Proceeding of the IEEE
International Conference on Robotics and Biomimetics
Dali, China, December 2019

A Learning-based Inverse Kinematics Solver for a Multi-Segment Continuum Robot in Robot-Independent Mapping

Jiewen Lai, Kaicheng Huang, and Henry K. Chu*

***Abstract*— Inverse kinematics (IK) is one of the most fundamental problems in robotics, as it makes use of the kinematics equations to determine the joint configurations necessary to reach a desired end-effector pose. In the field of continuum robot, solving the IK is relatively challenging, owing to kinematic redundancy with infinite number of solutions.**

In this paper, we present a simplified model to represent a multi-segment continuum robot using virtual rigid links. Based on the model, its IK can be solved using a multilayer perceptron (MLP), a class of feedforward neural network (FNN). The transformation between virtual joint space to task space is described using Denavit-Hartenberg (D-H) convention. Using 20,000 established training data for supervised learning, the MLP reaches a mean squared error of 0.022 for a dual-segment continuum robot. The trained MLP is then used to find the joints for different end-effector positions, and the results show a mean relative error of 2.90% can be on the robot configuration. Hence, this simplified model and its MLP provide a simple method to evaluate the IK solution of a two-segment continuum robot, which can also be further generalized and implemented in multi-segment cases.

I. Introduction

The invention of hyper-redundant manipulator [1] and continuum robot [2] have enriched the variety of robots. Theoretically, a continuum robot has infinite degrees of freedom (DOFs) as it can bend continuously. It has rigid yet flexible property through the use of elastic or soft materials. The inherent dexterity provides the serpentine robot arm with advantageous in sinuous surroundings. To date, various actuation mechanism for continuum robot have been proposed, including tendon or cable driven [3]–[5], pneumatic [6] or hydraulic driven [7], concentric rotational driven [8], phase-change material deformation driven [9], and ionic polymer-metal composites (IPMC) deformation driven [10]. In short, all actuation methods in continuum robot system aim at realizing the physical deformation of a manipulator.

In model-based approach, a mathematical representation of the robot using quantifiable model parameters is of the essence. Most of the recent literature employ the constant-curvature (or "piecewise constant curvature") approximation, which models the continuum robot segment as a series of circular arcs in a mutually tangent manner [11]. The geometry defines the centroid backbone of a single segment continuum robot as an ideal arc which can be parameterized by curvature (κ), rotation (ϕ), and bending (ϑ) [12]. These three factors or their variations define a transitional space called configuration space [11] which correlates between the actuator space and the task space. The mapping from arc parameters to robot Cartesians along the continuum robot, which is independent, can be attained by forward kinematics.

Inverse kinematics (IK) concerns the mathematical method of deriving the joint variables which can bring the robot (end-effector) to the desired position/orientation in task space. In general, the inverse kinematics can be categorized as three methods, namely, algebraic, geometric, and iterative [13]. Regarding the continuum manipulator, IK algorithm usually follows either an analytical or a numerical approach [14]–[16], which are not efficient in real-time operation because of their complexity and high computational cost.

Neural network approach can iteratively compute the IK in robotics. A neural network architecture is an initial information processing structure of interest in neurocomputing. The network learns to generate a functional relation from given training set, which maps input vectors to corresponding output vectors.The network stores the connection strengths (as weights) between processing units (as neurons). Weightings are repeatedly adjusted during the learning process until they reach the thresholds. According to the *universal approximation theorem*, a properly designed topological architecture features the network with the capability to solve complex mapping between new input and unknown output in spite of its non-linearity [17], [18]. Researchers attempted to design multiplayer feedforward neural networks (FNNs) to solve the IK of rigid robot arms used in the space station [19]. The training performance between Kohonen Maps (KM) and multilayer perceptron (MLP) for a rigid-body robot were compared and evaluated in [20]. Investigation [19] concluded that, with adequate training, a well-built neural network may provide a practical and efficient solution to the IK problem.

Solving the IK of continuum robot manipulator by using artificial neural network (ANN) have received much attention in recent years. Thuruthel *et al.* [21] introduced a machine learning-based kinematic controller for a 6-DOF tendon-driven continuum manipulator, which adopted a forward kinematic model using sensory feedback from experiments to estimate a time-dependent trajectory based on a step-wise differential IK. Grassmann *et al.* [22] built an ANN with ReLU (rectified linear unit) activation function to solve the IK of a 6-DOF concentric tubes manipulator, which obtained a satisfactory accuracy through prototype-based validations.

However, many of the existing learning-based IK solutions for continuum robots are limited to mapping parameters that are specific to a particular type of continuum robot. The lack of generality occurs due to the variations among different robot configurations. A more general model to represent the

The authors are with the Department of Mechanical Engineering, The Hong Kong Polytechnic University, Hong Kong. *Corresponding author, e-mail: henry.chu@polyu.edu.hk

978-1-7281-6321-5/19//$31.00 ©2019 IEEE

configuration of a continuum robot manipulator would be helpful for utilizing different techniques for robot planning and control.

In this context, this paper develops a simplified model to represent a general multi-segment continuum robot. Under the constant curvature assumption, the robot can be re-modeled as a rigid-link manipulator with multiple DOFs such that many conventional techniques can be applied for task execution and path planning in configuration space. With the approximate model, an IK solver is developed using MLP. The networks are trained with robot configuration expressed using virtual joints and six-dimensional information of the end-effector which are acquired from forward kinematics. This study provides an alternative mean to efficiently solve the robot IK problem for multi-segment continuum robot.

The paper will further illustrate the following aspects. In section II, a dual-segment continuum robot manipulator will be modeled using virtual links under the constant curvature assumption. Forward kinematics will be demonstrated in Denavit-Hartenberg (D-H) representation. Section III will introduce the proposed MLP-based IK solver in terms of network architecture and data analytic in *Knowledge Discovery in Databases* (KDD) [23] process. In section IV, performances of the proposed network will be analyzed. Utilizing the IK solver, virtual path planning will be performed to test the effectiveness of the solver. Conclusions will be addressed in Section V.

II. Forward Kinematic Modeling

In this section, a pseudo-configuration space of a continuum segment is described using virtual rigid-links. Under the constant curvature assumption, the frame representation of the end-effector with respect to (w.r.t.) robot base is given by forward kinematics projection.

A. Simplified Modeling of Continuum Robot

Theoretically, a continuum robot manipulator has an infinite DOF for bending. However, many continuum robots have been observed to exhibit approximate constant curvature behavior during bending, especially for manipulators which are light-weighted and proportionally short enough [11]. Based on the piecewise constant curvature assumption, a continuum robot manipulator can be simplified as a serial-link robot with several revolute joints. The model constitutes a configuration space since a continuum robot can be fully described. The robot configuration of single segment is shown in Fig. 1. Consider the n-th segment of the continuum manipulator with a length of L_n, the configuration of n-th continuous segment can be defined by δ_n and θ_n, where δ_n represents the angular deviation in terms of bending direction of the n-th segment w.r.t. the base frame of segment n, and θ_n denotes the extent of bending in such direction. The bending extent is defined as the complementary angle of the tangential intersection of two ends of the circular arc, which is θ_n as shown in Fig. 1. By doing so, the configuration of an ∞-DOF continuum robot can be interpreted as a 2-DOF series-link robot with two virtual rigid links with equal length. The length of virtual link $\frac{L_n^\dagger}{2}$ is given by $\frac{L_n^\dagger}{2} = \frac{L_n}{\theta_n}\tan\frac{\theta_n}{2}$, and it equals to $\frac{L}{2}$ when bending angle goes to 0, i.e., $\lim_{\theta_n \to 0} \frac{L_n^\dagger}{2} = \frac{L_n}{2}$.

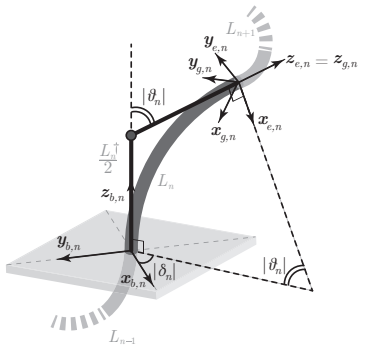

Fig. 1. Robot configuration at the n-th segment under the constant curvature approximation. Subscripts b, e, g represent the base frame, end-effector frame, and global frame, respectively.

Considering a generalized model, the configuration space of an n-segment continuum robot arm can be described by $\mathbf{\Psi}_n = [\delta_1, \theta_1, ..., \delta_n, \theta_n]^\top \in \mathbb{R}^{2n}$, where $\delta_n \in (0, 2\pi)$ denotes a rotational angle of the n-th segment w.r.t. the $(n-1)$-th continuum segment along robot body direction, and $\theta_n \in (0, \frac{\pi}{2})$ denotes the bending angle of the n-th segment. By this means, an ∞-DOF continuum manipulator system is simplified as a $2n$-DOF open-chain robot arm which is composed of finite virtual rigid links. The task space, which comprises the position and orientation in Euler angles (roll–pitch–yaw angles) of the end-effector of the n-th segment, is represented by $\mathbf{x}_e = [x_e, y_e, z_e, \alpha_e, \beta_e, \gamma_e]^\top \in \mathbb{R}^6$, in which $\alpha_e, \beta_e, \gamma_e$ are the $\mathbf{ZYX}$ Euler angles representing the final orientation of the body (end-effector) [24]. The robot-independent mapping from the configuration space $\mathbf{\Psi}_n$ to task space $\mathbf{x}_i$ is given by $\mathbf{x}_i = f(\mathbf{\Psi}_n)$, where $f(\cdot)$ signifies the forward kinematics, and $\mathbf{x}_i$ represents the i-th Cartesian coordinate. Vise versa, the inverse kinematics is given by $\mathbf{\Psi}_n = f^{-1}(\mathbf{x}_i)$.

The forward kinematics $f(\cdot)$ of the simplified model can be attained by multiplying homogeneous transformation matrices in $SE(3)$ while the robot is expressed in D-H convention. This paper will take a two-segment continuum robot manipulator (as shown in Fig. 2) as an example to conduct further investigation. Without loss of generality, the mathematical expression can be generalized for n-segment continuum manipulator case. The proposed method has been verified in 3-segment condition as well. Owning to the limitation of the scope, higher numbers of segment will not be discussed here.

B. Dual-Segment Continuum Robot Modeling

In Fig. 2, a two-segment continuum robot is modeled in link-joint structure. The bending point is considered as a revolute joint which is normal to both links. Two consecutive continuous segments are connected by another revolute joint whose normal vector is pointed to the next virtual link. The length of proximal and distal continuous segments are L_1 and

L_2, respectively. The length of virtual links are superscripted with †. The D-H parameters of the simplified robotic model are obtained in terms of i frames as listed in Table I.

TABLE I
D-H PARAMETERS OF A TWO-SEGMENT CONTINUUM ROBOT MODEL

$\boldsymbol{i}$	$\boldsymbol{\alpha}_i$	$\boldsymbol{a}_i$	$\boldsymbol{d}_i$	$\boldsymbol{\theta}_i$
1	0	0	0	δ_1^*
2	$-\frac{\pi}{2}$	0	$\frac{L_1^\dagger}{2}$	$-\frac{\pi}{2}$
3	$\frac{\pi}{2}$	0	0	θ_1^*
4	$-\frac{\pi}{2}$	0	$\frac{L_1^\dagger+L_2^\dagger}{2}$	δ_2^*
5	0	$\frac{L_2^\dagger}{2}$	0	$\theta_2^*-\frac{\pi}{2}$

$*$variable

As a dual-segment manipulator, the complete robot configuration is parameterized as $\boldsymbol{\Psi}_2 = [\delta_1, \theta_1, \delta_2, \theta_2]^\top \in \mathbb{R}^4$. There are six D-H frames $\mathbf{O}_i$ ($i \in \{0, 1, 2, ..., 5\}$) which are defined in order to represent the end-effector frame $\mathbf{O}_5$ w.r.t. the base frame $\mathbf{O}_0$. The homogeneous transformation that relates coordinate frame i to $(i-1)$ is denoted by

$$\mathbf{T}_i^{i-1} = \begin{bmatrix} \boldsymbol{\mathcal{R}}_i^{i-1} & \mathbf{p}_i^{i-1} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \tag{1}$$

where $\boldsymbol{\mathcal{R}}_i^{i-1}$ and $\mathbf{p}_i^{i-1}$ describe the rotation and translation of the coordinate frame $\mathbf{O}_i$ in relation to the coordinate frame $\mathbf{O}_{i-1}$, respectively. For series-link robot, the homogeneous transformation of the end-effector coordinate frame yields

$$\mathbf{T}_i^0 = \prod_{i=1}^{i} \begin{bmatrix} \boldsymbol{\mathcal{R}}_i^{i-1} & \mathbf{p}_i^{i-1} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \tag{2}$$

where, $\lim_{\boldsymbol{\Psi}_n \to \mathbf{0}} \left[\mathbf{p}_i^0\right] = \begin{bmatrix} 0 & 0 & \sum_{i=1}^{n} L_n \end{bmatrix}^\top$ reveals the end-effector position in singularity situation.

In the case of a dual-segment continuum robot shown as in Fig.2, the Cartesian coordinates of the end-effector w.r.t. the base system is

$$\mathbf{T}_5^0 = \mathbf{T}_1^0\mathbf{T}_2^1\mathbf{T}_3^2\mathbf{T}_4^3\mathbf{T}_5^4 = \left[\begin{array}{ccc|c} \boldsymbol{n}_{xe} & \boldsymbol{o}_{xe} & \boldsymbol{a}_{xe} & x_e \\ \boldsymbol{n}_{ye} & \boldsymbol{o}_{ye} & \boldsymbol{a}_{ye} & y_e \\ \boldsymbol{n}_{ze} & \boldsymbol{o}_{ze} & \boldsymbol{a}_{ze} & z_e \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{3}$$

where, $\mathbf{p}_e = [x_e, y_e, z_e]^\top$ and 1 indicates the matrix scale. The transformation from rotation matrix $\boldsymbol{\mathcal{R}}_5^0$ to Euler representatives follows the method in [24].

The explicit expression of Eq. (3) is given in Eq. (4), where $c(\cdot)$ and $s(\cdot)$ denote the cosine and sine operation, respectively. After converting the end-effector orientation from rotation matrix in $SO(3)$ to Euler angles, the end-effector Cartesian $\mathbf{x}_e$ w.r.t. task space can be attained. Robot workspace describes the reachable locations of the robot end-effector. The end-effector positions w.r.t. Cartesian space of the robot base are derived from Eq. (4) and plotted in Fig. 3. In Fig. 3, the point clouds represent the $\mathbf{p}_e \subseteq \mathbf{x}_e$ that mapped 20,000 sets of randomized $\boldsymbol{\Psi}_2$ from the configuration space.

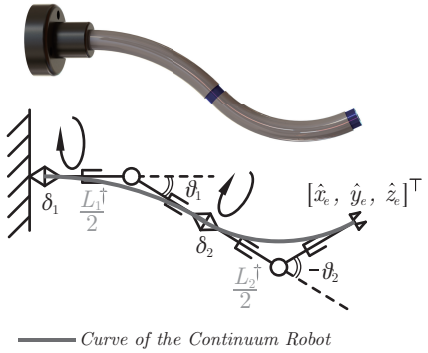

Fig. 2. A two-segment continuum robot and its simplified model.

III. SOLVING CONTINUUM ROBOT INVERSE KINEMATICS BY MULTILAYER PERCEPTRON

This section demonstrates an MLP-based IK solver for the multi-segment continuum robot in robot-independent spaces. A two-segment manipulator with segmental lengths $L_1 = 50$ mm, $L_2 = 50$ mm (as shown in Fig 2) will be used as an example to conduct the study. Network structure will be optimized on the basis of training performance.

A. Feedforward Neural Network

The proposed IK solver is utilizing MLP, which is a class of feedforward artificial neural network. A basic MLP

$$\mathbf{T}_5^0 = \begin{bmatrix} \sigma_2\sigma_4 - s\delta_1 s\theta_1\sigma_1 & -\sigma_1\sigma_4 - s\delta_1\sigma_2 s\theta_1 & c\delta_1 c\delta_2 - s\delta_1 s\delta_2 c\theta_1 & \frac{L_2^\dagger\sigma_2\sigma_4}{2} + s\delta_1 s\theta_1\sigma_5 - \frac{L_2^\dagger s\delta_1 s\theta_1\sigma_1}{2} \\ \sigma_2\sigma_3 - c\delta_1 s\theta_1\sigma_1 & -\sigma_1\sigma_3 + c\delta_1\sigma_2 s\theta_1 & c\delta_2 s\delta_1 + c\delta_1 s\delta_2 c\theta_1 & \frac{L_2^\dagger\sigma_2\sigma_3}{2} - c\delta_1 s\theta_1\sigma_5 - \frac{L_2^\dagger c\delta_1 s\theta_1\sigma_1}{2} \\ -c\theta_1\sigma_1 - c\delta_2\sigma_2 s\theta_1 & c\delta_2 s\theta_1\sigma_1 - c\theta_1\sigma_2 & s\delta_2 s\theta_1 & \frac{L_1^\dagger}{2} + c\theta_1\sigma_5 - \frac{L_2^\dagger c\theta_1\sigma_1 + L_2^\dagger c\delta_2\sigma_2 s\theta_1}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4}$$

$$\text{where, } \begin{cases} \sigma_1 = s(\theta_2 - \frac{\pi}{2}), \\ \sigma_2 = c(\theta_2 - \frac{\pi}{2}), \\ \sigma_3 = s\delta_1 s\delta_2 - c\delta_1 c\delta_2 c\theta_1, \\ \sigma_4 = c\delta_1 s\delta_2 + c\delta_2 s\delta_1 c\theta_1, \\ \sigma_5 = \frac{L_1^\dagger + L_2^\dagger}{2}. \end{cases} \text{ and } \begin{cases} L_1^\dagger = \dfrac{2L_1}{\theta_1}\tan\dfrac{\theta_1}{2}, \\ L_2^\dagger = \dfrac{2L_2}{\theta_2}\tan\dfrac{\theta_2}{2}. \end{cases}$$

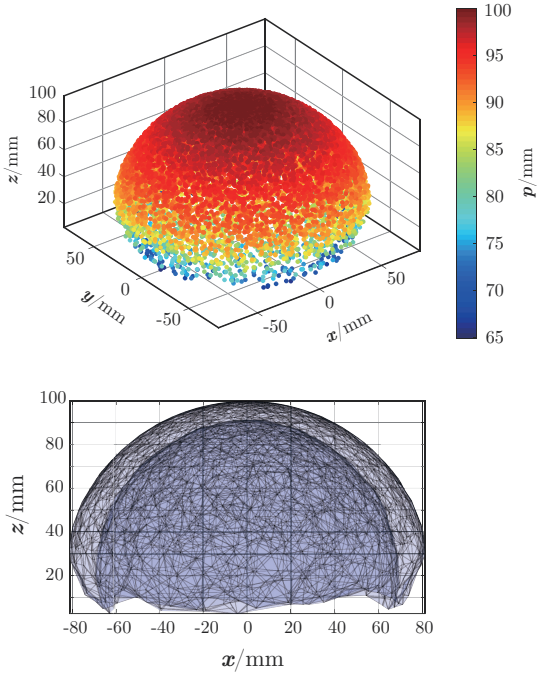

Fig. 3. The robot workspace derived from forward kinematics. Top: The point clouds generated 20,000 sets of randomized $\mathbf{\Psi}_2$ represents the positional distribution of a dual-segment continuum robot's end-effector (20,000 sets of random $\mathbf{\Psi}_2$ in the range $\delta_n \in (0, 2\pi)$, $\theta_n \in (0, \frac{\pi}{2})$ in "elbow-down" posture, and $L_1 = L_2 = 50$ mm). The color bar represents the distance vector $\boldsymbol{p}$ from the robot base $\mathbf{O}_0 = (0, 0, 0)$ to the end-effector $\mathbf{p}_e = (x_e, y_e, z_e)$, i.e., $||\boldsymbol{p}|| = \sqrt{{x_e}^2 + {y_e}^2 + {z_e}^2}$. Bottom: The $\boldsymbol{xz}$ view of point clouds boundary using triangular meshes shows that the workspace forms a spherical shell (light blue layer).

consists of at least three layers of nodes: an input layer, a hidden layer, and an output layer. The input layer has a connection with the network input. In hidden layers, each subsequent layer is fully connected from the previous layer with a non-linear activation function $\boldsymbol{\varphi}(\cdot)$. The output layer produces the network's output via linear functions. The network follows a feedforward learning process. A generic MLP architecture is given in Fig. 4.

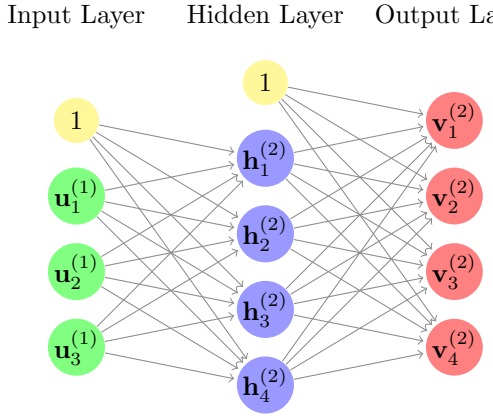

Fig. 4. A general feedforward neural network in 3-4-4 structure. $\mathbf{h}^{(l)}$ denotes the hidden nodes via activation functions.

For layer l, assuming that the total output is $\mathbf{V}^{(l)}$, in which $\mathbf{c}_j^{(l)}$ and $\mathbf{v}_j^{(l)}$ denote the input and output of node j. The connection between layer l and layer $(l-1)$ relies on a weighting matrix $\mathbf{W}^{(l)}$. The weights between the k-th node of layer $(l-1)$ and the j-th node of layer l is given by $\mathbf{w}_{jk}^{(l)}$. Take Fig. 4 as an example, the output of the second layer (hidden layer) is

$$\begin{aligned}\begin{bmatrix}\mathbf{v}_1^{(2)}\\ \mathbf{v}_2^{(2)}\\ \mathbf{v}_3^{(2)}\\ \mathbf{v}_4^{(2)}\end{bmatrix} &= \boldsymbol{\varphi}\left(\begin{bmatrix}\mathbf{w}_{11}^{(2)} & \cdots & \mathbf{w}_{14}^{(2)}\\ \vdots & \ddots & \vdots\\ \mathbf{w}_{41}^{(2)} & \cdots & \mathbf{w}_{44}^{(2)}\end{bmatrix}\begin{bmatrix}1\\ \mathbf{u}_1^{(1)}\\ \mathbf{u}_2^{(1)}\\ \mathbf{u}_3^{(1)}\end{bmatrix} + \begin{bmatrix}1\\ \mathbf{b}_1^{(2)}\\ \mathbf{b}_2^{(2)}\\ \mathbf{b}_3^{(2)}\end{bmatrix}\right)\\ &= \boldsymbol{\varphi}\left(\mathbf{W}^{(2)}\mathbf{U}^{(1)} + \mathbf{B}^{(2)}\right)\end{aligned} \tag{5}$$

where the vector $\mathbf{b}_j^{(l)}$ denotes the bias of the j-th node in layer l, and $\mathbf{B}^{(l)}$ represents the bias array of layer l. On this basis, the general expression of the output of layer l can be concluded as

$$\begin{cases}\mathbf{v}_j^{(l)} = \boldsymbol{\varphi}\left(\mathbf{c}_j^{(l)}\right),\\ \mathbf{c}_j^{(l)} = \sum_{l-1} \mathbf{w}_{jk}^{(l)}\mathbf{u}_k^{(l-1)} + \mathbf{b}_j^{(l)},\\ \mathbf{V}^{(l)} = \boldsymbol{\varphi}\left(\mathbf{c}^{(l)}\right) = \boldsymbol{\varphi}\left(\mathbf{W}^{(l)}\mathbf{U}^{(l-1)} + \mathbf{B}^{(l)}\right).\end{cases} \tag{6}$$

MLP utilizes a supervised learning technique called back-propagation, which allows approximate solutions for extremely complex mapping by updating the weights and biases that minimize the errors between the actual network output and the training output recursively until it reaches the preset threshold (loss function) in output layer. In this study, the proposed MLP takes the distal end-effector's position and orientation $\mathbf{x}_e \in \mathbb{R}^6$ as input, and $\mathbf{\Psi}_n$ (let $n = 2$, thus $\mathbf{\Psi}_2 \in \mathbb{R}^4$) as output. Making use of the backpropagation algorithm, the network can stochastically solve the mapping from input data to output data. Levenberg-Marquardt (LM) optimization is employed in backpropagation for its fast convergence in non-linear fitting problems.

B. Data Selection, Preprocessing, and Model Selection

The dataset is generated by means of the forward kinematics as shown in Eq. (3), where the input set $\mathbf{x}_e$ is derived by a randomized set of $\mathbf{\Psi}_2$. The size of dataset is 20,000. The visualized input set $\mathbf{x}_e$, known as robot workspace (orientation has not given) is shown in Fig. 3. Firstly, we divide the dataset into three subsets for cross-validation before network training. The dataset is arbitrarily distributed into training set (70%), validation set (15%), and test set (15%). Training set is used for computing the gradient and updating the network weights and biases. Validation set helps avoiding overfitting problem by supervising the error during the training. Eventually, the test set is employed for model evaluation in term of network performance [25]. The proposed MLP uses hyperbolic tangent function ($\boldsymbol{\varphi}(\mathbf{c}_j) = \tanh(\mathbf{c}_j)$) in the hidden layers as activation function. Hyperbolic tangent function is essentially saturated when the net input is greater than 3 (or approximately 170°).

It necessitates the normalization operation to the dataset as a data pre-processing step before training.

The topology of MLP is determined by two hyperparameters: the number of hidden layers N_H and the number of neurons N_N in each hidden layer. Performance varies from different size of the network. Yet, there is no standard solution to configure the network. The most reliable way to determine these hyperparameters is via systematic tests on specific dataset. Here, we apply the dataset to MLPs with different sizes, with $N_H = \{1, 2, 3, 4, 5\}$ and $N_N = \{6, 7, ..., 20\}$ (To simplify the model, each hidden layer is set to have same number of neurons). The training would cease automatically when either the validation fails in 6 consecutive epochs or the number of epoch reaches the maximum number of 1,000. Performance of different network sizes is assessed by mean squared error (MSE) as shown in Fig. 5.

Fig. 5 demonstrates the MSE values between predicted network output and desired output. The figure shows that the performance increases when the MLP is structured with more than one hidden layer. For multiple hidden layer cases, the MSE value gradually drops with increasing nodes per hidden layer, and tends to become steady when $N_N \geqslant 12$, and local minimum appears for the case where $N_N = 16$ and $N_H = 4$. The result indicates that, when $N_H \in \{3, 4, 5\}$ and $N_N \in \{16, ..., 20\}$, there is no significant difference in terms of MSE which fluctuates between 0.022 and 0.050. However, the computational time increases significantly if the number of layers and nodes are too large. In view of the computational cost and model effectiveness, we select the 6-20-20-20-4 network for further implementation and evaluation. By using an Intel® Core™ i7-8750H CPU at 2.20GHz, the network training process of a dataset size of 20,000 takes about 5 minutes.

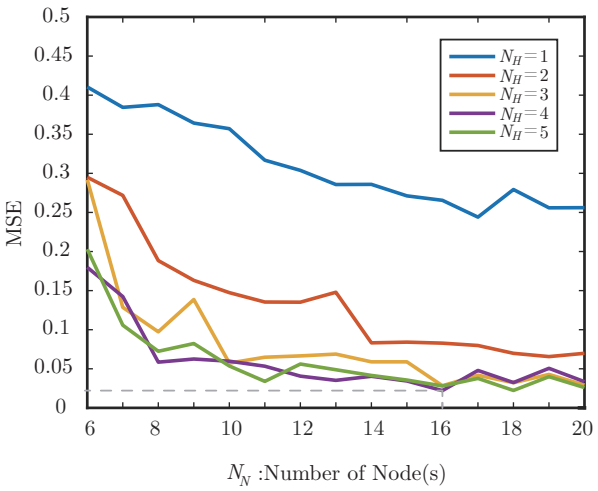

Fig. 5. Performance of MLPs using different network configurations. A total of 20,000 predictions ($\tilde{\mathbf{\Psi}}_2 \in \mathbb{R}^4$) are generated from 20,000 sets of input data ($\mathbf{x}_e \in \mathbb{R}^6$), and the target robot configuration $\mathbf{\Psi}_2$ is the desired vector to be predicted. The within-sample MSE of the predictor is computed as $\text{MSE} = \frac{1}{m}\sum_{i=1}^{m}(\mathbf{\Psi}_{2,i} - \tilde{\mathbf{\Psi}}_{2,i})^2$.

IV. Solver Evaluation and Path Simulation

The proposed MLP architecture, with a network configuration $N_H = 3$ and $N_N = 20$ is adopted for training. Using a normalized data vector which contains 20,000 sets of data, the IK of a dual-segment continuum robot, i.e., mapping from 6 dimensional end-effecter Cartesian to the configuration defined by 4 angles, will be computed accordingly. The trained MLP will enable the path planning for the robot's end-effector using virtual joints. The robot posture in the configuration space can also be visualized.

A. Network Performance

For general input-output fitting problems, the performance of an FNN is usually assessed in various aspects, e.g. error, correlation between target and predicted outputs, and R-Squared.

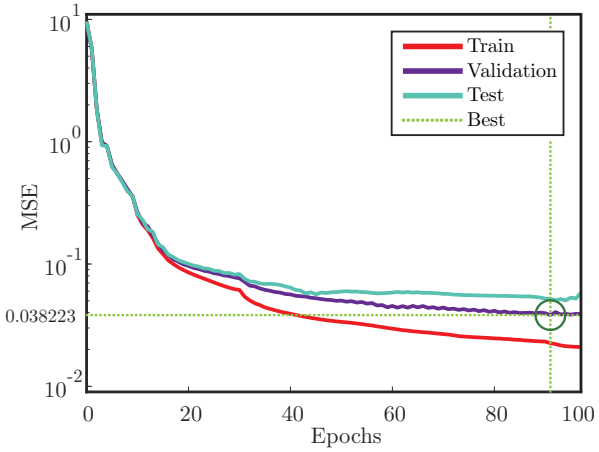

Fig. 6. Training performance of MLP in 6-20-20-20-4 structure. The best validation performance is 0.038223 at epoch 91.

In Fig. 6, the performance of the multilayer network selected in the previous section is shown in the form of diminishing MSE during the training. By applying the LM algorithm, rapid convergence can be observed within 60 epochs. It is noted that the training result varies from one trial to another, as it depends on the subsetting data used for training, validation, and test. Yet, differences are not significant and each training could perform as anticipated. Fig. 7 presents the Pearson correlation coefficient $\mathbf{R}$ between the target and the predicted output vector in three subsets, respectively. The overall $\mathbf{R}$ is greater than 0.99, and p-value $\leqslant 0.001$.

In terms of the robot configuration $\tilde{\mathbf{\Psi}}_2$ that predicted, a certain degree of error appears but not so notable. A simple test is conducted to examine the error of solver outcomes. One hundred new samples of the robot configuration (as target $\mathbf{\Psi}_2$) are used to calculate the end-effector Cartesian via forward kinematics, and these 6-D vectors would be treated as new input sets for the IK solver in order to compute the predicted robot posture. Thereafter, errors are evaluated. Fig. 8 shows the absolute error (left) and relative error (right) between the target and solver-computed robot configuration in 3-D. The boxes presented in Fig. 8 (left) illustrates the

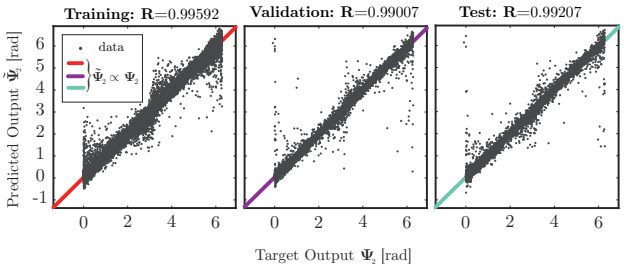

Fig. 7. Regression analysis between the target and predicted output data in three subsets, where $\mathbf{R}$ denotes the coefficient of correlation.

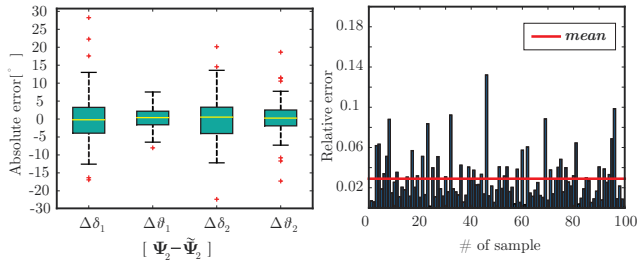

Fig. 8. In the case of a dual-segment continuum robot with the length $L_1 = L_2 = 50$ mm, 100 groups of data are generated to test the IK solver. Left: absolute error [°] of the robot configuration. The yellow line indicates the median level for each parameter. The top and bottom of each green "box" are the 25th and 75th percentiles of the absolute error, respectively. Outliers are displayed with a red "+" sign; Right: relative error from 100 new dataset. The horizontal red line indicates the mean value of relative error is 2.90%.

25th (Q_1) and 75th (Q_3) percentiles of the absolute error between the target output and the predicted output, also known as interquartile ranges. The quartile deviations given by $0.5(Q_3-Q_1)$ are $3.5971°$, $1.8924°$, $3.6712°$, $2.2100°$, and medians are $-0.3454°$, $0.2547°$, $0.3950°$, $0.2580°$, respectively for $\delta_1, \theta_1, \delta_2$, and θ_2 order. Fig. 8 (right) demonstrates the relative error, with an overall average of 2.90%. The result indicates that the error of angle parameters falls in an acceptable range.

B. Virtual Path Planning

A more intuitive evaluation for the solver's performance can be presented in the form of virtual path planning. In some practices, a continuum manipulator for minimally invasive surgery requires a fully control from surgeon who specifies the required position, or even orientation for the end-effector (6-D) in real time. Such user-defined manipulation demands the computation of robot IK. In this part, simulation on path planning w.r.t. end-effecter is conducted to examine the accuracy of the solver. Length of the continuum robot is set as $L_1 = L_2 = 50$ mm as previously mentioned, and the base frame $\mathbf{O}_0$ is assumed to be fixed. Some of the representative paths are first defined for simulation as given in Fig. 9. The robot configuration as the solver's output is shown in Fig. 10. The fitting degree of theoretical path and predicted path reflects the rationality of the solver. End-effector errors between the desired path and the predicted path are evaluated via Euclidean distances in Cartesian space. While substituting the robot configuration that computed from the IK solver into the forward kinematics equations, the simulated end-effector is able to approach to the desired position and orientation in a high accuracy. The result confirms the effectiveness of using a simplified robot model to evaluate robot parameters from the IK solver.

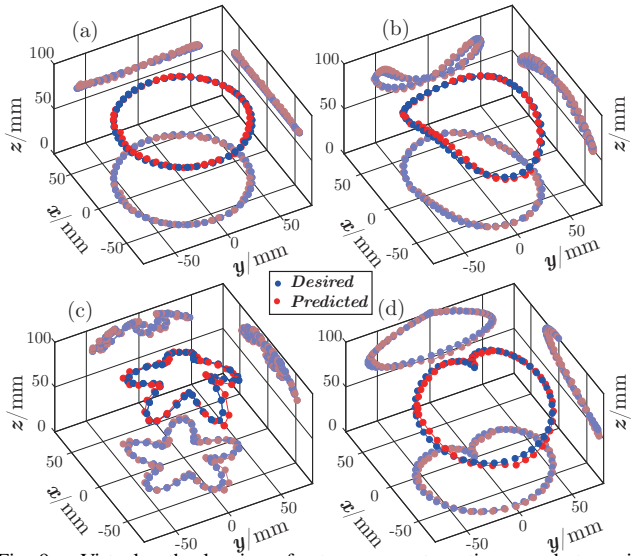

Fig. 9. Virtual path planning of a two-segment continuum robot manipulator. Robot base is fixed at $\mathbf{O}_0 = (0, 0, 0)$, and the total length of the robot is 100 mm. Blue points represent the desired step-by-step position of the end-effector, while red points represent the predicted step-wise position of the end-effector. All paths above are defined in 50 steps. (a) horizontal circle; (b) saddle shape; (c) pentagram; (d) heart shape.

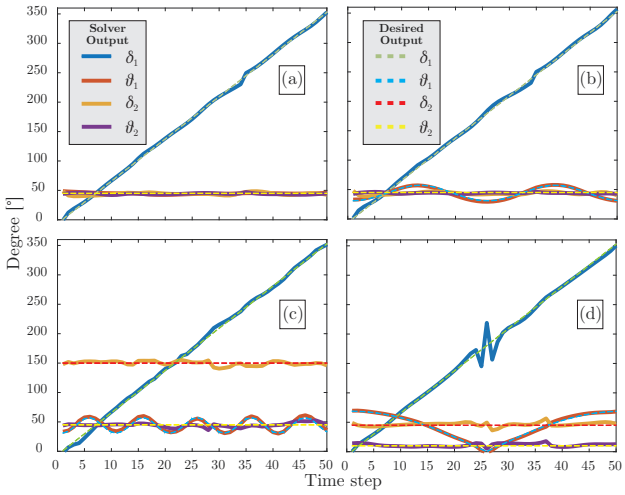

Fig. 10. The solver outputs and the desired outputs w.r.t. the path given in Fig. 9. The predicted paths can be calculated the solver outputs and corresponding to forward kinematics equations.

C. Maintaining Tip Orientation along a Desired Path

Similar to many robotic applications, it is desirable to maintain the orientation of the end-effector when moving along a planned path. This proposed solver is capable of finding a solution which can keep the pointing direction of the continuum robot unchanged. Employing the same idea, we examine it by training a simpler solver using only robot configuration $\mathbf{\Psi}_2 = [\delta_1, \theta_1, \delta_2, \theta_2]^\top$ and $\mathbf{x}_e = [x_e, y_e, z_e, \alpha_e]^\top$,

where α_e represents the tip pointing direction. As shown in Fig. 11, a circle path with the desired pointing direction is used as the training set, and the circle path, whose z-axis is -10 mm away from its original height, is used for testing. Based on the robot configuration obtained from the model, the actual pointing direction of the continuum robot can be computed directly through the transformation matrix for comparison. The result shows that the robot can keep the pointing direction while moving in a circle path.

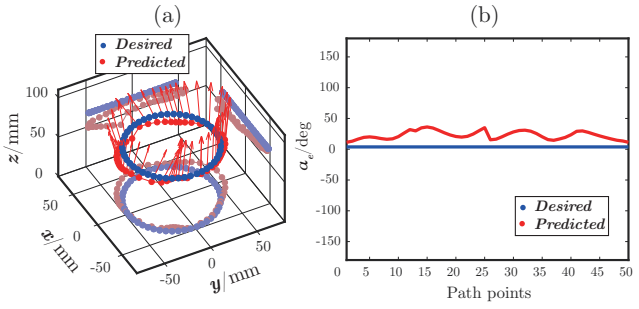

Fig. 11. (a) Desired path and solver-generated (predicted) path/orientation; (b) tip pointing direction in euler angle α_e.

V. Conclusions

Continuum robot manipulators offer flexibility and convenience to a varity of tasks, but increase complexity for control and analysis. In this paper, a simplified model is proposed for a multi-segment robot manipulator using virtual links and joints, enabling visualization of the robot configuration as conventional rigid-link robots. The inverse kinematic of the proposed robot model is obtained through supervised learning. Forward kinematic equations are first derived using D-H conventions and matrix transformation, and datasets containing the robot end-effector in Cartesian space and the virtual joint space are prepared accordingly. An MLP is then constructed as the IK solver, and the performance is evaluated under different model parameters. From the simulation, a mean squared error of 0.022 can be achieved. The trained MLP is then used to generate the required inputs for different trajectories. Comparing the computed and the desired end-effector positions, the error using the IK solver is between 2% to 4%, which is acceptable in many applications. The method can also be extended in 3-segment cases. Thus, the proposed model and the IK solver offers a simpler approach to control and visualize a multi-segment continuum robot manipulator.

References

[1] G. S. Chirikjian, "A continuum approach to hyper-redundant manipulator dynamics," in *Proceedings of 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '93)*, vol. 2, July 1993, pp. 1059–1066 vol.2.

[2] G. Robinson and J. B. C. Davies, "Continuum robots - a state of the art," in *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, vol. 4, May 1999, pp. 2849–2854 vol.4.

[3] D. B. Camarillo, C. F. Milne, C. R. Carlson, M. R. Zinn, and J. K. Salisbury, "Mechanics modeling of tendon-driven continuum manipulators," *IEEE Transactions on Robotics*, vol. 24, no. 6, pp. 1262–1273, Dec 2008.

[4] M. M. Dalvand, S. Nahavandi, and R. D. Howe, "An analytical loading model for n-tendon continuum robots," *IEEE Transactions on Robotics*, vol. 34, no. 5, pp. 1215–1225, Oct 2018.

[5] 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, Oct 2014.

[6] W. McMahan, V. Chitrakaran, M. Csencsits, D. Dawson, I. D. Walker, B. A. Jones, M. Pritts, D. Dienno, M. Grissom, and C. D. Rahn, "Field trials and testing of the octarm continuum manipulator," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, May 2006, pp. 2336–2341.

[7] Z. Guo, Z. Dong, K. Lee, C. L. Cheung, H. Fu, J. D. L. Ho, H. He, W. Poon, D. T. Chan, and K. Kwok, "Compact design of a hydraulic driving robot for intraoperative mri-guided bilateral stereotactic neurosurgery," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2515–2522, July 2018.

[8] R. J. Webster, J. M. Romano, and N. J. Cowan, "Mechanics of precurved-tube continuum robots," *IEEE Transactions on Robotics*, vol. 25, no. 1, pp. 67–78, Feb 2009.

[9] Y. Kim, S. S. Cheng, M. Diakite, R. P. Gullapalli, J. M. Simard, and J. P. Desai, "Toward the development of a flexible mesoscale mri-compatible neurosurgical continuum robot," *IEEE Transactions on Robotics*, vol. 33, no. 6, pp. 1386–1397, Dec 2017.

[10] H. Wang, J. Chen, H. Y. K. Lau, and H. Ren, "Motion planning based on learning from demonstration for multiple-segment flexible soft robots actuated by electroactive polymers," *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 391–398, Jan 2016.

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

[12] B. A. Jones and I. D. Walker, "Kinematics for multisection continuum robots," *IEEE Transactions on Robotics*, vol. 22, no. 1, pp. 43–55, Feb 2006.

[13] S. Tejomurtula and S. Kak, "Inverse kinematics in robotics using neural networks," *Information Sciences*, vol. 116, no. 2, pp. 147 – 164, 1999.

[14] 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 *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nov 2013, pp. 5033–5039.

[15] G. S. Chirikjian, "A general numerical method for hyper-redundant manipulator inverse kinematics," in *[1993] Proceedings IEEE International Conference on Robotics and Automation*, May 1993, pp. 107–112 vol.3.

[16] M. W. Hannan and I. D. Walker, "Kinematics and the implementation of an elephant's trunk manipulator and other continuum style robots," *Journal of Robotic Systems*, vol. 20, no. 2, pp. 45–63, 2003.

[17] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of control, signals and systems*, vol. 2, no. 4, pp. 303–314, 1989.

[18] R. Hecht-Nielsen, "Neurocomputing: picking the human brain," *IEEE spectrum*, vol. 25, no. 3, pp. 36–41, 1988.

[19] B. B. Choi and C. Lawrence, "Inverse kinematics problem in robotics using neural networks," *NASA Technical Memorandum*, no. 143125, p. 24, 1992.

[20] S. Tejomurtula and S. Kak, "Inverse kinematics in robotics using neural networks," *Information Sciences*, vol. 116, no. 2, pp. 147 – 164, 1999.

[21] T. George Thuruthel, E. Falotico, M. Manti, A. Pratesi, M. Cianchetti, and C. Laschi, "Learning closed loop kinematic controllers for continuum manipulators in unstructured environments," *Soft Robotics*, vol. 4, no. 3, pp. 285–296, 2017.

[22] R. Grassmann, "Learning the forward and inverse kinematics of a 6-dof concentric tube continuum robot in *SE(3)*," 2018.

[23] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, "From data mining to knowledge discovery in databases," *AI magazine*, vol. 17, no. 3, p. 37, 1996.

[24] K. M. Lynch and F. C. Park, *Modern Robotics*. Cambridge University Press, 2017.

[25] S. S. Haykin, *Neural networks and learning machines*, 3rd ed. Pearson Upper Saddle River, 2009.