Efficient Jacobian-Based Inverse Kinematics With Sim-to-Real Transfer of Soft Robots by Learning

Fang, Guoxin; Tian, Yingjun; Yang, Zhi Xin; Geraedts, Jo M.P.; Wang, Charlie C.L.

DOI
10.1109/TMECH.2022.3178303

Publication date
2022

Document Version
Final published version

Published in
IEEE/ASME Transactions on Mechatronics

Efficient Jacobian-Based Inverse Kinematics With Sim-to-Real Transfer of Soft Robots by Learning

Guoxin Fang, *Member, IEEE*, Yingjun Tian, Zhi-Xin Yang, *Member, IEEE*, Jo M. P. Geraedts, and Charlie C. L. Wang, *Senior Member, IEEE*

***Abstract*—This article presents an efficient learning-based method to solve the *inverse kinematic* (IK) problem on soft robots with highly nonlinear deformation. The major challenge of efficiently computing IK for such robots is due to the lack of analytical formulation for either forward or inverse kinematics. To address this challenge, we employ neural networks to learn both the mapping function of forward kinematics and also the Jacobian of this function. As a result, Jacobian-based iteration can be applied to solve the IK problem. A sim-to-real training transfer strategy is conducted to make this approach more practical. We first generate a large number of samples in a simulation environment for learning both the kinematic and the Jacobian networks of a soft robot design. Thereafter, a sim-to-real layer of differentiable neurons is employed to map the results of simulation to the physical hardware, where this sim-to-real layer can be learned from a very limited number of training samples generated on the hardware.**

***Index Terms*—Inverse kinematics (IKs), Jacobian, learning, sim-to-real, soft robots.**

I. Introduction

With the use of flexible material, soft robots have the ability to make a large deformation and interact safely with the environment [1], which leads to a broad range of applications, such as exoskeleton/wearable devices [2], soft manipulators [3], [4] and surgery assistance [5]. However, as a hyper-redundant system with high nonlinearity in both material property and geometric deformation, it is difficult to formulate an effective kinematic model for solving the control task. The analytical *forward kinematics* (FKs) solution only exists for specific designs with a relatively simple shape (e.g., [6] and [7]). For a general soft robot with complicated structures/shapes, efficiently computing its inverse kinematic (IK) solution remains a challenging problem. For soft robots with redundancy, fast and reliable IK solution is a very important means for improving the control precision and response frequency in practical tasks [8].

A. Related Work

To efficiently model the behavior of soft robotic systems (i.e., computing FK), both analytical formulation and numerical simulation were conducted in previous research. Those analytical solutions, based on the differential geometry [6], [7], [9] and the mechanics analysis [10], are difficult to be generalized for soft robots with a complex shape, where numerical simulation by the *finite element method* (FEM) is usually employed [11], [12]. Computational efficiency is a bottleneck of applying FEM in the IK computation, as the simulation needs to be repeatedly conducted to estimate the Jacobian [13]. To overcome this, a reduced model by voxel representation [14], or computing quasistatic equilibrium [15], is presented to accelerate. However, these methods can easily become nonrealistic after applying large rotational deformation. The geometry-oriented simulation pipeline [16] can precisely compute the deformation of a variety of soft robots even in large rotation, which is later extended into a general IK solver [17] by using the Jacobian-based iteration. A model reduction method is applied to further accelerate the numerical-based simulation [18]. However, it is still difficult to directly include the simulator in the loop of iteration and achieve fast IK computing.

The data-driven methods used in soft robotics are often treated as regression problems of machine learning where kinematic models can be effectively learned from datasets [8]. To enable the IK tasks on soft robots, an intuitive solution is to directly learn the mapping of IK, which takes the motion as the input of a network and generates the corresponding parameters of actuation as output [19]–[25]. However, this intuitive method does not perform well in a redundant system, as the one-to-many mapping from task space to actuator space is generally difficult to learn. Although this issue can be partly solved

Manuscript received 22 May 2021; revised 27 February 2022; accepted 9 May 2022. Date of publication 8 June 2022; date of current version 14 December 2022. Recommended by Technical Editor P.-C. Lin and Senior Editor X. Chen. This work was supported by the chair professor fund of Charlie.C.L. Wang provided by the University of Manchester. *(Corresponding author: Charlie C. L. Wang.)*

Guoxin Fang and Jo M. P. Geraedts are with the Faculty of Industrial Design Engineering, Delft University of Technology, 2628 Delft, The Netherlands (e-mail: g.fang-1@tudelft.nl; j.m.p.geraedts@tudelft.nl).

Yingjun Tian and Charlie C. L. Wang are with the Department of Mechanical, Aerospace and Civil Engineering, The University of Manchester, Manchester M13 9PL, U.K. (e-mail: yingjun.tian@postgrad.manchester.ac.uk; charlie.c.l.wang@gmail.com).

Zhi-Xin Yang is with the State Key Laboratory of Internet of Things for Smart City, Department of Electromechanical Engineering, University of Macau Macau 999078, China (e-mail: zxyang@um.edu.mo).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TMECH.2022.3178303.

Digital Object Identifier 10.1109/TMECH.2022.3178303

TABLE I
COMPARISON OF LEARNING-BASED METHODS FOR SOLVING IKS ON SOFT ROBOTS

Property	Learning-based methods		
	Learning for IK mapping [19]–[25]	Jacobian by FK network gradient [26]	Learning Jacobian and FK (our work)
Requirement on network-type	General	Analytically differentiable network	General
Smooth motion planning	Special extra effort needed [24], [25]	Yes	Yes
Target outside learning space	No	Yes	Yes
Converge speed	No iteration	Good	Good
Complexity of computation†	$O(hb)$	$O(hb^2)$‡	$O(hb)$

†We evaluate the complexity of network-based IK computing on networks with $O(h)$ hidden layers and $O(b)$ neurons per layer.
‡High complexity in [26] is caused by applying the chain rule to an FK network to obtain its Jacobian, which results in nested functions.

by setting constraints in the actuator space [24] or specifying the preference of configurations in the IK equation [25], we solve the problem by using a different method to combine learning with Jacobian-based IK. Our method is efficient when planning a smooth motion (i.e., by minimizing variation in the actuator space) for soft robot systems with redundancy. To reach a similar goal, Thuruthel *et al.* [27], [28] attempted to learn the differential IK model with local mapping. Another method was presented in [29] to estimate the soft robot's Jacobian by the Kalman filter approximation. Recently, Bern *et al.* [26] presented a method to effectively evaluate the Jacobian by using the gradients of the FK network, which however limits the type of network used for FK learning and requires more time to compute the Jacobian for determining IK solutions. In our work, both the mapping functions of FK and the Jacobian are learned by neural networks as explicit functions. This makes our method far more efficient. A comparison of three types of learning-based methods is given in Table I.

On the other hand, learning a kinematic model for soft robots usually needs a large number of samples, which can be very time-consuming when generating the data in a physical environment either by the motion capture system [4] or embedded sensors [30], [31]. Moreover, to explore the boundary of the workspace, a large extension in material under large actuation needs to be applied [32]. Soft materials on a robot can become fragile and might generate plastic deformation after repeating such deformation may times [4]. Consequently, the learned model becomes inaccurate. Furthermore, errors generated during the fabrication of a specimen can make the network learned on this specimen difficult to be used on other specimens with the same design. To reduce the cost of generating training data, Kubus *et al.* [33] reported a data-efficient method by exploiting structural properties of the kinematic mapping. Another solution is to generate the accurate dataset in the simulation environment, and then convert the model learned from simulation into physical reality by transfer learning [34]–[37]. The hybrid model contains the analytical formulation, and the network-based correction was conducted in [25] and [38], where more precise control of the soft robot was achieved. Similarly, FEM was used in [39] to generate a simulation dataset for training a hybrid kinematic model by transfer learning. A more efficient numerical simulator [17] is adopted in this work to generate the training dataset. When working together with the sim-to-real network, IK with high accuracy can be achieved. The comparison of IK with sim-to-real learning by using the reduced analytical model versus our simulation-based model can be found in Section IV-C.

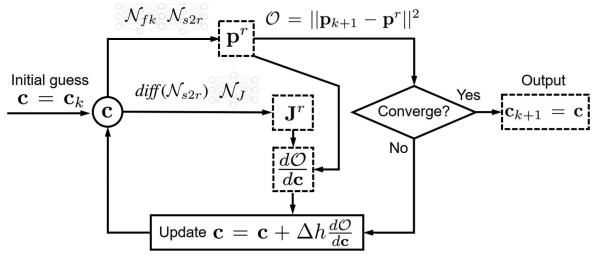


Fig. 1. Pipeline of the Jacobian-based method to determine the actuation parameters $\mathbf{c}_{k+1}$ for the waypoint $\mathbf{p}_{k+1} \in \mathcal{L}$ that solves the IK problem of soft robots by minimizing $\mathcal{O}(\cdot)$ in (1). Both the position $\mathbf{p}^r$ in task space and the Jacobian $\mathbf{J}^r$ are effectively estimated by the networks $\mathcal{N}_{fk}$, $\mathcal{N}_J$, and $\mathcal{N}_{s2r}$ obtained from the offline training. When $\mathcal{O} < \epsilon^2$, we regard the convergence as having been achieved (e.g., 0.1% of the workspace width is employed as ϵ).

B. Our Method

Three networks: 1) FKs $\mathcal{N}_{fk}$; 2) Jacobian $\mathcal{N}_J$; and 3) sim-to-real mapping $\mathcal{N}_{s2r}$, are trained to support the effective computing of IK for soft robots in both virtual and physical spaces at a fast speed. With an objective function defined in quadratic form and the network-based efficient estimation of FK and Jacobian, the Jacobian-based iteration is used to compute the IK solution. The pipeline of our method is shown in Fig. 1 with detail discussed in Section II.

The technical contributions of our work are as follows:

1) A direct pipeline for learning both the FK and Jacobian from accurate numerical simulation results, to support effective IK computing for soft robots. This method can compute IK solutions in a fast speed and has the capability to plan a smooth motion for soft robotic systems with redundancy.
2) A two-step learning strategy by using the sim-to-real transfer learning to eliminate the gap between the prediction based on simulation and the physical behavior, which can greatly reduce the required amount of empirical data when compared to directly learning a predictor from the physical experiment.

The behavior of our method has been verified on two hardware setups of soft robots giving 2-D and 3-D motions. The effectiveness of our method is quantitatively evaluated and compared with other approaches in the IK tasks of soft robots. Experimental tests are also conducted to demonstrate the performance of our method on soft robots with the same design, but fabricated with different materials.

II. Jacobian-Based Kinematics and Learning

In this article, we focus on solving the IK problem for soft robots—specifically, to determine the parameters of actuation that can drive a soft robot to reach a target position/shape. As the analytical IK solution cannot be obtained, we adopt a Jacobian-based numerical method where a target-oriented objective function $\mathcal{O}(\cdot)$ is minimized to determine the parameters of actuation. In this section, we first introduce the Jacobian-based IK computation for the path-following task. After this, we demonstrate how it can be solved practically by applying the training in a virtual environment and then the sim-to-real transformation.

A. Jacobian-Based IK Solution

The path-following problem of a soft robot is described as driving a marker on its end-effector to move along a path $\mathcal{L}$ presented by a set of target waypoints $\{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_i, \mathbf{p}_{i+1}, \ldots\}$ in the task space. For each waypoint $\mathbf{p}_i$ to be reached by the marker, numerical computation of IKs attempts to minimize the distance between $\mathbf{p}_i$ and the marker's position. This is formulated as an optimization problem

$$\mathbf{c}_i = \underset{\mathbf{c}}{\operatorname{argmin}} \mathcal{O}(\mathbf{p}_i, \mathbf{c}) = \underset{\mathbf{c}}{\operatorname{argmin}} \|\mathbf{p}_i - \mathbf{p}(\mathbf{c})\|^2 \tag{1}$$

where $\mathbf{p}(\cdot) \in \mathbb{R}^n$ denotes the FK function to compute the position of the marker, the input of $\mathbf{p}(\cdot)$ is a vector of actuation parameters $\mathbf{c} = (c_1, c_2, \ldots, c_m) \in \mathbb{R}^m$, and n and m are the dimensions of the task space and the actuator space, respectively.

To find the solution of (1), the gradient of $\mathcal{O}(\cdot)$ is

$$\frac{d\mathcal{O}}{d\mathbf{c}} = -2(\mathbf{p}_i - \mathbf{p}(\mathbf{c}))\mathbf{J}(\mathbf{c}) \tag{2}$$

will be employed to update the value of $\mathbf{c}$ with $\mathbf{J}(\mathbf{c}) = d\mathbf{p}/d\mathbf{c} \in \mathbb{R}^{n\times m}$ being the Jacobian matrix that describes the moving trend of a soft robot's body at certain actuation parameters. The value of $\mathbf{c}$ is updated by $\mathbf{c} = \mathbf{c} + \Delta h \frac{d\mathcal{O}}{d\mathbf{c}}$, where Δh is a step size to minimize the value of $\mathcal{O}(\cdot)$ along the gradient direction, which can be determined by soft linear search [17]. Fig. 1 shows the illustration of this algorithm.

When a physics-based simulation is employed to evaluate the FK function $\mathbf{p}(\cdot)$, the Jacobian matrix $\mathbf{J}$ can be obtained by numerical difference [17], [39]. The kth column of $\mathbf{J}$ is computed as

$$\mathbf{J}_k = \frac{\partial \mathbf{p}(\mathbf{c})}{\partial c_k} \approx \frac{\mathbf{p}(\ldots, c_k + \Delta c, \ldots) - \mathbf{p}(\ldots, c_k - \Delta c, \ldots)}{2\Delta c} \tag{3}$$

where Δc is a small constant determined according to experiments and assigned as $1/10$ N of the actuation range, where N is the number of samples for each actuation parameter presented in Section III-B. Notice that it can be time-consuming to evaluate the values of $\mathbf{p}(\cdot)$ and $\mathbf{J}(\cdot)$ by physics-based simulation in IK computing. We therefore introduce a learning-based method to learn both the FK and the Jacobian model in the offline stage, which can support a fast IK computing during online usage. In the meantime, the difference between the simulation and the physical behavior is fixed by the sim-to-real transfer learning.

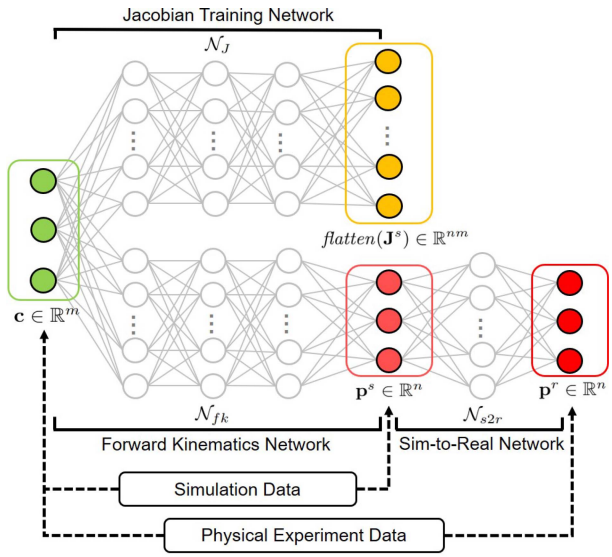


Fig. 2. Network structure used in our approach to train the kinematic model and the sim-to-real transfer.

B. Learning-Based Model for IK Computing

We learn both the FK model and its Jacobian from simulations,- denoted by $\mathbf{p}^s(\cdot)$ and $\mathbf{J}^s(\cdot)$, respectively, which are transferred to physical hardware by learning a sim-to-real mapping function $\mathbf{r}(\cdot)$. Denoting the location of a traced marker on physical hardware as $\mathbf{p}^r$, the function of sim-to-real mapping is required to have $\mathbf{r}(\mathbf{p}^s) \approx \mathbf{p}^r$. Neural networks are employed to learn these functions (see the architecture of neural networks shown in Fig. 2).

In the simulation environment, $\mathbf{p}^s(\cdot)$ and $\mathbf{J}^s(\cdot)$ are trained on two networks: 1) $\mathcal{N}_{fk}$; and 2) $\mathcal{N}_J$, by spanning the workspace of actuators with a large number of samples. Note that the output layer for $\mathcal{N}_J$ is a column vector as the flattened Jacobian matrix $\mathbf{J}^s$. After obtaining the network $\mathcal{N}_{fk}$, the sim-to-real mapping function $\mathbf{r}(\cdot)$ is trained on a differentiable network $\mathcal{N}_{s2r}$ by using a few samples obtained from a physical experiment conducted on the hardware setup.

With the help of these trained networks, we can estimate the Jacobian on the hardware setup as

$$\mathbf{J}^r(\mathbf{c}) = \frac{d\mathbf{p}^r}{d\mathbf{p}^s}\frac{d\mathbf{p}^s}{d\mathbf{c}} \approx \text{diff}(\mathcal{N}_{s2r})\mathbf{J}^s(\mathbf{c}). \tag{4}$$

Considering the difficulty of data acquisition on hardware specimens, the *feedforward neuronal network* (FNN) with a single layer of fully connected neurons is adopted in our implementation for $\mathcal{N}_{s2r}$. The differentiation $\text{diff}(\mathcal{N}_{s2r})$ as an $n \times n$ matrix can be computed analytically by differentiating the network's activation functions. As most of the complexity in kinematics can be effectively captured by $\mathcal{N}_{fk}$ and $\mathcal{N}_J$, a lightweight network $\mathcal{N}_{s2r}$ trained by a small dataset obtained from the physical experiment can already show very good performance on eliminating the inconsistency in material properties and fabrication.

Through this learning-based model, the gradient of the IK objective function in the physical environment can then be

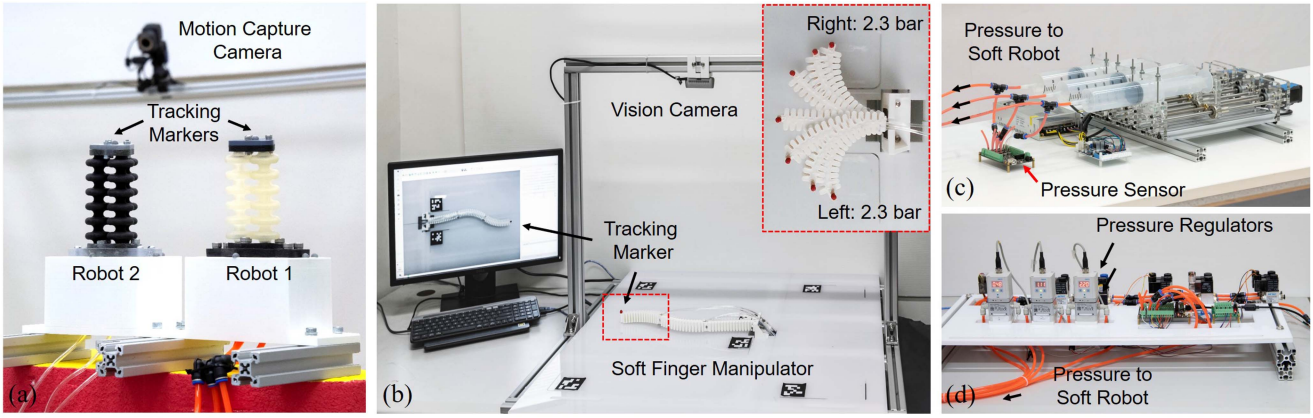


Fig. 3. Two hardware setups employed in our experiments to collect data and verify the performance of our method.(a) Soft actuator with multiple chambers, (c) which are actuated by an array of syringes. (b) Three connected soft fingers, (d) which can be actuated individually by proportional pressure regulators.

computed by

$$\frac{d\mathcal{O}}{d\mathbf{c}} = -2(\mathbf{p}_i - \mathbf{p}^r(\mathbf{c}))\mathbf{J}^r(\mathbf{c}) \tag{5}$$

$$\approx -2(\mathbf{p}_i - \mathbf{r}(\mathbf{p}^s(\mathbf{c})))\text{diff}(\mathcal{N}_{s2r})\mathbf{J}^s(\mathbf{c}). \tag{6}$$

Note that the real positions of markers, $\mathbf{p}^r(\mathbf{c})$ in (5), can also be obtained from a hardware setup (e.g., by a motion-capture system). However, using positions predicted by $\mathcal{N}_{fk}$ and $\mathcal{N}_{s2r}$ networks can avoid physically actuating the hardware inside the loop of numerical iteration. After training the networks $\mathcal{N}_{fk}$, $\mathcal{N}_J$ and $\mathcal{N}_{s2r}$, an iteration-based algorithm (as shown in Fig. 1) is used to effectively solve (1). The actuation parameters $\mathbf{c_{i-1}}$ for realizing $\mathbf{p}_{i-1}$ are employed as the initial guesses when computing $\mathbf{c_i}$ for $\mathbf{p}_i$. As a result, the iteration converges rapidly and the continuity of motion in configuration space can be preserved.

III. Data Generation and Training

We first present two hardware setups that are used in our research to verify the performance of the abovementioned learning-based method. After introducing the steps for generating datasets, the training details are provided.

A. Soft Robotic Hardware

Two hardware setups are constructed to investigate the performance of our IK solver. Both setups are equipped with cameras to capture the real positions of markers for the purpose of training and verification.

1) Actuator With 3-D Motion: The first setup is a 3-D printed soft actuator with three chambers that can be actuated individually [4]. Its soft body can extend and bend in a 3-D task space. To verify the behavior of our sim-to-real method, two specimens are fabricated by the same Object350 Connex 3-D printer but using slightly different materials—the Agilus black and Agilus transparent materials. Both have the softness 70 A according to their factory specification. These two models are shown as Robots 1 and 2 in Fig. 3(a). The soft robot is actuated by an array of syringes that has closed-loop control with the help of pressure sensors, as shown in Fig. 3(c). For this setup, we have the same dimension for the workspace ($m = 3$) and the actuator space ($n = 3$).

2) Planar Finger Manipulator: The second setup is a soft manipulator that can move in the xy-plane [see Fig. 3(b)]. The manipulator contains three soft finger sections that are rigidly connected. We use Festo Pressure Regular VPPE-3-1/8-6-010 to provide the pressure for each section [see Fig. 3(d)]. Every finger section contains dual chambers that can bend symmetrically for both sides up to 120°. To maximize the deformation of each finger section, we only actuate one side for a segment, each time with the pressed air in the range of $[0, 3]$ bar. When considering both sides of a segment, this results in a range of $[-3, 3]$ as actuation,- i.e., "+" for actuating the chamber at one side and "−" for the other side. This is a redundant system with the dimension $n = 2$ for the workspace and $m = 3$ in the actuator space.

B. Data Generation on Simulator

In our work, FKs of soft robots in a virtual environment is computed by a geometry-oriented simulator [16], [17]. When employ this simulator to generate datasets for training the FK network $\mathcal{N}_{fk}$ and the Jacobian network $\mathcal{N}_J$, the computation time for generating single sample point is 4.3 s (the three-chamber robot) and 1.2 s (the finger manipulator), respectively. It is worth mentioning that as a general training pipeline, the dataset used to train $\mathcal{N}_{fk}$ and $\mathcal{N}_J$ can be generated by different kinematic models,- e.g., those analytically computed by piecewise constant curvature [3] or numerically by the FEM software, such as Abaqus. Here, we choose the geometry-based simulator, as it can further reduce the cost of data generation than FEM. Moreover, it needs to capture less physical data than the analytical model for training the sim-to-real network (discussed in Section IV-C).

We now present the sampling method in actuator space for generating training data points. We uniformly divide the pressure range of each actuator into N segments to make sure that the distance between sample points is less than 1% of the workspace width. Sampling results of the two hardware setups are shown

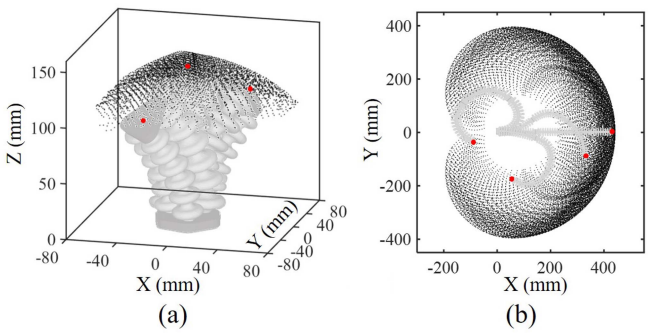


Fig. 4. Results of the simulation are employed as training samples (present in black dots) to learn the FK network $\mathcal{N}_{fk}$ and the Jacobian network $\mathcal{N}_J$, where these samples also span the workspace $\mathcal{P}^w$ of a robot. Red dots represent some example points as targets for motion in the workspace.

in Fig. 4, which also presents the workspaces $\mathcal{P}^w$ of these two soft robots. In our experiment, $N = 16$ and $N = 29$ are used for these two setups, respectively. This results in $16^3 = 4096$ samples for the three-chamber actuator [see Fig. 4(a)] and $29^3 = 24389$ samples for the finger manipulator [see Fig. 4(b)]. Notice that the difference in choosing the N value is due to the redundancy of the finger setup, which also has a larger range in actuation. Based on our tests, datasets selected in these sizes can already well-trained $\mathcal{N}_{fk}$ and $\mathcal{N}_J$ to capture the kinematic behavior of soft robots (see the results in the following section).

C. Data Generation on Hardware

For the purpose of training sim-to-real network $\mathcal{N}_{s2r}$, datasets are generated on two hardware setups. We uniformly span the actuator space to generate physical data, which are classified into the training (70%) and the test (30%) datasets. Since the efficiency of the training pipeline depends on the number of samples generated on hardware setups, we test and determine the appropriate sample number used to train $\mathcal{N}_{s2r}$—details are presented in Section III-D.

1) Actuator With 3-D Motion: To trace the 3-D motion of this soft actuator, we place a marker at the center of its top plane and several markers on its static base. The motion capture system, which contains eight Vicon Bonita ten cameras and ten Vicon Vantage five cameras, is used to capture the movements at the rate of 30 Hz. Because of the viscoelasticity of soft materials used to fabricate this robot, it takes a relatively long time for the position of a marker to become stable (i.e., less than 0.05-mm change between neighboring image frames). This makes the process of data collection more time-consuming than a robotic system with rigid bodies. As a result, the average time for collecting one sample in the physical environment is 4.0 s.

2) Planar Finger Manipulator: As only planar coordinates are needed when tracking the positions of a marker, we use a RealSense D435 camera mounted at the top of the setup. We place a red marker on the tip of the manipulator and adopt the OpenCV library as software to track the marker's position in the plane. QR code is employed to build the mapping between the coordinates in image space and the coordinates in the real-world. The speed of data acquisition for this system is 10 Hz. For this hardware setup, the average time for collecting one sample point is 3.5 s.

D. Details of Training

In this article, the type and structure of training models used in the experiment are carefully selected based on their performance. For training $\mathcal{N}_{fk}$ and $\mathcal{N}_J$, FNN and *long short-term memory* (LSTM) are tested, as they can both adequately capture the nonlinear behavior in a training dataset, including the many-to-one FK mapping for redundant systems. For the sim-to-real transfer network $\mathcal{N}_{s2r}$, it needs to be differentiable with analytic gradients. Meanwhile, it should be lightweight as only a limited number of samples can be obtained from physical experiments. For these reasons, single-layer FNN is selected for $\mathcal{N}_{s2r}$. All networks are trained by using the deep Llearning toolbox of MATLAB running on an NVIDIA GeForce RTX 2070 graphics card.

1) Training for FK and Jacobian: We first study the effectiveness of training $\mathcal{N}_{fk}$ and $\mathcal{N}_J$ by using a different number of layers and different numbers of neurons. Each dataset is divided into training and test subsets in the ratio of $70\% : 30\%$. For all networks, the activation function is set as tan–sigmoid

$$f(x) = \frac{2}{1 + e^{-2x}} - 1 \tag{7}$$

as it can well-fitted the nonlinearity in kinematic mapping. Moreover, it is differentiable and can provide a faster training speed. We set the batch size as 200, the maximum number of epochs as 13 500, and the learning rate as 0.04. The Levenberg–Marquardt backpropagation is employed for training.

The estimation errors for both soft robot setups are evaluated on the test datasets, as shown in Fig. 5, where we can find that FNN and LSTM can both converge to a good result after carefully tuning the network parameters. It is hard to find significant improvement in the accuracy by using network with feedback connections (i.e., LSTM). This is because the training data only contain quasistatic information of the system, where the time-related network structure cannot show its advantage. On the other hand, it is found that the structure of the network for learning the Jacobian $\mathcal{N}_J$ on a redundant system (i.e., the planar finger manipulator) needs to be selected more carefully. FNN is selected as the final network structure, and the best performance for training $\mathcal{N}_J$ is observed on this hardware setup when FNN with $h = 2$ hidden layers and $b = 64$ neurons per layer is employed to learn $\mathcal{N}_J$. Differently, FNN with $h = 3$ layers and $b = 30$ neurons per layer gives the best results in training $\mathcal{N}_{fk}$. The error of position prediction by using $\mathcal{N}_{fk}$ is less than 0.5 mm (i.e., 0.58% of the workspace's width). For the three-chamber actuator, the numbers of layers and neurons have less influence on the training result. For this setup, we select $h = 2$ and $b = 35$ for both networks, which results in a FK prediction with error less than 0.17 mm on the test dataset (i.e., 0.34% of the workspace's width). With such accurate predictions generated by $\mathcal{N}_{fk}$ and $\mathcal{N}_J$, we can obtain IK solutions efficiently and accurately (see the behavior study given in Section IV).

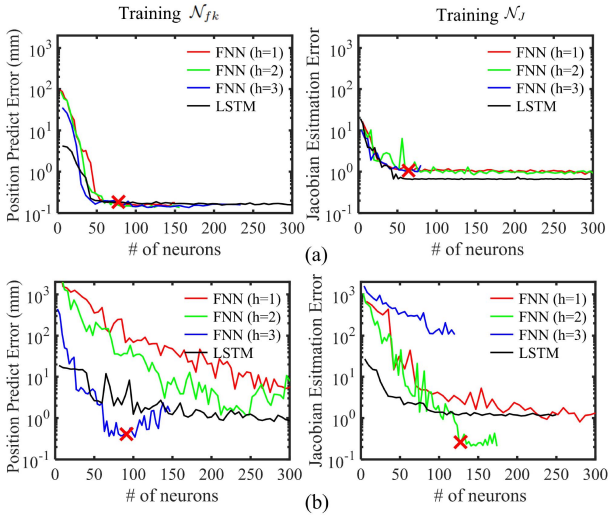


Fig. 5. Comparison of learning results by using a different number of layers as h and a different number of neurons b per layer (i.e., the total number of neurons in a network is hb). Tests are conducted on (a) robotic setup without redundancy (the three-chamber actuator with $m = n = 3$) versus (b) setup with redundancy (the planar finger manipulator having $m = 3$ and $n = 2$). Red crosses indicate the network parameters used in the physical experiment.

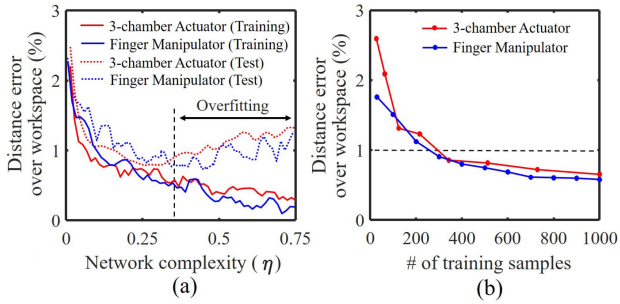


Fig. 6. Experimental study for the performance influence in the sim-to-real network $\mathcal{N}_{s2r}$ by using (a) different numbers of neurons; and (b) datasets in different sizes. With the properly selected complexity of network structure, the overfitting problem can be avoided. The distance-predict error can be controlled within 1% of the workspace width for both setups when a limited number of training samples are used [see part (b)].

2) Training for Sim-to-Real Transfer: When training for $\mathcal{N}_{s2r}$, an important parameter here is the number of neurons, which is selected as η times the number of samples to avoid overfitting on the training dataset. Fig. 6(a) shows the behavior on both the training dataset (denoted by the solid curves) and the test dataset (denoted by the dashed curves) when using different values of η. Based on the analysis, $\eta = 1/4$ is selected for our experiment to avoid overfitting.

As the time used to collect physical data points should be controlled, we also study the behavior of $\mathcal{N}_{s2r}$ with different numbers of training samples. For this purpose, the prediction errors as the ratios of the distance errors over the workspace widths are shown in Fig. 6(b) to study the effectiveness of using different numbers of samples. In these tests, the number of neurons is always assigned as $\eta = 1/4$ of the training samples. For both setups, we find that the network $\mathcal{N}_{s2r}$ can be

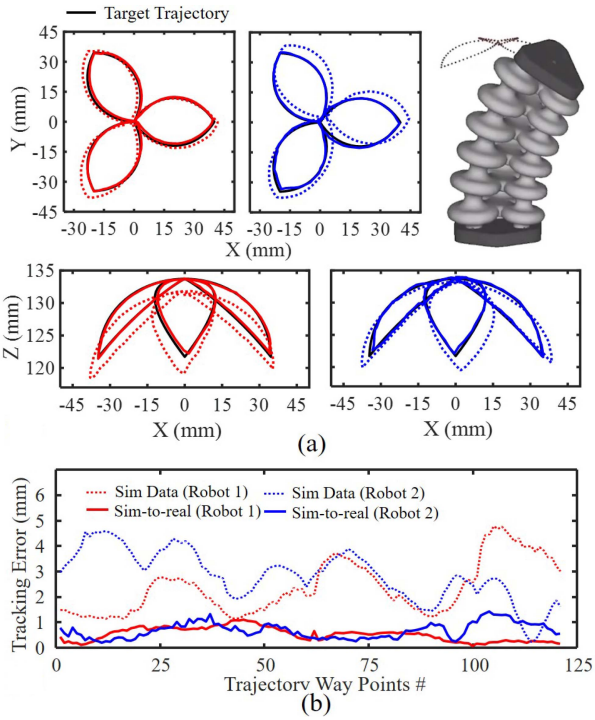


Fig. 7. Results of the path-following task on two soft robots with the same design, but fabricated with different materials [shown as robots 1 and 2 in Fig. 3(a)]. (a) Trajectory of the soft robots by applying IK solutions with (solid line) and without (dashed line) the sim-to-real network. (b) Visualized tracking errors on trajectory waypoints.

well-trained when using a limited number of training samples. In our implementation, 1% is selected as the threshold of accurate prediction, and this threshold is used to determine the number of samples for training $\mathcal{N}_{s2r}$. As a result, 343 samples are used for the three-chamber actuator and 620 samples are conducted for the finger manipulator. The datasets for training $\mathcal{N}_{s2r}$ on two hardware setups can both be collected within 30 min.

IV. Experiment Results

In this section, we present all the experiment results of IK computing for soft robots by using our learning-based Jacobian iteration. The results are generated in both the virtual and the physical environments. Computation of the learned neural networks in prediction is implemented in C++ and integrated into our control platform to gain the best computational efficiency. All the IK computations were efficiently run on a laptop PC with Intel i7-9750H 2.60 GHz CPU and 16 GB memory. Note that the prediction made by networks and the IK algorithm is entirely run on a CPU.

A. Path Following by Actuator With 3-D Motion

We first test the behavior of the learning-based IK computing method in the task of path following on the soft actuator with three chambers. Given 3-D trajectories of the "flower" shape (see Fig. 7) and the "box" shape (see Fig. 8), 120 and 240 waypoints are sampled on the paths in uniform distances, respectively. The proposed IK solver is then used to compute

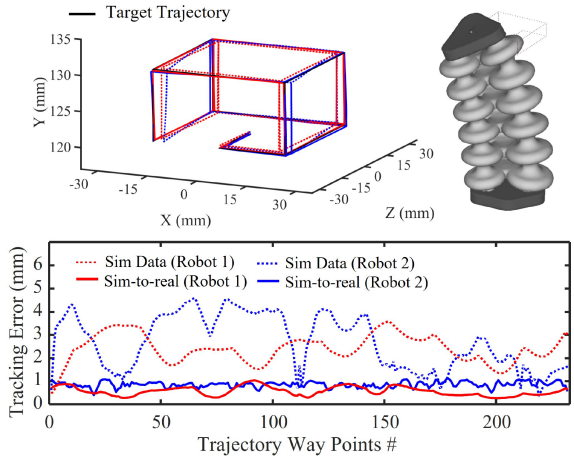


Fig. 8. Results of path following for the "box" shape trajectory demonstrate the vertical motion of three-chamber soft robot. It can also find that the maximal tracking errors are reduced from 4.5 to 1.0 mm for both robots after applying the sim-to-real network.

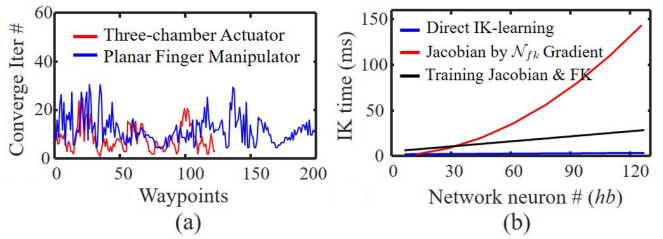


Fig. 9. Quantitative analysis for (a) speed of convergence; and (b) time efficiency of our Jacobian-based training method. Three learning strategies for computing IK are compared in part (b).

TABLE II
COMPUTING SPEED AND IK ACCURACY BY DIFFERENT METHODS

	Model-based solution		Learning-based method†	
	Analytical model [9]	Numerical model [17]	Direct IK learning [19]	This work
IK time	12.6 ms	138 s	3.2 ms	28 ms
Ave. error	8.2 mm	4.7 mm	0.97 mm	0.78 mm

†Same dataset was applied to learn the direct IK mapping [19] and sim-to-real network $\mathcal{N}_{s2r}$ in our pipeline.

the actuation parameters for each waypoint. This leads to the actuation sequence that can drives soft robot to move along the trajectories. When running in the simulation environment, the trained networks can generate actuation that results in very accurate trajectories with the average tracking error as 0.13 mm. In the physical environment, we learned the sim-to-real networks separately on two soft robots, as shown in Fig. 3(a). If we directly apply the actuation parameters obtained from IK computing in the simulation environment, the error of path following is high (i.e., up to 5 mm). At the same time, the variation caused by fabrication and material can be clearly observed from the difference between robots 1 and 2, as shown in Fig. 7. By incorporating the sim-to-real transfer in our method, we can successfully reduce the error in the physical environment to less than 1.2 mm for both robots [see Fig. 7(b)], which is 1.71% of the workspace width. For the tests shown in Fig. 8, the maximal errors are reduced from 4.5 to 1.0 mm. It is interesting to see that tests with circular and line trajectories shown similar tracking error. This is because the IK mapping from task space to actuator space is nonlinear.

Besides the accuracy, another advantage of our learning-based approach is its low computational cost. Thanks to the efficient forward propagation process of FNN networks and fast converge speed of the Jacobian-based algorithm, the time used to compute single waypoint IK for the three-chamber setup is less than 30 ms, even when a large number of neurons $hb = 128$ is used. As a final result, the time used to compute IK solutions for the entire "flower" and "box" are 2.53 and 4.12 s, respectively. The quantitative analysis of the converge speed and IK computing time of our algorithm is shown in Fig. 9, and also compared with other existing solutions (see Table II). It can be found that the learning-based method (for both analytical and numerical methods) can generally provide a more accurate IK result than the model-based solution, as it can well-captured the uncertainties that are hard to calibrate (e.g., material shifting, fabrication error, etc.). When compared between learning-based methods, direct IK learning is the fastest [see Fig. 9(b)]. Our Jacobian-based method provides the best accuracy and can also ensure a real-time computing speed (i.e., at the rate of 35+ waypoints per second). In addition, also as aforementioned in Section I-A, our solution can well-handled the redundant soft robots to ensure minimum variation when travelling along the trajectory. This is difficult to be handled by direct IK training.

B. Experiment With Soft Finger Manipulator

The soft finger manipulator, as shown in Fig. 3(b), is a redundant system, which has a higher degrees of freedom (DOFs) in its actuation space ($m = 3$) than the task space ($n = 2$). Therefore, an input waypoint can have multiple IK solutions. Both the path following and the interactive positioning tasks are conducted to validate the performance of our learning-based IK solver on the redundant systems.

1) Path Following: We first present the results of following an "8"-shaped trajectory that contains 200 waypoints, as shown in Fig. 10(a). The actuation parameters obtained from the Jacobian-based method are compared with those resulting from the direct IK-learning. Our Jacobian-based IK by learning demonstrates excellent performance in the accuracy of tracking precision. The average and maximum tracking errors for all waypoints are 0.08 and 0.18% of the workspace width, respectively. As shown in Fig. 10(c), our method is able to ensure a smooth motion that minimizes the variation in actuator space. As a comparison, large variation (i.e., jumps) in the actuator space can be found in the results of direct IK-learning [circled by dashed lines in Fig. 10(c)]. This problem of direct IK learning is mainly caused by its lack of capability to support the one-to-many IK mapping. For this trajectory, the IK solutions can be efficiently computed by our method at the average speed of 39 ms per waypoint.

It is worth mentioning that the configuration of motion computed by our method is highly dependent on the selection of the

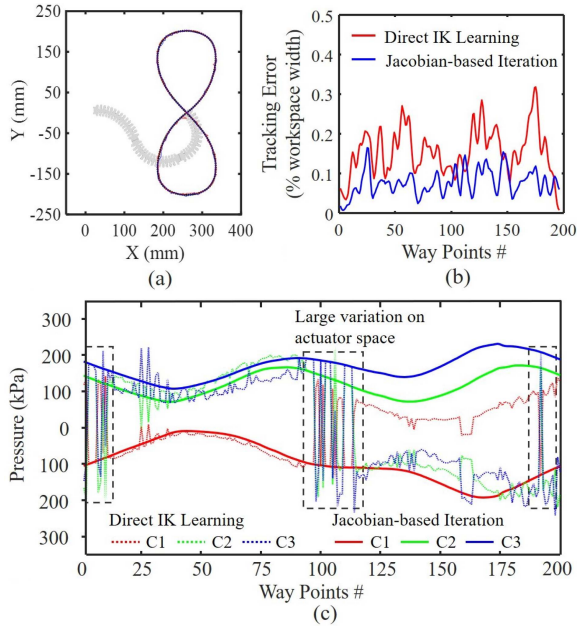


Fig. 10. (a) Path-following result on the soft finger manipulator with the "8"-shape trajectory. (b) Comparison of tracking errors in the task space from the direct IK learning versus the Jacobian-based iteration by learning (our method). (c) Visualization of IK solutions in the actuator space, where C1, C2, and C3 present the actuation parameters (i.e., pressures) in three different chambers—large variation can be found from the results obtained by direct IK learning.

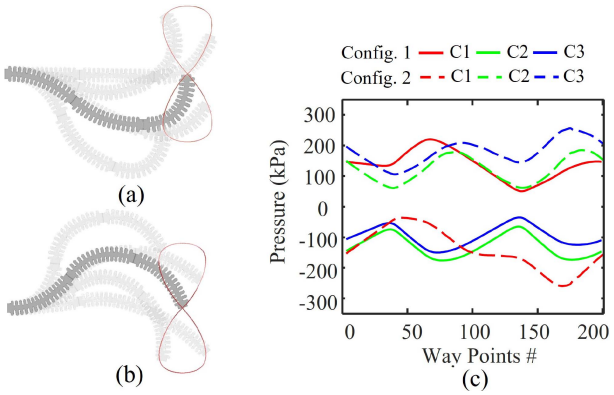


Fig. 11. Two different configurations of motion are shown in (a) and (b), both of which are feasible IK solutions for following the "8"-shaped path by the soft finger manipulator. (c) Smoothness in motion is guaranteed by the Jacobian-based iteration for both results is shown, where the actuation in every chamber has minimal variation in control parameters between neighboring waypoints.

IK solution at the starting waypoint (i.e., the initial value). As shown in Fig. 11, the configuration of the finger manipulator determined by our IK solver at a waypoint is always close to the IK solution of the previous waypoint where this dependency can trace back to the beginning point. This is because our Jacobian-based iteration tends to minimize the distance-based objective function defined in (1) while minimizing the change in actuation parameters. This preferred property is also kept when computing IK solutions for a motion passing through

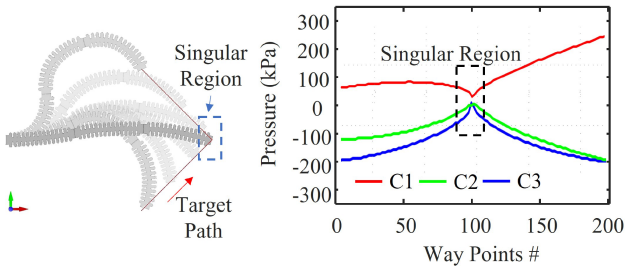


Fig. 12. Results for the path-following task passing through the singularity region. Our method can successfully compute feasible smooth motions when meeting singularity.

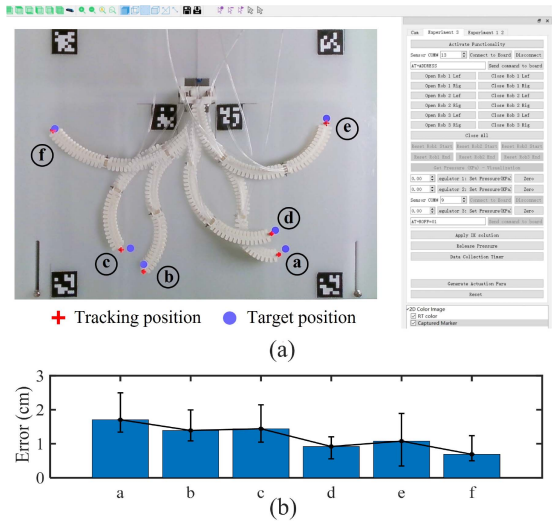


Fig. 13. Interactive positioning results for the soft manipulator with three finger actuators. (a) With a user-specified position given through the software interface, our Jacobian-based method is applied to determine the IK solution. (b) Bar chat presents the tracking errors for different target positions, where the repeatability is also studied and displayed as the range of deviation in tracking errors.

the singularity region (see the example in Fig. 12 for following an "L"-shaped path). As a Jacobian-based iterative solver, our method can always generate a nearly optimal solution for singularity points by applying appropriate terminal conditions (e.g., minimal variation in the value of the objective function).

2) Interactive Positioning: The experiment of interactive positioning is also conducted on the soft finger setup. As shown in Fig. 13(a), users can select the desired point location for the manipulator's tip through our interface, and our planner will compute the IK solutions as the corresponding actuation parameters. The computation can be efficiently completed at an average speed of 47 ms together with the sim-to-real network $\mathcal{N}_{s2r}$. As a result, users can interactively position the manipulator's tip—see also the supplementary video[1]. When different positions are selected in the workspace, the soft manipulator can move among configurations with large variations. The errors of positioning are evaluated and presented in Fig. 13(b) as a bar chart. It is found that all six target positions can be realized in the physical environment with tracking errors less than 0.9% of the

[1][Online]. Available: https://doi.org/10.1109/TMECH

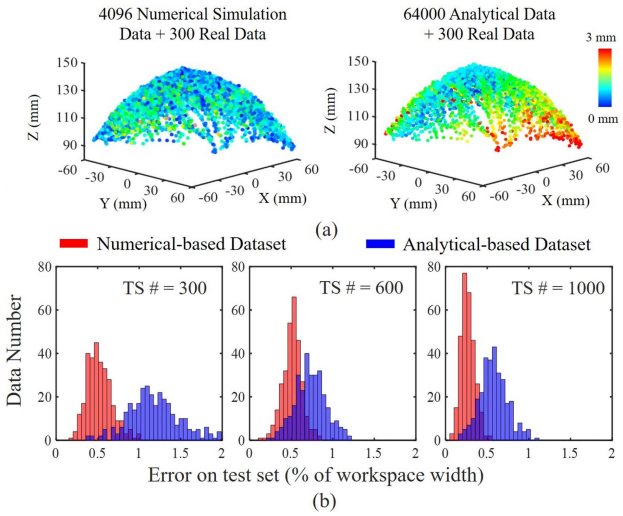


Fig. 14. Comparison for studying the performance of sim-to-real learning on numerical simulation-based (our approach) versus analytical computation-based [4], [9] training datasets. (a) Visualization of prediction errors in the task space. (b) Histograms show the distributions of prediction errors when using different numbers (mark as TS #) of samples to train $\mathcal{N}_{s2r}$.

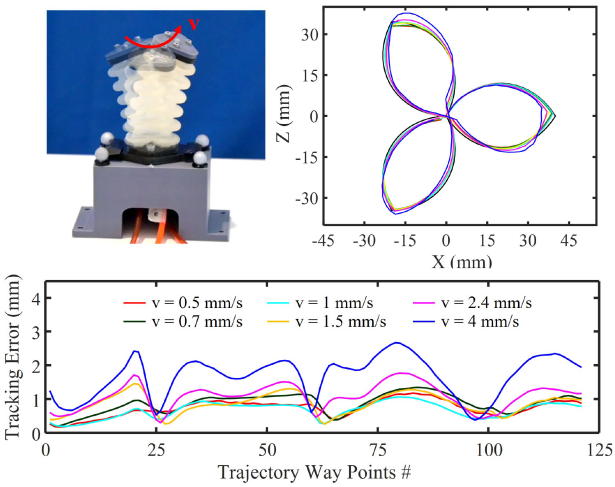


Fig. 15. Comparison of the behavior in path following by different speeds of motion on the three-chamber soft robot.

workspace's width. Note that each of these six target positions is tested ten times in random order to study the repeatability of our system. The results are displayed as the range of derivation on the bar chart.

It is observed that our method can generate results in different configurations for two close waypoints (e.g., the points b and c shown in Fig. 13) when using initial values that are always far from the resultant configurations (zero is adopted for all actuation parameters in this case). Together with the results presented in trajectory-following experiments (e.g., Fig. 11), our method shows the capability of determining one "nearest" IK solution among all feasible IK solutions.

C. Statistical Analysis for Sim-to-Real Transfer

Experiments have been conducted on the setup of three-chamber soft robot to explicitly compare the behavior of sim-to-real transfer by using different models in the virtual environment. Compared to the numerical simulator used in this work, the reality gap becomes larger when the dataset for training $\mathcal{N}_{fk}$ and $\mathcal{N}_J$ is obtained from an analytical model [4], [9]. When using the analytical model, more samples are needed for training the sim-to-real network $\mathcal{N}_{s2r}$ to achieve the similar accuracy. As observed in Fig. 14(a), the model trained by the dataset obtained from the numerical simulation (i.e., our approach) shows smaller prediction errors when using the same number of samples to learn the sim-to-real network $\mathcal{N}_{s2r}$. Note that this less accurate result is still observed even after using more samples generated from the analytical model (e.g., 64 000 in our experiment) when the number of real samples is fixed.

This experiment also proves that the sim-to-real network can effectively eliminate the gap between simulation and reality—although requiring different numbers of samples for the model learned from numerical simulation (i.e., our method) and the analytical model. Fig. 14(b) shows that the accuracy within 1% of the workspace width can be obtained at most points when TS# $=$ 300 and TS# $=$ 1000 are used by our method and the analytical model, respectively. As a general case, it is more costly to generate a large number of physical training samples than simulation-based samples. Generating a dataset of empirical samples in a large number is very time-consuming and may result in material failure. Our method proposed in this article converts this challenge into an approach, which is easier to realize; in other words, learning a more accurate predictor from more accurate samples generated by numerical simulation. As a result, the gap between prediction and reality can be reduced and fixed by a light sim-to-real network.

V. Discussion

This section discusses the limitation of our learning-based IK solver, where a rigorous analysis is conducted to show the influence with different velocities and external loads on soft robot. On the other hand, the possible extension of pose estimation by our method is presented.

In the proposed Jacobian-based learning, we focus on the computation of quasistatic kinematics. The training datasets in both virtual and physical setups are collected under the quasistatic status, where the hysteresis problem in soft materials is neglected. As one major limitation of this work, only considering the quasistatic model will bring large errors in the task of path-following when the dynamic behavior of soft robot is performed (e.g., the velocity of motion is high). Experiments are conducted to study the influence of speed in motion to the trajectory's accuracy, and the results are shown in Fig. 15. When the speed of motion is set as less than 1.5 mm/s, the tracking errors can be controlled less than 1.3 mm. However, when the speed is increased to 4 mm/s, i.e., the maximum speed that the actuation system shown in Fig. 3(c) can support, a relatively large error (i.e., the maximal error as 2.7 mm) can be found on the trajectory. One possible solution to incorporate the hysteresis property of soft materials and dynamic behavior of soft robots in IK computing is to apply the time-variant network structure (e.g., recurrent neural network [40]) based on datasets generated in

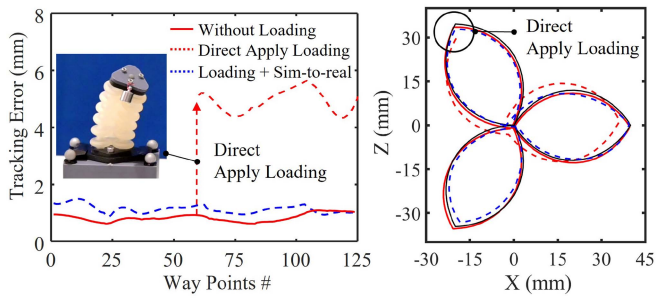


Fig. 16. Performance of the proposed IK solver when external load is applied to the end-effector, where black shows the target trajectory.

different velocities. This requires much larger training datasets, the generation of which is very time-consuming [29].

In this work, the proportional integral derivative (PID controller is applied to achieve stable pressure to drive soft robots. The vision system is only used to generate ground-truth data for the training purpose. In the verification process, the open-loop system is applied to allow soft robots to achieve given tasks. The trained network with sim-to-real transfer can already achieve an acceptable precision in motion without using the closed-loop controller.

On the other aspect, the trained sim-to-real network will lose its capability to accurately compute the IK solution when external loads are added to the end-effector of a soft robot. As shown in Fig. 16, significantly enlarged tracking errors can be observed when external load is directly applied (present by dashed lines in red). As an additional test, we train a new sim-to-real network by using the soft robot with load. It can be found that the tracking errors become small again by using this newly trained network. This demonstrates the functionality of the sim-to-real transfer to handle external loads.

Another drawback of this work is that we neglect pose information in the pipeline. As an important extension of the learning framework, poses of a soft robot (i.e., including orientation) are to be considered [41], [42]. One possible solution is to directly add the rotation into the output layers of $\mathcal{N}_{fk}$ and $\mathcal{N}_J$. Meanwhile, training positions and orientations together need to consider the balance between their different units. Higher DoFs in actuator space are needed to enhance the feasibility of IK solutions.

VI. Conclusion

In this article, we presented a method to train the FK model and its Jacobian together as two neural networks to realize the real-time computation of IKs on soft robots, which is formulated as an optimization problem. As our method can generate smooth motion in a redundant system, it outperformed the existing approaches of direct IK learning. Considering the difficulty in generating large datasets on hardware setups, we adopted a highly effective simulator to generate the training datasets, and later applied a sim-to-real network to transfer the kinematic model onto hardware. A lightweight network was employed for sim-to-real mapping, so that it can be trained by using a small number of samples. This sim-to-real strategy allows our approach to work on different soft robots that have variations caused by materials and fabrication processes. The main advantages of our method include the efficient computation and the ease of applying the sim-to-real learning transfer.

We tested the behavior of our learning-based method in the tasks of path-following and interactive positioning on two different soft robotic setups. Our method can solve the IK problem for soft robots effectively and make a good control for the kinematic tasks. As a future work, we plan to explore the possibility of using time-related data for sim-to-real transfer learning that may further enhance the accuracy of IK computing. Moreover, it is also interesting to develop a more transferable learning pipeline that makes the trained model of kinematics can be easily applied to similar designs (e.g., when only the sizes of soft robots are changed).

References

[1] D. Rus and M. Tolley, "Design, fabrication and control of soft robots," *Nature*, vol. 521, pp. 467–75, 2015.
[2] P. Polygerinos, Z. Wang, K. C. Galloway, R. J. Wood, and C. J. Walsh, "Soft robotic glove for combined assistance and at-home rehabilitation," *Robot. Auton. Syst.*, vol. 73, pp. 135–143, 2015.
[3] A. Melingui, O. Lakhal, B. Daachi, J. B. Mbede, and R. Merzouki, "Adaptive neural network control of a compact bionic handling arm," *IEEE/ASME Trans. Mechatronics*, vol. 20, no. 6, pp. 2862–2875, Dec. 2015.
[4] D. Drotman, M. Ishida, S. Jadhav, and M. T. Tolley, "Application-driven design of soft, 3-D printed, pneumatic actuators with bellows," *IEEE/ASME Trans. Mechatronics*, vol. 24, no. 1, pp. 78–87, Feb. 2019.
[5] T. Ranzani, G. Gerboni, M. Cianchetti, and A. Menciassi, "A bioinspired soft manipulator for minimally invasive surgery," *Bioinspiration Biomimetics*, vol. 10, no. 3, May 2015, Art. no. 035008.
[6] G. S. Chirikjian and J. W. Burdick, "A modal approach to hyper-redundant manipulator kinematics," *IEEE Trans. Robot. Autom.*, vol. 10, no. 3, pp. 343–354, Jun. 1994.
[7] B. A. Jones and I. D. Walker, "Kinematics for multisection continuum robots," *IEEE Trans. Robot.*, vol. 22, no. 1, pp. 43–55, Feb. 2006.
[8] T. G. Thuruthel, Y. Ansari, E. Falotico, and C. Laschi, "Control strategies for soft robotic manipulators: A survey," *Soft Robot.*, vol. 5, no. 2, pp. 149–163, 2018.
[9] M. Rolf and J. J. Steil, "Constant curvature continuum kinematics as fast approximate model for the bionic handling assistant," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2012, pp. 3440–3446.
[10] F. Renda, M. Giorelli, M. Calisti, M. Cianchetti, and C. Laschi, "Dynamic model of a multibending soft robot arm driven by cables," *IEEE Trans. Robot.*, vol. 30, no. 5, pp. 1109–1122, Oct. 2014.
[11] 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," *Adv. Eng. Mater.*, vol. 18, no. 6, pp. 978–988, 2016.
[12] M. S. Xavier, A. J. Fleming, and Y. K. Yong, "Finite element modeling of soft fluidic actuators: Overview and recent developments," *Adv. Intell. Syst.*, vol. 3, no. 2, 2021, Art. no. 2000187.
[13] D. E. Orin and W. W. Schrader, "Efficient computation of the Jacobian for robot manipulators," *Int. J. Robot. Res.*, vol. 3, no. 4, pp. 66–75, 1984.
[14] J. Hiller and H. Lipson, "Dynamic simulation of soft multimaterial 3D-printed objects," *Soft Robot.*, vol. 1, no. 1, pp. 88–101, 2014.
[15] C. Duriez, "Control of elastic soft robots based on real-time finite element method," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2013, pp. 3982–3987.
[16] G. Fang, C.-D. Matte, T.-H. Kwok, and C. C. L. Wang, "Geometry-based direct simulation for multi-material soft robots," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 4194–4199.
[17] G. Fang, C.-D. Matte, R. B. N. Scharff, T.-H. Kwok, and C. C. L. Wang, "Kinematics of soft robots by geometric computing," *IEEE Trans. Robot.*, vol. 36, no. 4, pp. 1272–1286, Aug. 2020.
[18] O. Goury and C. Duriez, "Fast, generic, and reliable control and simulation of soft robots using model order reduction," *IEEE Trans. Robot.*, vol. 34, no. 6, pp. 1565–1576, Dec. 2018.

[19] M. Giorelli, F. Renda, G. Ferri, and C. Laschi, "A feed-forward neural network learning the inverse kinetics of a soft cable-driven manipulator moving in three-dimensional space," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst*., 2013, pp. 5033–5039.
[20] M. Giorelli, F. Renda, G. Ferri, and C. Laschi, "A feed forward neural network for solving the inverse kinetics of non-constant curvature soft manipulators driven by cables," in *Proc. Dyn. Syst. Control Conf.*, 2013, vol. 56147, Art. no. V003T38A001.
[21] M. Rolf and J. J. Steil, "Efficient exploratory learning of inverse kinematics on a bionic Elephant trunk," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 6, pp. 1147–1160, Jun. 2014.
[22] M. Giorelli, F. Renda, M. Calisti, A. Arienti, G. Ferri, and C. Laschi, "Neural network and Jacobian method for solving the inverse statics of a cable-driven soft arm with nonconstant curvature," *IEEE Trans. Robot.*, vol. 31, no. 4, pp. 823–834, Aug. 2015.
[23] J. Chen and H. Y. K. Lau, "Learning the inverse kinematics of tendon-driven soft manipulators with K-nearest neighbors regression and Gaussian mixture regression," in *Proc. 2nd Int. Conf. Control, Automat. Robot.*, 2016, pp. 103–107.
[24] R. Grassmann, V. Modes, and J. Burgner-Kahrs, "Learning the forward and inverse kinematics of a 6-DoF concentric tube continuum robot in SE(3)," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst*., 2018, pp. 5125–5132.
[25] R. F. Reinhart, Z. Shareef, and J. J. Steil, "Hybrid analytical and data-driven modeling for feed-forward robot control," *Sensors*, vol. 17, no. 2, 2017, Art. no. 311.
[26] J. M. Bern, Y. Schnider, P. Banzet, N. Kumar, and S. Coros, "Soft robot control with a learned differentiable model," in *Proc. IEEE Int. Conf. Soft Robot.*, 2020, pp. 417–423.
[27] T. G. Thuruthel, E. Falotico, M. Cianchetti, F. Renda, and C. Laschi, "Learning global inverse statics solution for a redundant soft robot," in *Proc. 13th Int. Conf. Inform. Control, Automat. Robot.*, 2016, pp. 303–310.
[28] T. G. Thuruthel, E. Falotico, M. Cianchetti, and C. Laschi, "Learning global inverse kinematics solutions for a continuum robot," in *Proc. Symp. Robot Des., Dyn. Control*, 2016, pp. 47–54.
[29] M. Li, R. Kang, D. T. Branson, and J. S. Dai, "Model-free control for continuum robots based on an adaptive Kalman filter," *IEEE/ASME Trans. Mechatronics*, vol. 23, no. 1, pp. 286–297, Feb. 2018.
[30] T. G. Thuruthel, B. Shih, C. Laschi, and M. T. Tolley, "Soft robot perception using embedded soft sensors and recurrent neural networks," *Sci. Robot.*, vol. 4, no. 26, 2019, Art. no. eaav1488.
[31] R. B. N. Scharff, R. M. Doornbusch, E. L. Doubrovski, J. Wu, J. M. P. Geraedts, and C. C. L. Wang, "Color-based proprioception of soft actuators interacting with objects," *IEEE/ASME Trans. Mechatronics*, vol. 24, no. 5, pp. 1964–1973, Oct. 2019.
[32] Y. Sun, Y. S. Song, and J. Paik, "Characterization of silicone rubber based soft pneumatic actuators," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst*., 2013, pp. 4446–4453.
[33] D. Kubus, R. Rayyes, and J. J. Steil, "Learning forward and inverse kinematics maps efficiently," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst*., 2018, pp. 5133–5140.
[34] E. S. Marquez, J. S. Hare, and M. Niranjan, "Deep cascade learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 11, pp. 5475–5485, Nov. 2018.
[35] S. Kriegman *et al.*, "Scalable sim-to-real transfer of soft robot designs," in *Proc. 3rd IEEE Int. Conf. Soft Robot.*, 2020, pp. 359–366.
[36] H. Park, J. Cho, J. Park, Y. Na, and J. Kim, "Sim-to-real transfer learning approach for tracking multi-DOF ankle motions using soft strain sensors," *IEEE Robot. Autom.*, vol. 5, no. 2, pp. 3525–3532, Apr. 2020.
[37] H. Donat, S. Lilge, J. Burgner-Kahrs, and J. J. Steil, "Estimating tip contact forces for concentric tube continuum robots based on backbone deflection," *IEEE Trans. Med. Robot. Bionics*, vol. 2, no. 4, pp. 619–630, Nov. 2020.
[38] M. S. Malekzadeh, S. Calinon, D. Bruno, and D. G. Caldwell, "Learning by imitation with the stiff-flop surgical robot: A biomimetic approach inspired by octopus movements," *Robot. Biomimetics*, vol. 1, no. 1, pp. 1–15, 2014.
[39] M. Wiese, G. Runge-Borchert, B.-H. Cao, and A. Raatz, "Transfer learning for accurate modeling and control of soft actuators," in *Proc. 4th IEEE Int. Conf. Soft Robot.*, 2021, pp. 51–57.
[40] T. G. Thuruthel, E. Falotico, F. Renda, and C. Laschi, "Model-based reinforcement learning for closed-loop dynamic control of soft robotic manipulators," *IEEE Trans. Robot.*, vol. 35, no. 1, pp. 124–134, Feb. 2019.
[41] R. Grassmann and J. Burgner-Kahrs, "On the merits of joint space and orientation representations in learning the forward kinematics in SE(3)," in *Proc. Robot.: Sci. Syst.*, pp. 1–10, Jun. 2019.
[42] V. Peretroukhin, M. Giamou, D. M. Rosen, W. N. Greene, N. Roy, and J. Kelly, "A smooth representation of SO(3) for deep rotation learning with uncertainty," in *Proc. Robot.: Sci. Syst.*, pp. 1–9, Jul. 2020.

Guoxin Fang (Member, IEEE) received the B.E. degree in mechanical engineering from the Beijing Institute of Technology, Beijing, China, in 2016. He is currently working toward the Ph.D. degree in advanced manufacturing with the Delft University of Technology, Delft, The Netherlands.

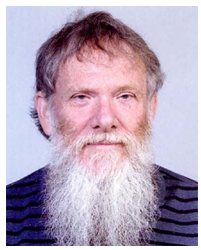
He is currently a Research Assistant with the Department of Mechanical, Aerospace and Civil Engineering, The University of Manchester, Manchester, U.K. His research interests include computational design, digital fabrication, and robotics.

Yingjun Tian received the B.Eng. in mechanical engineering and automation from the University of Science and Technology of China, Hefei, China, in 2019. He is currently working toward the Ph.D. degree in mechanical engineering with Smart Manufacturing Group, Department of Mechanical, Aerospace and Civil Engineering, The University of Manchester, Manchester, U.K.

His research interests include computational design and shape control of soft robots.

Zhi-Xin Yang (Member, IEEE) received the B.Eng. degree in mechanical engineering from the Huazhong University of Science and Technology, Wuhan, China, in 1992, and the Ph.D. degree in industrial engineering and engineering management from the Hong Kong University of Science and Technology, Hong Kong, in 2000.

He is currently an Associate Professor of Electromechanical Engineering with the State Key Laboratory of Internet of Things for Smart City, Department of Electromechanical Engineering, Faculty of Science and Technology, University of Macau, Zhuhai, China. His research interests include fault diagnosis and prognosis, machine learning, and computer vision-based robotic control.



Jo M. P. Geraedts received the Ph.D. degree in physics from Radboud University, Nijmegen, The Netherlands, in 1983.

He then joined Océ, today a Canon group company, where he worked on the development of digital print processes and workflow for document and industrial printing. From 2000 to 2013, he was a Manager with Océ Industrial Design Department. In 2008, he became a Full Professor and the Chair of Mechatronic Design with the Faculty of Industrial Design Engineering, Delft University of Technology, Delft, The Netherlands. His research interests include 3-D scanning, 3-D multimaterial printing, digital reproduction of fine arts, digital manufacturing, and soft robotics.

Charlie C. L. Wang (Senior Member, IEEE) received the B.Eng. degree in mechatronics engineering from the Huazhong University of Science and Technology, Wuhan, China, in 1998, and the Ph.D. degree in mechanical engineering from The Hong Kong University of Science and Technology, Hong Kong, in 2002.

He is currently a Professor and the Chair of Smart Manufacturing with The University of Manchester, Manchester, U.K. His research interests include digital manufacturing, computational design, additive manufacturing, soft robotics, mass personalization, and geometric computing.

Prof. Wang was elected as a fellow of the American Society of Mechanical Engineers in 2013.