomputing*, vol. 78, no. 1, pp. 72–82, 2012.

[21] C. Wang and D. J. Hill, "Learning from neural control," *IEEE Transactions on Neural Networks*, vol. 17, no. 1, pp. 130–146, 2006.

[22] ——, "Deterministic learning and rapid dynamical pattern recognition," *IEEE Transactions on Neural Networks*, vol. 18, no. 3, pp. 617–630, 2007.

[23] J. Park and I. W. Sandberg, "Approximation and radial-basis-function networks," *Neural computation*, vol. 5, no. 2, pp. 305–316, 1993.

[24] ——, "Universal approximation using radial-basis-function networks," *Neural computation*, vol. 3, no. 2, pp. 246–257, 1991.

[25] D. Gorinevsky, "On the persistency of excitation in radial basis function network identification of nonlinear systems," *IEEE Transactions on Neural Networks*, vol. 6, no. 5, pp. 1237–1244, 1995.

[26] C. Wang and D. J. Hill, *Deterministic learning theory for identification, recognition, and control*. CRC Press, 2009.

[27] J. Allard, S. Cotin, F. Faure, P.-J. Bensoussan, F. Poyer, C. Duriez, H. Delingette, and L. Grisoni, "Sofa-an open source framework for medical simulation," in *MMVR 15-Medicine Meets Virtual Reality*, vol. 125. IOP Press, 2007, pp. 13–18.

[28] T. Chen, C. Wang, and D. J. Hill, "Rapid oscillation fault detection and isolation for distributed systems via deterministic learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 6, pp. 1187–1199, 2013.