Learning-Based Forward Kinematics and Its Inverse Kinematics Implementation for Concentric-Tube Robots

Chao Zhang[1], Shuang Song[1,*], Jiaole Wang[2,*]

***Abstract*—Concentric-tube robots (CTRs) offers great promise in minimally invasive surgery due to their compact size and inherent hollow channels. Recent learning-based methods have achieved excellent accuracy in forward kinematics (FK) estimation from real-world data. However, these models are often complex and non-differentiable, hindering rapid computation and further derivations. Additionally, direct learning of inverse kinematics (IK) remains challenging. This paper presents a precise, explicit, and model-free kinematic method for CTRs, including a refined learning-based FK approach and an IK root-finding method using the learned FK functions. We first design a boundary-dense sampling strategy to collect CTR configuration data from a CTR prototype. Then, using fully-connected feedforward neural network with tangent hyperbolic activation function, we learn and extract explicit FK function. For IK, we apply the Newton-Raphson method, with the Jacobian derived by direct differentiation of the FK function. We also introduce an initialization strategy to provide initial guesses and handle infeasible CTR configurations for IK iterations. Comprehensive experiments demonstrate that our FK method provided concise, differentiable functions while maintaining accuracy comparable to state-of-art learning-based FK methods (with an average tip position error of about 0.5% of the robot's length) and achieved high FK computation speeds of up to 158,742 Hz. We demonstrated precise IK computation and successfully addressed tasks such as path following and multi-solution calculation.**

***Index Terms*—Concentric-tube robots, learning-based forward kinematics, inverse kinematics, minimally invasive surgery.**

I. Introduction

CONCENTRIC-TUBE robots (CTRs) are ideally suitable for medical diagnosis and minimally invasive surgery because of the natural hollow channel and compact profile with relatively high stiffness [1], [2] . As shown in Fig. 1 a), a CTR consists of several telescoped pre-curved super-elastic Nitinol tubes, and each tube can be translated and rotated relative to each other. By controlling the kinematic inputs acting on the proximal end of each tube, it produces specific shapes due to the elastic interaction between the tubes [3].

Establishing the kinematics model of CTR, including the forward kinematics (FK) and inverse kinematics (IK), is a fundamental problem for its control and planning. To describe the shape of CTR for a given kinematic input, two physical-based FK models were presented. One is the torsionally rigid model which neglects the torsion of tubes and models the shape of CTR as the weighted curvature superposition of the individual pre-curved tubes [4]. It is relatively easy to implement and fast to compute, but has poor accuracy. Another is the torsionally compliant model, which significantly enhances modeling accuracy (approximating the tip position within 3% of total robot length) but substantially increases both the complexity of modeling and computational demands. The method involves: 1) Segmenting CTR based on tube overlap conditions. 2) Establishing geometric relationships between curvature and torsion at each arc cross-section. 3) Introducing the Constitutive Equation and Cosserat Rod Model to characterize the relationship between tube forces and deformations, thereby deriving second-order differential equations for each segment. 4) Solving these equations using boundary conditions to obtain curvature at each arc cross-section. 5) Integrating curvatures to obtain the Frenet-Serret frame rotations. 6) Integrating the frame rotations to ultimately determine the shape of CTR. As demonstrated, the combination of multiple complex differential operations and two successive integration steps makes FK computation extremely time-consuming, which renders the model impractical for real-

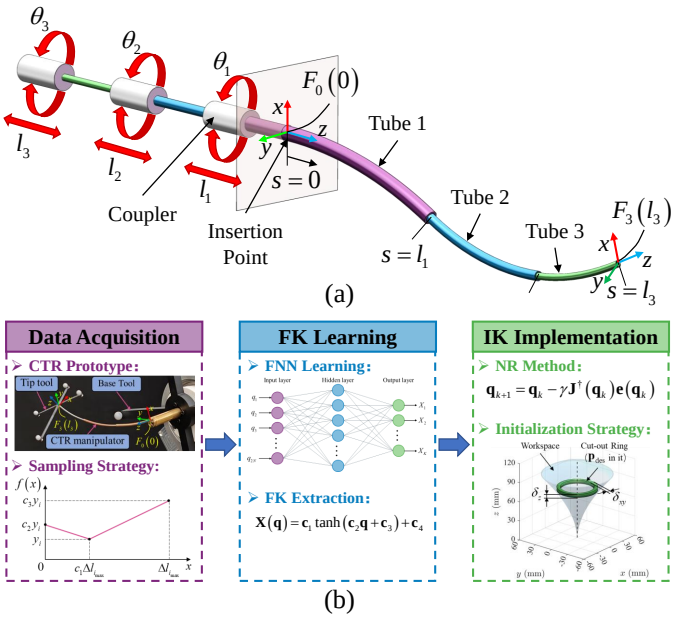


Fig. 1. (a) A three-tube concentric-tube robot (CTR). By controlling the rotations and translations of each tube, it will generates specific shapes due to the elastic interaction between the tubes. (b) Framework of our method. 1) Data Acquisition, utilizing a boundary-dense sampling strategy to acquire CTR configurations from a prototype; 2) Forward Kinematics (FK) Learning, employing neural network to learn the FK model and extracting FK functions; and 3) Inverse Kinematics (IK) Implementation, solving IK using the Newton-Raphson (NR) method with an initialization strategy.

This work was supported in part by the National Key Research and Development Program of China under Grant 2022YFB4703201, in part by the Talent Recruitment Project of Guangdong under Grant 2021QN02Y839, and in part by the Science and Technology Innovation Committee of Shenzhen under Grant JCYJ20220818102408018.

[1]Chao Zhang and Shuang Song are with the School of Robotics and Advanced Manufacturing, Harbin Institute of Technology (Shenzhen), Shenzhen 518055, China.

[2]Jiaole Wang are with the School of Biomedical Engineering and Digital Health, Harbin Institute of Technology (Shenzhen), Shenzhen 518055, China.

*Corresponding authors: Jiaole Wang wangjiaole@hit.edu.cn and Shuang Song songshuang@hit.edu.cn

time applications [5], [6]. As for the analytical-based IK solving of CTR, it remains challenging at present [3].

Towards real-time position control of CTR, Dupont *et al.* used multidimensional truncated Fourier series to fit the FK model using pre-calculated model-based FK solutions, and then used Gauss-Newton algorithm to compute the IK [5]. Girerd *et al.* further considered the elastic stability, workspace limit, and reinitialization of infeasible configurations of CTR to make the method more robust [7]. Because the time-consuming torsionally compliant model is fitted, the computational efficiency of FK can be greatly promoted. Furthermore, the root-finding algorithm in IK exhibits accelerated convergence (quadratic) in comparison to alternative differential kinematics-based approaches [8]–[11]. Therefore, the truncated Fourier series method enables the computation of IK at a frequency of up to 1000 Hz, rendering it suitable for online and offline applications [5]. However, the data used for fitting is generated from the physical model implementation, which indicates the accuracy of the fitted FK will not exceed that of the model itself. Besides, the Gibbs phenomenon of truncated Fourier series introduces additionally detrimental impact on the fitting accuracy [12].

In recent years, learning-based approaches have been utilized in kinematic modeling of CTRs. Bergeles *et al.* were the first to learn the FK and IK of a specific CTR using multi-layer perceptron [13]. Grassmann *et al.* improved the joint and pose representations of CTR to promote the training performance of FK [14], [15]. Kuntz *et al.* learned the full shape of CTR using a mixture of simulated and measured data by deep neural networks [16], and Liang *et al.* learned IK of CTR from shape as input [17]. In addition, Iyengar *et al.* introduced deep reinforcement learning to the CTR kinematic control and path following by directly learning a control policy [18]–[20]. Overall, learning the CTR FK has achieved high accuracy (approximating the tip position within 1% with respect to the robot length), but the learned FK models are often complex and non-differentiable, making them difficult to be used for further derivation and fast computation. Moreover, learning the IK for a general CTR that relies solely on tip pose information remains a significant challenge [3], [21].

In this article, a precise, explicit, and model-free kinematic method for CTRs is proposed, including a refined learning-based FK approach and an IK root-finding method using the learned FK functions, rather than directly learning the IK. First, a boundary-dense sampling strategy is designed to acquire a dataset of actual CTR configurations from a CTR prototype. This strategy mitigates the problem inherent in the widely used uniform sampling method [14]–[16], [21], which results in an overly dense distribution of tip position samples in the center of the workspace, but sparse at the boundaries. Then, a fully-connected feedforward neural network (FNN) using tangent hyperbolic function as the activation function is employed to learn the FK model, from which the explicit form of the learned FK model is extracted based on the network structure, thereby making the resulting FK function concise and differentiable. After that, the CTR IK is solved using the Newton-Raphson method, wherein the Jacobian is obtained by directly differentiating the extracted FK functions. To provide suitable initial values and deal with infeasible CTR configurations during the IK iterations, an initialization strategy is devised. Finally, comprehensive experiments have been conducted to evaluate the proposed CTR kinematic method, including both FK learning and IK solving performances.

The main contributions are: 1) an FK learning and extraction method for CTRs to get explicit FK functions that are precise, computationally efficient, and differentiable; 2) a boundary-dense data sampling strategy to collect dataset of actual CTR configurations from CTR prototype; 3) an initialization strategy to provide suitable initial guesses and handle infeasible CTR configurations during IK iterations.

II. Data Acquisition

A. Representation of Configurations

1) Joint Space Representation: As shown in Fig. 1 a), for a CTR with N tubes, each tube has two degrees of freedom (rotation and translation), its kinematic input can be expressed with $2N$ variables as

$$\mathbf{q} = [\theta_1, l_1, \cdots, \theta_N, l_N]^{\mathrm{T}} , \tag{1}$$

where θ_i and l_i denote the i-th tube rotation angle at its proximal end and the extended length from the outlet, respectively. The subscript indices $i = 1, \cdots, N$ refer to individual tubes with tube 1 being the outermost and tube N being the innermost. We do not use the wildly used trigonometric representation of θ_i, which converts θ_i into $[\cos\theta_i, \sin\theta_i]$ to handle the periodicity of tube rotation [14], because it will introduce intermediate variables, adding extra complexity to the subsequent utilization of the final FK functions.

For a given set of joint values, the tube rotations are independent of each other, and we constrain them to $(-\pi, \pi]$ to eliminate the periodicity in rotation. For the tube translation variables, they are interdependent that each tube should always be aligned with or extended by a limited length relative to the adjacent outer tube. Therefore, the joint variables has to satisfy the inequalities

$$\begin{cases} \theta_i \in (-\pi, \pi] \\ l_i \in [0, l_{i_{\max}}] \\ l_1 \leq \cdots \leq l_N \end{cases} , \tag{2}$$

in which $l_{i_{\max}}$ represents the designed maximum extended length of i-th tube relative to the outlet.

2) Pose Representation in Task Space: As illustrated in Fig. 1 a), the tip pose of the innermost tube (attached frame $F_N\,(l_N)$) relative to the fixed base frame $F_0\,(0)$ serves as the tip pose of the CTR, and we use

$$\mathbf{p} = [p_x, p_y, p_z]^{\mathrm{T}} \tag{3}$$

to represent the tip position. Since the CTR is inherently hollow, practical applications typically concern with its axial pointing direction at the tip [5]. Therefore, the tip tangent vector is employed to represent the tip orientation of the CTR for its clearly physical significance, that is

$$\mathbf{t} = [t_x, t_y, t_z]^{\mathrm{T}} . \tag{4}$$

Then, the tip pose of CTR can be expressed as $\mathbf{T} = [\mathbf{p}; \mathbf{t}]$, and the CTR configuration can be written as $\mathbf{c} = [\mathbf{q}; \mathbf{T}]$.

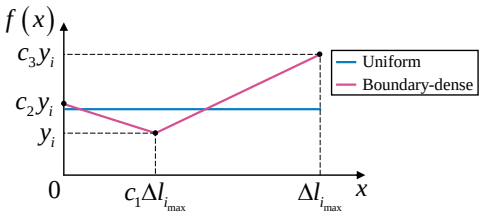


Fig. 2. Probability density function of the uniform and the proposed boundary-dense distributions for translation variables.

B. Data Acquisition using Boundary-Dense Sampling Strategy

To learn the FK model, it is essential to have a dataset containing sufficient CTR configurations, each consisting of a kinematic input $\mathbf{q}$ and its corresponding tip pose $\mathbf{T}$. Although collecting data from a real CTR requires additional sensing setups and entails some sampling time, it allows to obtain real-world data, which is fundamental for learning a precise FK model. In contrast, simulated data fails to account for errors introduced by the actuation system, friction, clearances, and inaccuracies in physical parameters like tube material elastic modulus and pre-curvatures, resulting in significant discrepancies between simulated and real data.

For the sampling of the kinematic inputs, the rotation components in each sample is generated obeying a uniform distribution $\mathcal{U}$ as

$$\theta_i = \mathcal{U}(-\pi, \pi) \ . \tag{5}$$

Existing data-driven CTR kinematics methods uniformly sample each translation variable based on the relative extension lengths between tubes. Because of the interdependencies among these translation variables, multiple uniform distributions will be superimposed. However, the convolution principle will make the combined distributions no longer conform to a uniform distribution. Additionally, since the CTR workspace is typically flared, the uniform sampling strategy exacerbates the uneven distribution of samples, resulting in extremely dense at the center but sparse at the boundaries.

To mitigate this problem, a boundary-dense sampling strategy for the translation variables is proposed as

$$\begin{cases} l_1 = \mathcal{B}(0, \Delta l_{1_{\max}}) \\ l_i = l_{i-1} + \mathcal{B}(0, \Delta l_{i_{\max}}) \end{cases}, \tag{6}$$

where $\Delta l_{i_{\max}} = l_{i_{\max}} - l_{i-1_{\max}}$ denotes the maximum extended length of i-th tube relative to its neighboring outer tube, and $\Delta l_{1_{\max}} = l_{1_{\max}}$. The probability density function (PDF) of the proposed boundary-dense distribution $\mathcal{B}$ is shown in Fig. 2, in which the sampling density at both ends of the interval $[0, \Delta l_{i_{\max}}]$ is increased in order to achieve a more even distribution of the CTR tip positions in the workspace. The PDF of $\mathcal{B}$ is a combination of two linear functions as

$$f(x) = \begin{cases} \dfrac{(1-c_2)\, y_i}{c_1 \Delta l_{i_{\max}}} x + c_2 y_i, \ x \in [0, c_1 \Delta l_{i_{\max}}) \\ \dfrac{(c_3 - 1)\, y_i}{(1-c_1)\, \Delta l_{i_{\max}}} x + \dfrac{(1 - c_1 c_3)\, y_i}{1 - c_1}, \ x \in [c_1 \Delta l_{i_{\max}}, \Delta l_{i_{\max}}] \end{cases} \tag{7}$$

in which c_1, c_2, and c_3 are coefficients used to fine-tune the shape of the PDF, and $y_i = 2/(\Delta l_{i_{\max}}(1 + c_3 + c_1 c_2 - c_1 c_3))$ according to the normalization property of PDF.

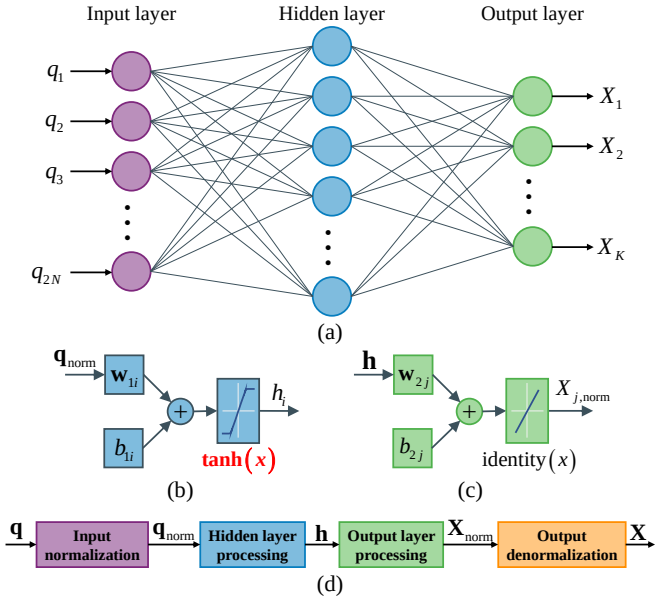


Fig. 3. The structure of the fully-connected feedforward neural network with a single hidden layer. (a) is the network structure. (b) is the structure of hidden layer node. (c) is the structure of output layer node.

Using the above data representation and sampling strategy, a dataset of kinematic inputs can be generated. Although it is feasible to send each kinematic input to the CTR prototype, we can optimize the order of these inputs by formulating the cumulative motor movements as a Traveling Salesman Problem (TSP) to significantly reduce sampling time [21]. Utilizing suitable pose sensing equipment, such as optical or electromagnetic tracking systems, we can obtain the required dataset $\mathcal{P}$ of CTR configurations by actuating the CTR prototype according to the sorted sequence of the kinematic inputs and collecting the pose data after each movement.

III. Forward Kinematics Learning and Extraction

A. Learning FK using FNN

Using learning-based methods to approximate the FK of CTRs has proven effective for obtaining precise FK models due to the powerful nonlinear fitting capabilities of neural networks [13]–[17], [21]. However, these methods often yield FK models that are complex and non-differentiable, hindering their further mathematical operations. According to Kolmogorov's theorem, any reasonable function can be approximated arbitrarily well using an FNN with a single hidden layer [22]. Therefore, as depicted in Fig. 3, we employ an FNN with only one hidden layer to learn the FK of CTRs, providing sufficient fitting capability without unnecessary complexity. To ensure the learned FK function is differentiable, the tangent hyperbolic function $\tanh(x)$ is used in the hidden layer instead of non-differentiable functions like ReLU.

As shown in Fig. 3 (a), the input $\mathbf{q}$ is represented by $2N$ nodes in the input layer, corresponding to each element of $\mathbf{q}$. The hidden layer contains M nodes using $\tanh$ activation function to introduce nonlinearity for the network. The output of the network can be either $\mathbf{p}$, $\mathbf{t}$, or $\mathbf{T}$, and the output layer has K nodes, where identity (x) is the activation function. By

training the network with the collected dataset $\mathcal{P}$, we obtain a learned CTR FK function and the corresponding network parameters: weight $\mathbf{w}_1 \in \mathbb{R}^{M\times 2N}$ and bias $\mathbf{b}_1 \in \mathbb{R}^{M\times 1}$ in the hidden layer, and weight $\mathbf{w}_2 \in \mathbb{R}^{K\times M}$ and bias $\mathbf{b}_2 \in \mathbb{R}^{K\times 1}$ in the output layer.

B. Extraction of Learned FK Function

Based on the FNN's structure, we can extract the explicit FK function for the CTR, which allows convenient mathematical operations like Jacobian derivation and enables efficient, cross-platform implementation – for instance, coding the FK function in C/C++ to speed up computations. According to the FNN data workflow, as illustrated in Fig. 3 (d), there are four steps from the kinematic input $\mathbf{q}$ to the final output $\mathbf{X}$: input normalization, hidden layer processing, output layer processing, and output denormalization.

Firstly, each component q_i in the kinematic input $\mathbf{q}$ will be normalized in the interval $[-1, 1]$ as

$$q_{i_{\text{norm}}} = 2\left(q_i - q_{i_{\min}}\right)/\left(q_{i_{\max}} - q_{i_{\min}}\right) - 1\,, \tag{8}$$

where $q_{i_{\min}}$ and $q_{i_{\max}}$ are the minimum and maximum of all sampled q_i used for training in dataset $\mathcal{P}$. Then, the normalized input $\mathbf{q}_{\text{norm}}$ can be written as

$$\mathbf{q}_{\text{norm}} = \mathbf{a}_1\mathbf{q} + \mathbf{a}_2\,, \tag{9}$$

where $\mathbf{a}_1 = \operatorname{diag}\left\{\frac{2}{q_{1\max}-q_{1\min}}, \cdots, \frac{2}{q_{2N\max}-q_{2N\min}}\right\}$ and $\mathbf{a}_2 = \left[-\frac{q_{1\max}+q_{1\min}}{q_{1\max}-q_{1\min}}, \cdots, -\frac{q_{2N\max}+q_{2N\min}}{q_{2N\max}-q_{2N\min}}\right]^{\mathrm{T}}$.

Secondly, as shown in Fig. 3 (b), $\mathbf{q}_{\text{norm}}$ is passed to each node of the hidden layer, and it will multiply the weight $\mathbf{w}_1$ and add the bias $\mathbf{b}_1$, yielding

$$\mathbf{h} = \tanh\left(\mathbf{w}_1\mathbf{q}_{\text{norm}} + \mathbf{b}_1\right)\,, \tag{10}$$

where $\mathbf{h} \in \mathbb{R}^{M\times 1}$ is the hidden layer representation, and the activation function $\tanh(x) = \left(e^x - e^{-x}\right)/\left(e^x + e^{-x}\right)$.

Thirdly, as depicted in Fig. 3 (c), $\mathbf{h}$ is transmitted to each node of the output layer, which then outputs as

$$\mathbf{X}_{\text{norm}} = \mathbf{w}_2\mathbf{h} + \mathbf{b}_2\,. \tag{11}$$

In the last step, the output layer representation $\mathbf{X}_{\text{norm}}$ will be denormalized to the final output $\mathbf{X}$, and each component in $\mathbf{X}$ is given by

$$X_j = \left(X_{j_{\text{norm}}} + 1\right)\left(X_{j_{\max}} - X_{j_{\min}}\right)/2 + X_{j_{\min}}\,, \tag{12}$$

in which $X_{j_{\max}}$ and $X_{j_{\min}}$ are the maximal and minimal values of all sampled X_j for training in dataset $\mathcal{P}$, respectively. The denormalized output can be written as

$$\mathbf{X} = \mathbf{a}_3\mathbf{X}_{\text{norm}} + \mathbf{a}_4\,, \tag{13}$$

where $\mathbf{a}_3 = \operatorname{diag}\left\{\frac{X_{1\max}-X_{1\min}}{2}, \cdots, \frac{X_{K\max}-X_{K\min}}{2}\right\}$ and $\mathbf{a}_4 = \left[\frac{X_{1\max}+X_{1\min}}{2}, \cdots, \frac{X_{K\max}+X_{K\min}}{2}\right]^{\mathrm{T}}$.

Finally, from (9), (10), (11), and (13), the learned CTR FK function can be extracted in an explicit form as

$$\mathbf{X}\left(\mathbf{q}\right) = \mathbf{c}_1\tanh\left(\mathbf{c}_2\mathbf{q} + \mathbf{c}_3\right) + \mathbf{c}_4\,, \tag{14}$$

where $\mathbf{c}_1 = \mathbf{a}_3\mathbf{w}_2 \in \mathbb{R}^{K\times M}$, $\mathbf{c}_2 = \mathbf{w}_1\mathbf{a}_1 \in \mathbb{R}^{M\times 2N}$, $\mathbf{c}_3 = \mathbf{w}_1\mathbf{a}_2 + \mathbf{b}_1 \in \mathbb{R}^{M\times 1}$, and $\mathbf{c}_4 = \mathbf{a}_3\mathbf{b}_2 + \mathbf{a}_4 \in \mathbb{R}^{K\times 1}$. In practice, $\mathbf{X}$ can be either $\mathbf{p}$, $\mathbf{t}$, or $\mathbf{T}$, as desired.

IV. Inverse Kinematics based on Learned FK

A. Root-Finding of Inverse Kinematics

Given a desired tip pose $\mathbf{T}_{\text{des}}$, the IK problem for CTR can be formulated as a root-finding problem to find $\mathbf{q}$ such that $\mathbf{T}(\mathbf{q}) - \mathbf{T}_{\text{des}} = 0$. To solve this, the Newton-Raphson (NR) method is employed, which offers quadratic convergence and is straightforward to implement.

First, the error of pose can be defined as

$$\mathbf{e} = \mathbf{W}_e\left(\mathbf{T}\left(\mathbf{q}\right) - \mathbf{T}_{\text{des}}\right)\,, \tag{15}$$

in which $\mathbf{W}_e = \operatorname{diag}\left\{w_{e_1}, \cdots, w_{e_K}\right\}\left(w_{e_i} > 0\right)$ is the weight matrix to reflect the priority level of position and orientation errors and also to absorb the difference of the metrics between position and orientation.

Then, on the basis of the learned FK function (14), the Jacobian can be directly gotten by derivation of (15) as

$$\mathbf{J}\left(\mathbf{q}\right) = \mathbf{W}_e\frac{\partial\mathbf{T}\left(\mathbf{q}\right)}{\partial\mathbf{q}}\,. \tag{16}$$

The numerical iteration process of NR method follows

$$\mathbf{q}_{k+1} = \mathbf{q}_k - \gamma\mathbf{J}^{\dagger}\left(\mathbf{q}_k\right)\mathbf{e}\left(\mathbf{q}_k\right)\,, \tag{17}$$

in which $\mathbf{J}^{\dagger}$ denotes the pseudoinverse of Jacobian and $\mathbf{J}^{\dagger} = \mathbf{J}^{\mathrm{T}}\left(\mathbf{J}\mathbf{J}^{\mathrm{T}}\right)^{-1}$, and $\gamma \in (0, 1)$ is a coefficient that controls the iterative step size of the algorithm.

Ideally, $\|\mathbf{e}\left(\mathbf{q}_k\right)\|$ will converge to a small threshold ϵ after several iterations, at which point the kinematic input $\mathbf{q}_{\text{des}}$ corresponding to the desired pose $\mathbf{T}_{\text{des}}$ can be determined.

B. Initialization and Reinitialization of Configurations

In the NR method, as with many other numerical methods, improper selection of initial values $\mathbf{q}_0$ can lead to divergence of the iterations. Additionally, even with careful initial value selection, the updated joint variables may still exceed the joint limits (2) during the iterations.

To address these issues, an initialization strategy for CTR configurations is proposed. The key to this strategy is to generate a candidate subset of CTR configurations $\mathcal{P}_c$ based on $\mathbf{T}_{\text{des}}$, and the generation of $\mathcal{P}_c$ consists of three main steps.

The first step, see in Fig. 4, is to cut out a ring containing the desired tip position $\mathbf{p}_{\text{des}}$ in the CTR workspace. The upper and lower planes of the ring are at $z = p_{\text{des},z} \pm \delta_z/2$, and its inner and outer radii are given by $r = \|[\mathbf{e}_x, \mathbf{e}_y, \mathbf{0}]\,\mathbf{p}_{\text{des}}\| \pm \delta_{xy}/2$, where δ_z and δ_{xy} are small tolerances, $\mathbf{e}_x$ and $\mathbf{e}_y$ are the standard basis vectors in the x and y directions, respectively. CTR configurations from $\mathcal{P}$ whose positions lie within the ring will be further used to get the candidate subset $\mathcal{P}_c$.

In the second step, see in Fig. 4 (b), each CTR configuration $[\mathbf{q}; \mathbf{p}; \mathbf{t}]$ is rotated around the z-axis by an angle of β to the cross-section where $\mathbf{p}_{\text{des}}$ is located, yielding

$$\begin{cases} \mathbf{q}^r = [\theta_1 + \beta, l_1, \cdots, \theta_N + \beta, l_N]^{\mathrm{T}} \\ \mathbf{p}^r = \mathbf{R}_z\left(\beta\right)\mathbf{p} \\ \mathbf{t}^r = \mathbf{R}_z\left(\beta\right)\mathbf{t} \end{cases}\,, \tag{18}$$

where $\mathbf{R}_z\left(\beta\right)$ is the rotation matrix around the z-axis. If $(\theta_i + \beta)$ exceeds the interval $(-\pi, \pi]$, we adjust it by adding or

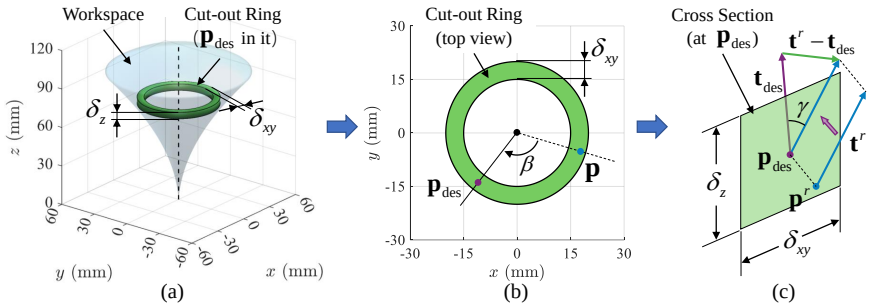


Fig. 4. Candidate subset selection strategy of CTR configurations for initialization in IK computation. (a) We fist cut out a ring from the CTR workspace using the desired tip position $\mathbf{p}_{\text{des}}$ as the anchor point. (b) All CTR configurations within the ring are rotated around the z-axis to align with the cross-section where $\mathbf{p}_{\text{des}}$ is located. (c) The rotated configurations are further filtered to select those whose orientation $\mathbf{t}^r$ closely match the desired orientation $\mathbf{t}_{\text{des}}$.

subtracting 2π to ensure it lies within the interval. To compute β, see in Fig. 4 (c), we derive the following expression since $\mathbf{p}^{\mathbf{r}}$ and $\mathbf{p}_{\text{des}}$ are located within the same cross-section as

$$\frac{[\mathbf{e}_x, \mathbf{e}_y, \mathbf{0}]\, \mathbf{p}_{\text{des}}}{\|[\mathbf{e}_x, \mathbf{e}_y, \mathbf{0}]\, \mathbf{p}_{\text{des}}\|} = \frac{[\mathbf{e}_x, \mathbf{e}_y, \mathbf{0}]\, \mathbf{p}^r}{\|[\mathbf{e}_x, \mathbf{e}_y, \mathbf{0}]\, \mathbf{p}^r\|}. \tag{19}$$

From (18) and (19), we can get

$$\begin{cases} \sin\beta = \dfrac{p_x \cdot p_{\text{des},y} - p_y \cdot p_{\text{des},x}}{\sqrt{p_x^2 + p_y^2} \cdot \sqrt{p_{\text{des},x}^2 + p_{\text{des},y}^2}} \\ \cos\beta = \dfrac{p_x \cdot p_{\text{des},x} + p_y \cdot p_{\text{des},y}}{\sqrt{p_x^2 + p_y^2} \cdot \sqrt{p_{\text{des},x}^2 + p_{\text{des},y}^2}} \end{cases}, \tag{20}$$

and then β can be determined as $\beta = \text{atan2}(\sin\beta, \cos\beta)$.

In the third step, as shown in Fig. 4 (c), we further select CTR configurations whose rotated orientations $\mathbf{t}^r$ closely align with the desired orientation $\mathbf{t}_{\text{des}}$. The difference between $\mathbf{t}^r$ and $\mathbf{t}_{\text{des}}$ is given by $\|\mathbf{R}_z(\beta)\,\mathbf{t} - \mathbf{t}_{\text{des}}\|$, which equals to $2\sin(\gamma/2)$, where γ is the angle between $\mathbf{t}^r$ and $\mathbf{t}_{\text{des}}$.

After completing these three steps, the final candidate subset can be obtained as

$$\begin{aligned} \mathcal{P}_c = \{\mathbf{c} \in \mathcal{P} \mid & \left\|(\mathbf{p} - \mathbf{p}_{\text{des}})\, \mathbf{e}_z^{\mathrm{T}}\right\| \le \delta_z/2, \text{ and} \\ & \text{abs}\left(\|[\mathbf{e}_x, \mathbf{e}_y, \mathbf{0}]\, \mathbf{p}\| - \|[\mathbf{e}_x, \mathbf{e}_y, \mathbf{0}]\, \mathbf{p}_{\text{des}}\|\right) \le \delta_{xy}/2, \\ & \text{and } \|\mathbf{R}_z(\beta)\, \mathbf{t} - \mathbf{t}_{\text{des}}\| \le \delta_o\} \end{aligned} \tag{21}$$

where $\mathbf{e}_z$ denotes the standard basis vectors along the z-axis, and δ_o is a small tolerance constraining the orientation difference. By employing this selection strategy for $\mathcal{P}_c$ – cutting out a ring and rotating the CTR configurations – we significantly increase the number of sampled configurations available for initialization.

After obtaining the candidate subset $\mathcal{P}_c$, a metric to quantify the closeness between each configuration's pose in $\mathcal{P}_c$ and the desired pose $\mathbf{T}_{\text{des}}$ is defined as

$$v_c = \|\mathbf{W}_c\,(\mathbf{T} - \mathbf{T}_{\text{des}})\|, \tag{22}$$

where $\mathbf{W}_c = \text{diag}\{w_{c_1}, \cdots, w_{c_K}\}\,(w_{c_i} > 0)$ is a weighting matrix. We then sort all samples in $\mathcal{P}_c$ in ascending order based on v_c.

Using the sorted $\mathcal{P}_c$, we select configuration sample with the smallest v_c as the initial value for iteration. If the iterative process yields an infeasible $\mathbf{q}_{k+1}$, we reinitialize with the next sample from $\mathcal{P}_c$, repeating this procedure until a feasible $\mathbf{q}$ is found. If all samples in $\mathcal{P}_c$ are exhausted without convergence (a rare occurrence), we take the first configuration sample as the IK solution.

Algorithm 1: Inverse Kinematics Computation Algorithm for Concentric-Tube Robots.

Input: $\mathbf{T}_{\text{des}}$, $\mathcal{P}$, $\mathbf{W}_c$, $\mathbf{W}_e$, δ_z, δ_{xy}, δ_o, ϵ, γ, $i = 1$
Output: $\mathbf{q}_{\text{cur}}$

Candidate subset: $\mathcal{P}_c \leftarrow$ (18), (20), and (21);
Closeness metric: $v_c \leftarrow$ (22);
Sorting $\mathcal{P}_c$ in ascending order of v_c;
Number of samples in $\mathcal{P}_c$: N_c;
Initial kinematic input: $\mathbf{q}_{\text{cur}} = \mathbf{q}_{\mathcal{P}_c(1)} \in$ sorted $\mathcal{P}_c$;
Initial pose error: $\mathbf{e} = \mathbf{W}_e\,(\mathbf{T}\,(\mathbf{q}_{\text{cur}}) - \mathbf{T}_{\text{des}})$;
while $\|\mathbf{e}\| > \epsilon$ **do**
  $\mathbf{J}^{\dagger}\,(\mathbf{q}_{\text{cur}}) \leftarrow$ (16);
  $\mathbf{q}_{\text{next}} = \mathbf{q}_{\text{cur}} - \gamma \mathbf{J}^{\dagger}\,(\mathbf{q}_{\text{cur}})\,\mathbf{e}$;
  if $\mathbf{q}_{next}$ *does not satisfy (2)* **then**
    $i = i + 1$;
    if $i > N_c$ **then**
      $\mathbf{q}_{\text{cur}} = \mathbf{q}_{\mathcal{P}_c(1)}$;
      break;
    end
    $\mathbf{q}_{\text{next}} = \mathbf{q}_{\mathcal{P}_c(i)} \in$ sorted $\mathcal{P}_c$;
  end
  $\mathbf{e} = \mathbf{W}_e\,(\mathbf{T}\,(\mathbf{q}_{\text{next}}) - \mathbf{T}_{\text{des}})$;
  $\mathbf{q}_{\text{cur}} = \mathbf{q}_{\text{next}}$;
end
return $\mathbf{q}_{\text{cur}}$

Combining the NR method with this initialization strategy, an IK computation algorithm is proposed for CTR based on the learned FK functions, as shown in Algorithm 1. Additionally, the strategy helps to address the challenge of multiple-solution problem by performing IK computations using all configuration samples in $\mathcal{P}_c$ as initial values. Duplicate solutions are then consolidated, resulting in multiple IK solutions corresponding to the target pose. Due to the inherent limitations of numerical methods, obtaining all possible solutions cannot be guaranteed. However, our initialization strategy generates a substantial number of initial values, whereby facilitating the identification of as many solutions as possible.

TABLE I
DESIGN PARAMETERS OF PRE-CURVED NITINOL TUBES FOR THREE-TUBE CTR

Parameters	Values		
Tube Index	1 (Outer)	2 (Middle)	3 (Inner)
Outer Diameter (mm)	2.70	1.80	1.26
Inner Diameter (mm)	2.00	1.46	1.10
Overall Length (mm)	60	115	160
Pre-curved Length (mm)	50	40	30
Pre-curvature (mm^{-1})	1/130	1/85	1/65
Young's Modulus (GPa)	60	60	60
Poisson's Ratio	0.3	0.3	0.3

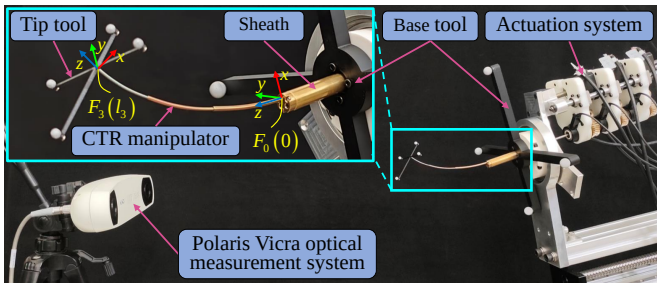


Fig. 5. Experimental platform for data acquisition testing. It mainly comprises a CTR manipulator, an actuation system, an optical measurement system, and optical tools attached to the base of the sheath and the tip of the manipulator.

V. EXPERIMENTAL EVALUATION

In this section, a representative CTR with three pre-curved Nitinol tubes was used as the experimental subject, and the design parameters of the tubes are detailed in Table I.

A. Data Acquisition of Configurations

Based on Table I, the kinematic input was defined as $\mathbf{q} = [\theta_1, l_1, \theta_2, l_2, \theta_3, l_3]^{\mathrm{T}}$. The rotation variables θ_i were sampled according to (5), and the translation variables l_i followed the boundary-dense sampling strategy (6), with $\Delta l_{1_{\max}} = 50$ mm, $\Delta l_{2_{\max}} = 40$ mm, and $\Delta l_{3_{\max}} = 30$ mm. For the PDF of distribution $\mathcal{B}$ for the translation variables, the coefficients c_1, c_2, and c_3 in (7) were set to 1/3, 4, and 8, respectively. By integrating these PDFs, we derived the cumulative distribution functions (CDFs). Using inverse transform sampling on the CDFs, kinematic input samples conforming to distribution $\mathcal{B}$ were generated. We produced 5,000 kinematic input samples each time, and repeated this seven times, for a total of 35,000 samples. Each group was then sorted by formulating a TSP that minimizes total cumulative motor rotations, following the method described in [21].

As shown in Fig. 5, the experimental platform mainly consists of a CTR manipulator, an actuation system, an optical measurement system (Polaris Vicra, Northern Digital Inc., Canada), and optical tools attached to the base and tip. The three Nitinol tubes (Peiertech, China) forming the CTR manipulator were manufactured according to the parameters in Table I. We utilized our previously developed actuation [23] to provide motion to the manipulator. The optical measurement system has a 3D root mean squire error (RMSE) of ≤ 0.25 mm and a maximum update rate of 20 Hz. Optical tools equipped with four markers each were fixed to the base of the sheath and the tip of the manipulator. By aligning the their registered frames with the base frame $F_0(0)$ and tip frame $F_3(l_3)$, we measured the pose of $F_3(l_3)$ relative to $F_0(0)$.

TABLE II
COMPUTATION SPEED OF POSITIONAL FK FUNCTIONS

Num. of Nodes	20	40	80	160	320
Computation Speed (Hz)	232,619	220,410	190,964	158,742	85,709

During the data acquisition, we aligned the pre-curvature planes of the Nitinol tubes and calibrated the CTR manipulator to its initial configuration. Then, the sorted kinematic input samples were sequentially sent to the actuation system. After each movement, we recorded the tip pose $\mathbf{T}$ five times at 100 ms intervals using the optical measurement system. The position $\mathbf{p}$ was calculated as the average of these five readings, and the unit tangent vector $\mathbf{t}$ was obtained by normalizing the averaged measurements. Data acquisition was conducted in seven sessions, each gathering 5,000 samples and taking 3.88 hours (2.80 seconds per sample) on average. By merging and randomly shuffling the data from all sessions, we obtained the dataset $\mathcal{P}$ containing 35,000 CTR configurations samples. The accompanying dataset can be downloaded from https://github.com/hitzhangchao/CTR/tree/main/Kinematics.

B. FK Learning and Function Extraction

To learn the FK functions using dataset $\mathcal{P}$, single-hidden-layer FNNs were implemented using Deep Learning Toolbox in MATLAB. We reserved 3,000 samples from $\mathcal{P}$ for testing and used the remaining 32,000 samples for training, splitting it into training and validation sets in an 85% to 15% ratio. The FK functions mapping the kinematic input $\mathbf{q}$ to the position and orientation were trained separately. The input layer of the network had 6 nodes, and the output layer had 3 nodes, since the dimensions of $\mathbf{q}$ and output ($\mathbf{p}$ or $\mathbf{t}$) are 6 and 3, respectively. Activation function used in the hidden layer was "*tansig*" (the hyperbolic tangent sigmoid transfer function $\tanh$), and that in the output layer was "*purelin*" (the linear transfer function). We employed mean squared error as the loss function and "*trainlm*" as the training function, which updates weights and biases using Levenberg-Marquardt optimization with an adaptive learning rate. By adjusting the amount of training data and the number of hidden layer nodes, we trained the network for 1,000 epochs using "*train*" command. For each setup, training was repeated five times, and we selected the model with the lowest RMSE as the final outcome. All FNNs were trained on a desktop PC with an Intel Core i7-12700 CPU and 32 GB RAM, utilizing 12-core CPU parallel computing to accelerate the training.

After each training, a learned network object was obtained. We then used "*genFunction*" command to extract the function of this object, and reformulated it into the form (14). To evaluate computation speeds of learned and extracted FK functions, we trained five positional FK networks with 20, 40, 80, 160, and 320 hidden layer nodes using 16,000 samples. We then fed 3,000 kinematic inputs from the reserved test

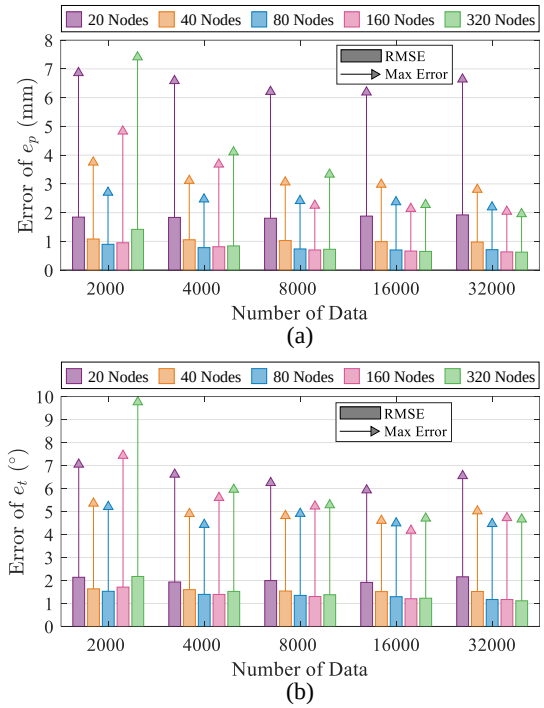


Fig. 6. Accuracy of the learned FK functions trained with varying data sizes and numbers of hidden layer nodes, evaluated on the reserved of 3,000 test samples. (a) and (b) show the root-mean-square error (RMSE) and maximum error for position and orientation, respectively.

set into the functions, recording computation times. Repeating the experiment five times, the average computation frequencies are shown in Table II. Although the computation speed of the FK functions decreases as the number of hidden layer nodes increases, it still remains extremely high.

C. Evaluation of Learned FK

1) Effect of Training Parameters on FK Learning Results: To evaluate the fitting accuracy of the learned FK functions, the position fitting error is defined as

$$e_p = \|\mathbf{p} - \tilde{\mathbf{p}}\| \,, \tag{23}$$

where $\mathbf{p}$ denotes the measured tip position and $\tilde{\mathbf{p}}$ represents the calculated position by the learned FK function. The orientation fitting error is defined as

$$e_t = 2\arcsin\left(\left\|\mathbf{t} - \tilde{\mathbf{t}}\right\|/2\right) \tag{24}$$

that represents the angle between the measured tangent vector $\mathbf{t}$ and its approximated value $\tilde{\mathbf{t}}$.

By varying the training data size (2,000; 4,000; 8,000; 16,000; and 32,000 samples) and adjusting the number of hidden layer nodes (20, 40, 80, 160, and 320), we trained and extracted a series of FK functions for both position and orientation. As shown in Fig. 6, the fitting accuracy of these FK functions was evaluated on the reserved testing set of 3,000 samples. For the position FK, see in Fig. 6 (a), the lowest RMSE was 0.626 mm, and the smallest maximum error was 1.955 mm, both achieved in the FK function trained with 32,000 samples and 320 hidden layer nodes. For orientation FK, see in Fig. 6 (b), the minimum RMSE of 1.117° was obtained with 32,000 samples and 320 nodes, while the smallest maximum error of 4.174° occurred in the FK function with 16,000 samples and 160 nodes.

We further observed that, first, when the hidden layer contains few nodes (20 nodes) or the training dataset is small (2000 samples), the FK functions exhibit relatively large fitting errors in both RMSE and maximum error. Second, for a fixed amount of data, increasing the hidden layer nodes generally reduces fitting errors initially, but the improvement plateaus or even reverses as the network starts overfitting with too many nodes. Third, for the same number of hidden layer nodes, increasing the training data decreases fitting errors, but the rate of improvement diminishes. Beyond 8,000 samples, error reduction becomes marginal, suggesting that additional data contributes few new features to the network.

Therefore, since increasing the data volume yields diminishing marginal improvements in accuracy and extends both data acquisition and training times (more nodes also lengthen training duration), it is essential to balance accuracy requirement, data volume, and the network complexity to achieve effective FK learning results.

2) Comparison between Boundary-Dense and Uniform Sampling Strategies: To evaluate the impact of our proposed boundary-dense sampling strategy on the distribution of CTR configuration samples and FK fitting accuracy compared to the conventional uniform sampling, we fetched out 16,000 samples from the previously obtained dataset $\mathcal{P}$. Then, another dataset was collected using the data acquisition setup in subsection V-A, but with uniformly sampling strategy.

As shown in Fig. 7 (a)-(f), the distribution histograms of joint variables are presented for samples from both sampling strategies. The rotation variables are uniformly sampled within $[-180°, 180°]$. For the translation variables, under the uniform sampling strategy, only the translation samples for tube 1 are uniformly distributed, while tubes 2 and 3 exhibit isosceles triangular-like distributions due to the superposition of multiple uniform distributions. Under the boundary-dense sampling strategy, the translation variable samples of the innermost tube 3 exhibits a half-flare shape similar to the shape of the workspace. Fig. 7 (g) and (h) illustrate the $x-o-z$ cross-sectional views of actual sample distributions in the workspace for both strategies, in which the local density of each sample was color-coded based on the number of its neighboring samples within 5 mm. Additionally, Fig. 7 (i) and (j) depict the number of samples in each slice by slicing the workspace along the x- and z-axis, respectively. Intuitively, the boundary-dense method reduced the excessive clustering of samples in the center of workspace and increases the number of samples at the boundaries.

To quantitatively characterize the spatial distribution of sampling points, the average nearest neighbor distance (ANND) was introduced as

$$\bar{d} = \frac{1}{N}\sum_{i=1}^{N} d_i \,, \tag{25}$$

where N is the number of samples, d_i denotes the distance from i-th sample to its nearest neighboring sample. A larger

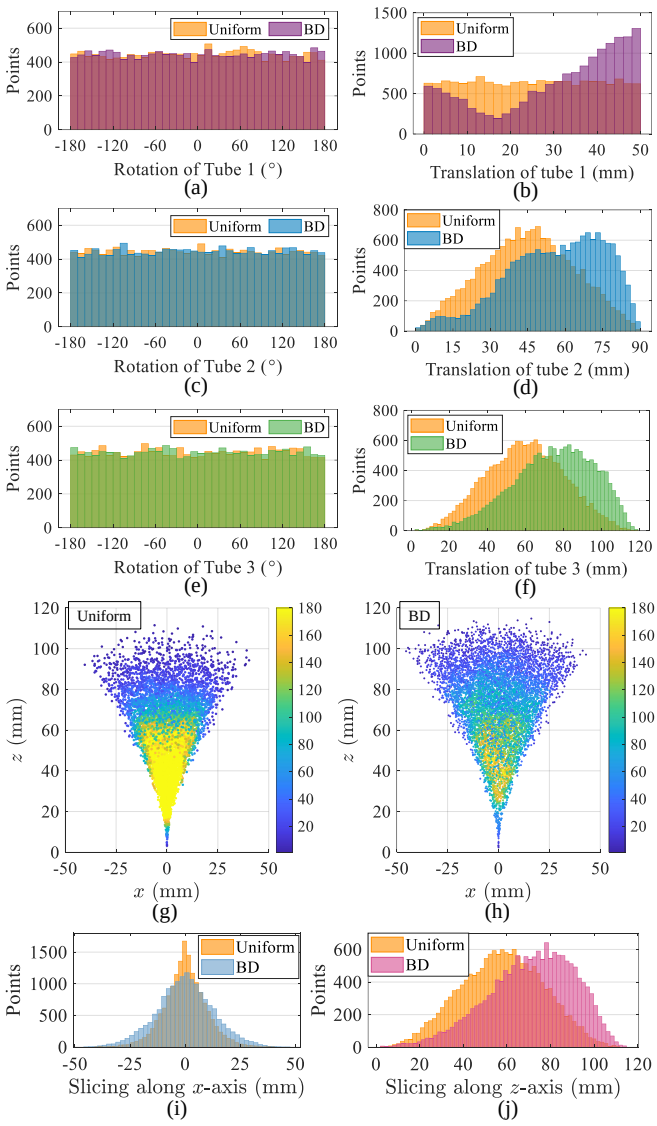


Fig. 7. Distribution of 16,000 configuration samples in in joint space and workspace using boundary-dense (BD) and uniform sampling strategies (a)-(f) are distribution histograms of the rotational and translational joint variables for tube 1, 2, and 3, respectively. (g) and (h) show the $x-o-z$ cross-sectional views of actual sample distributions in the workspace obtained through uniform and BD sampling methods. (i) and (h) depict the number of samples in each slice by slicing the workspace along the x-axis and z-axis, respectively.

ANND implies a more uniform spatial distribution of samples. For the uniformly sampling strategy, the ANND $\bar{d}_{\text{uniform}} = 0.963$ mm. In contrast, the boundary-dense sampling strategy achieved an ANND of $\bar{d}_{\text{BD}} = 1.187$ mm, representing a 23.268% improvement over the uniformly sampling strategy.

As for the FK learning accuracy, we trained FK functions using varying amount of uniformly sampled data (2,000; 4,000; 8,000; 16,000 samples) with 160 hidden layer nodes. Table III presents the fitting error performances of the FK functions from both sampling strategies. For the same data volume, we found that FK functions trained on boundary-dense sampled data generally achieved better accuracy than those trained on uniformly sampled data, which suggests that, given the same data volume, the boundary-dense sampling method introduces more features to the network.

TABLE III
ERROR PERFORMANCE OF FK FUNCTIONS LEARNED FROM DATA USING BOUNDARY-DENSE AND UNIFORM SAMPLING STRATEGIES

Num. of Data	Position Error (mm)				Orientation Error (°)			
	Boundary-Dense		Uniform		Boundary-Dense		Uniform	
	RMSE	Max	RMSE	Max	RMSE	Max	RMSE	Max
2,000	**0.952**	**4.829**	0.989	5.451	**1.711**	**7.434**	1.776	7.531
4,000	**0.733**	**2.218**	0.839	2.434	**1.393**	5.602	1.463	4.833
8,000	0.702	**2.249**	0.661	2.420	1.301	**5.227**	1.300	5.551
16,000	**0.665**	**2.138**	0.711	2.340	**1.206**	**4.174**	1.218	4.185

3) Comparison with Other FK Methods: To fairly compare our method with other typical CTR FK methods, we used the same CTR setup presented in Table I. We evaluated the performance – in terms of accuracy, computational speed, and differentiability – of the torsionally rigid model [4], the torsionally compliant model [5], the truncated Fourier series method [5], Grassmann *et al.*'s learning-based method [21], and our FK learning and extraction method.

For the torsionally rigid [4] and compliant models [5], we implemented both FK models in MATLAB based on the design parameters in Table I. Joint variables from the reserved test set of 3,000 samples were input into each model, and we recorded their output poses, and measured the total computation time. Computation speed was calculated by dividing 3,000 by the total computation time. The errors between the computed and actual poses were then calculated, with the results summarized in Table IV.

Regarding the Fourier series method [5], [7], we generated 75,000 configuration data using the torsionally compliant model with the uniform sampling strategy. A reduced representation of the joint variables was used by expressing the rotation angles of the inner tubes relative to the outermost tube, as not applying this dimensionality reduction would significantly decrease computational efficiency for the Fourier series method. We then implemented the multidimensional truncated Fourier series method in MATLAB to fit the generated data and derive the fitted FK functions. Using the same testing method as before, we evaluated the error performance and computation speed, the results are presented in Table IV.

For Grassmann *et al.*'s learning-based method [21], their network architecture was replicated in MATLAB using the Deep Learning Toolbox, strictly adhering to the same network parameters – including data normalization, data division ratios, the ADAM optimizer, ReLU activation function, an initial learning rate of 0.0005, and a batch size of 128. We trained the FK model using 16,000 uniformly sampled data and employed 160 hidden layer nodes, and used trigonometric representation for the kinematic input. The accuracy and computation speed of the trained FK model were then evaluated, with the results shown in Table IV.

For our proposed method, we employed the FK functions learned and extracted from 16,000 boundary-dense sampled data with 160 hidden layer nodes. The error performance and computation speed of our method are also listed in Table IV.

From the table, several observations can be made: 1) In terms of error performance, our method achieves position and

TABLE IV
PERFORMANCE COMPARISON OF SEVERAL TYPICAL FK METHODS AND OUR PROPOSED METHOD FOR THE SAME THREE-TUBE CTR

FK Method	Position RMSE	Position RMSE in %	Position Max Error	Orientation RMSE	Orientation Max Error	Num. of Parameters	Computation Speed	Differentiable
Torsionally Rigid Model [4]	3.122 mm	2.602%	11.381 mm	7.542°	19.886°	–	2,392 Hz	No
Torsionally Compliant Model [5]	1.996 mm	1.664%	9.427 mm	5.397°	15.271°	–	25 Hz	No
Fourier Series Method [5], [7]	2.716 mm	2.264%	15.762 mm	6.368°	17.554°	9,375	37,232 Hz	Yes
Learning-based Method [21]	0.626 mm	0.522%	6.172 mm	0.722°	7.207°	2,083	1,068 Hz	No
Our Method	**0.665 mm**	**0.554%**	**2.138 mm**	**1.206°**	**4.174°**	**1,603**	**158,742 Hz**	**Yes**

orientation RMSE comparable to those of Grassmann *et al.*'s method and exhibits best maximum error performance than the other methods. This indicates that our method provides high accuracy in predicting the FK of the CTR. 2) Regarding computation speed, our method attains the fastest FK computation speed among al the compared methods. This significant improvement primarily due to the extraction of explicit functions from the trained network, which simplifies the computation and reduces processing time. 3) With respect to differentiability, by employing the continuous hyperbolic tangent activation function in the hidden layer, our method produces differentiable FK functions. This feature facilitates subsequent mathematical derivations and operations, such as Jacobian computation, enhancing the method's applicability in control and optimization algorithms.

D. IK Solving based on the Learned FK Functions

1) IK Solutions for Given CTR Tip Poses: To evaluate the effectiveness of the proposed CTR IK computation method based on the learned FK functions, we randomly selected 500 poses from the reserved test set as target poses. The IK Algorithm 1 was implemented in MATLAB. The pose weighting matrix was set to $\mathbf{W}_e = \text{diag}\{1, 1, 1, 0.0115, 0.0115, 0.0115\}$, assigning equal weight to a position deviation of 1 mm and an angular deviation of 5°. We defined the pose error threshold as $\epsilon = 0.0005$ and the step size as $\gamma = 0.2$. The FK functions used in this evaluation were trained on 16,000 samples with 160 hidden layer nodes, and the Jacobian matrix was obtained by directly differentiating the FK functions according to (16). For the initialization strategy, 20,000 configuration samples were selected from the dataset $\mathcal{P}$, setting the position tolerances to $\delta_z = \delta_{xy} = 6$ mm and the orientation tolerance to $\delta_o = 0.2091°$ (corresponding to $\gamma = 12°$ in Fig. 4 (c)) in (21), and the weighting matrix $\mathbf{W}_c = \mathbf{W}_e$.

The IK computations for the 500 target poses achieved an average total computation time of 2.467s (approximately 203 Hz). Configuration reinitialization occurred during the IK iterative process for 254 out of the 500 target poses, demonstrating that the proposed initialization strategy effectively ensured the success of the IK computations.

The computed joint variables were sent to the prototype shown in Fig. 5 for execution, while the actual tip poses were measured. The errors between the 500 actual poses and their corresponding target poses are summarized in Table V. The results demonstrate that the average position error was 0.850 ± 0.455 mm and the average orientation error was under $1.970 \pm 1.087°$, indicating high accuracy in IK computation.

TABLE V
ERRORS BETWEEN 500 MEASURED TIP POSES AND CORRESPONDING TARGET POSES IN CTR IK COMPUTATION

Position Mean Error	Position Max Error	Orientation Mean Error	Orientation Max Error	Computation Speed
0.850 ± 0.455 mm	2.814 mm	$1.970 \pm 1.087°$	7.569°	203 Hz

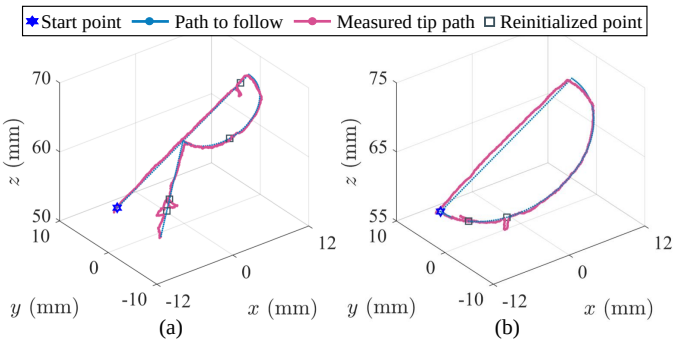
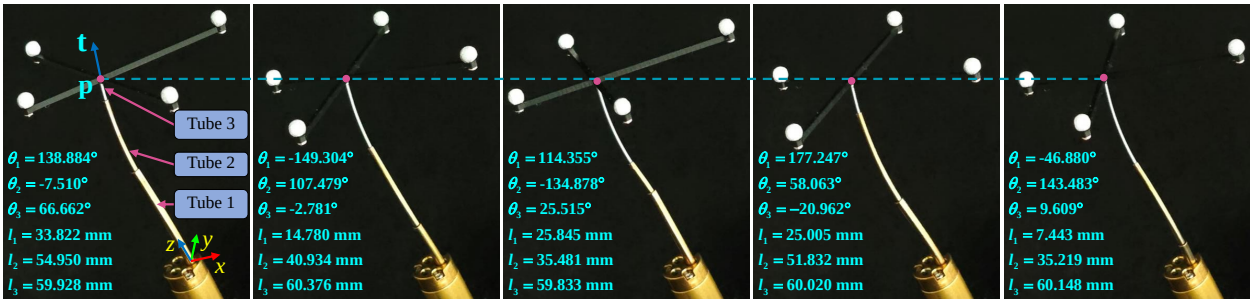


Fig. 8. Path following performance of the CTR along two designed paths. (a) Path "R", consisting of 193 planned waypoints. (b) Path "D", consisting of 206 planned waypoints.

2) Path Following: To assess the effectiveness of the proposed IK method in tip path-following tasks, experiments were conducted to evaluate positioning accuracy, computational speed, and the impact of the initialization strategy. As shown in Fig. 8, two paths – designated as "R" and "D" – were designed for the CTR to follow. Both paths represent letters that can be drawn in a single stroke, incorporating both straight and curved segments. Each path consists of numerous discrete waypoints, spaced 0.3 mm in Euclidean distance, with path "R" and "D" containing 193 and 206 waypoints, respectively.

To track the planned paths, we first computed the IK for the start point to obtain the initial CTR configuration. Since the paths contain only position information, the error weighting matrix was set to $\mathbf{W}_e = \text{diag}\{1, 1, 1\}$. We set $\epsilon = 0.0001$ and $\gamma = 0.1$. For reinitialization, 20,000 sampled configurations were used, and $\delta_z = \delta_{xy} = 5$ mm. The position FK function trained and extracted using 16,000 boundary-dense samples and 160 hidden layer nodes was applied. Starting from the start point, we set the position of the next waypoint as the new target and used the resolved joint variables from the current waypoint as the initial guess for iteration. This process was repeated until joint variables for all waypoints were obtained. When an infeasible CTR configuration was detected during the IK iteration, the initialization strategy was activated. In such cases, the sorting metric (22) was



(a) (b) (c) (d) (e)

Fig. 9. Five joint variable solutions and their corresponding actual shapes and tip poses of the 10th target pose tabulated in Table VII. Despite significant variations in joint variables, the tip poses remain nearly identical.

TABLE VI
POSITION ERRORS BETWEEN THE MEASURED PATHS AND THE PLANNED PATHS AT CORRESPONDING WAYPOINTS

Path	Way-points	Reinit-points	Compuing Time	Trail	Position Mean Error	Position Max Error
R	193	4	0.244s	1	0.625 ± 0.293 mm	1.121 mm
				2	0.628 ± 0.287 mm	1.144 mm
				3	0.648 ± 0.285 mm	1.168 mm
D	206	2	0.223s	1	0.747 ± 0.299 mm	1.358 mm
				2	0.774 ± 0.292 mm	1.349 mm
				3	0.810 ± 0.298 mm	1.381 mm

TABLE VII
ERROR PERFORMANCE BETWEEN TEN TARGET POSES AND THE CORRESPONDING ACTUAL POSES OF MULTIPLE SOLUTIONS

Desired Pose No.	Num. of Solutions	Position Mean Error	Position Max Error	Orientation Mean Error	Orientation Max Error
1	3	0.547 mm	0.791 mm	1.565°	2.623°
2	3	0.672 mm	0.953 mm	1.426°	2.292°
3	5	1.098 mm	1.701 mm	1.656°	2.456°
4	3	0.838 mm	0.950 mm	1.694°	2.122°
5	3	0.577 mm	0.755 mm	1.482°	1.921°
6	3	0.579 mm	0.624 mm	1.188°	1.749°
7	3	0.307 mm	0.406 mm	1.076°	1.283°
8	1	1.348 mm	1.348 mm	1.643°	1.643°
9	3	0.841 mm	1.028 mm	0.982°	1.055°
10	5	1.074 mm	1.195 mm	1.566°	2.942°

replaced by $v_c = \sum_{i=1}^{N} (\Delta\theta_i/\pi + \Delta l_i/l_{i_{\max}})$, where $\Delta\theta_i$ and Δl_i represent the changes in rotational and translational joint variables relative to the previous waypoints, respectively. We implemented the IK computations for both paths in MATLAB, repeating the calculations five times. For path "R", the average total computing time was 0.244s, with reinitialization required at 4 waypoints, taking 0.035s. For path "D", the average computing time was 0.223s, with reinitialization occurring at 2 waypoints, taking 0.021s.

The computed joint variable sequences for each path were then sent to the CTR prototype. The optical tracking system continuously recorded the CTR tip positions, specifically marking the positions upon reaching each waypoint. The measured tip paths are shown in Fig. 8. Each experiment was conducted three times, and the position errors between the measured and the planned paths at corresponding waypoints are summarized in Table VI. The results demonstrate average tracking errors of less than 0.65 mm for path "R" and 0.85 mm for path "D", with maximum errors not exceeding 1.4 mm. At waypoints where reinitialization occurred, although some abrupt changes were observed in the measured paths, the proposed strategy primarily ensured successful IK computation, preventing potentially larger abrupt changes.

3) Multiple Solutions of IK Solving: To evaluate the ability of our IK method to handle multiple solutions and further verify whether the learned FK functions have effectively captured multi-solution mappings, we randomly selected 10 target poses from the reserved test set. For each target pose, the method described in (21) and (22) was used to obtain the candidate subset $\mathcal{P}_c$. Then, points from $\mathcal{P}_c$ were sequentially selected as initial values for the IK iteration, solving for joint variables corresponding to different initializations. The ϵ was set to 0.0001, and other parameters were the same as those in subsection V-D1. Since many initial values converged to identical joint variable solutions, duplicates were removed, yielding multiple distinct solutions for each target pose.

Each set of joint variables was then sent to the CTR prototype and the actual tip poses corresponding to each solution were measured. The errors between these 10 target poses and the corresponding actual poses of multiple solutions are summarized in Table VII. The results indicate that the occurrence of multiple solutions in CTR IK computation is common with this design, and the corresponding tip poses consistently exhibit small error margins. Furthermore, Fig. 9 illustrates the CTR configurations for five solutions of the 10th target pose in Table VII. As can be seen, despite significant variations in joint variables across solutions, the tip poses remain nearly identical. This demonstrated the capability of our IK method in addressing the multi-solution problem and confirmed that the learned FK functions have effectively captured the multi-solution mappings from the joint space to Cartesian space.

VI. CONCLUSIONS

In this paper, we have proposed a precise, explicit, and model-free kinematic method for CTRs, including a refined learned-based FK approach and an IK solver based on the learned FK functions. Our boundary-dense sampling strategy obtained a real-world CTR configuration dataset with more evenly distributed samples in the workspace compared to

traditional methods. Using single-hidden-layer FNNs with tangent hyperbolic activation function, we learned and extracted explicit FK functions. Leveraging the learned FK, we applied the Newton-Raphson method to solve the IK, incorporating a initialization strategy to provide reasonable initial guesses and handle infeasible configurations, thereby improving the IK success rate. Comprehensive experiments demonstrate that our FK method provided a concise and differentiable expression with accuracy comparable to the state-of-art learning-based FK methods (average tip position error of approximately 0.5% of the robot's length), and achieving highest FK computation speed of up to 158,742 Hz. We demonstrated precise IK computation and successfully addressed tasks such as path following and multi-solution computation. This work advances the application of learning-based kinematic methods in CTR, and future efforts will focus on extending this approach to CTR motion planning and feedback control.

REFERENCES

[1] P. E. Dupont, N. Simaan, H. Choset, and C. Rucker, "Continuum robots for medical interventions," *Proceedings of the IEEE*, vol. 110, no. 7, pp. 847–870, 2022.

[2] M. F. Rox, D. S. Ropella, R. J. Hendrick, E. Blum, R. P. Naftel, H. C. Bow, S. D. Herrell, K. D. Weaver, L. B. Chambless, and R. J. Webster III, "Mechatronic design of a two-arm concentric tube robot system for rigid neuroendoscopy," *IEEE/ASME Transactions on Mechatronics*, vol. 25, no. 3, pp. 1432–1443, 2020.

[3] Z. Mitros, S. H. Sadati, R. Henry, L. Da Cruz, and C. Bergeles, "From theoretical work to clinical translation: Progress in concentric tube robots," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 5, no. 1, pp. 335–359, 2022.

[4] P. Sears and P. Dupont, "A steerable needle technology using curved concentric tubes," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006, pp. 2850–2856.

[5] P. E. Dupont, J. Lock, B. Itkowitz, and E. Butler, "Design and control of concentric-tube robots," *IEEE Transactions on Robotics*, vol. 26, no. 2, pp. 209–225, 2010.

[6] D. C. Rucker, B. A. Jones, and R. J. Webster III, "A geometrically exact model for externally loaded concentric-tube continuum robots," *IEEE Transactions on Robotics*, vol. 26, no. 5, pp. 769–780, 2010.

[7] C. Girerd and T. K. Morimoto, "Design and control of a hand-held concentric tube robot for minimally invasive surgery," *IEEE Transactions on Robotics*, vol. 37, no. 4, pp. 1022–1038, 2021.

[8] D. C. Rucker and R. J. Webster, "Computing jacobians and compliance matrices for externally loaded continuum robots," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 945–950.

[9] R. Xu, A. Asadian, A. S. Naidu, and R. V. Patel, "Position control of concentric-tube continuum robots using a modified jacobian-based approach," in *2013 IEEE International Conference on Robotics and Automation*, 2013, pp. 5813–5818.

[10] H. Azimian, T. Looi, and J. Drake, "Closed-loop inverse kinematics under inequality constraints: Application to concentric-tube manipulators," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 498–503.

[11] J. Burgner, D. C. Rucker, H. B. Gilbert, P. J. Swaney, P. T. Russell, K. D. Weaver, and R. J. Webster, "A telerobotic system for transnasal surgery," *IEEE/ASME Transactions on Mechatronics*, vol. 19, no. 3, pp. 996–1006, 2014.

[12] D. Gottlieb and C.-W. Shu, "On the gibbs phenomenon and its resolution," *SIAM review*, vol. 39, no. 4, pp. 644–668, 1997.

[13] C. Bergeles, F. Lin, and G. Yang, "Concentric tube robot kinematics using neural networks," in *Hamlyn symposium on medical robotics*, vol. 6, 2015, pp. 1–2.

[14] 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 *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 5125–5132.

[15] R. Grassmann and J. Burgner-Kahrs, "On the merits of joint space and orientation representations in learning the forward kinematics in se (3)." in *Robotics: science and systems*, 2019.

[16] A. Kuntz, A. Sethi, R. J. Webster, and R. Alterovitz, "Learning the complete shape of concentric tube robots," *IEEE Transactions on Medical Robotics and Bionics*, vol. 2, no. 2, pp. 140–147, 2020.

[17] N. Liang, R. M. Grassmann, S. Lilge, and J. Burgner-Kahrs, "Learning-based inverse kinematics from shape as input for concentric tube continuum robots," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 1387–1393.

[18] K. Iyengar and D. Stoyanov, "Deep reinforcement learning for concentric tube robot control with a goal-based curriculum," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 1459–1465.

[19] K. Iyengar, S. Sadati, C. Bergeles, S. Spurgeon, and D. Stoyanov, "Sim2real transfer of reinforcement learning for concentric tube robots," *IEEE Robotics and Automation Letters*, vol. 8, no. 10, pp. 6147–6154, 2023.

[20] K. Iyengar, S. Spurgeon, and D. Stoyanov, "Deep reinforcement learning for concentric tube robot path following," *IEEE Transactions on Medical Robotics and Bionics*, vol. 6, no. 1, pp. 18–29, 2024.

[21] R. M. Grassmann, R. Z. Chen, N. Liang, and J. Burgner-Kahrs, "A dataset and benchmark for learning the kinematics of concentric tube continuum robots," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 9550–9557.

[22] A. Pinkus, "Approximation theory of the mlp model in neural networks," *Acta numerica*, vol. 8, pp. 143–195, 1999.

[23] C. Zhang, G. Cen, X. Yang, X. Xu, M. Q.-H. Meng, S. Song, and J. Wang, "Design optimization of pyramid-shaped transmission system for multiarm concentric-tube robots," *IEEE/ASME Transactions on Mechatronics*, vol. 29, no. 5, pp. 3427–3438, 2024.

Chao Zhang (Student Member, IEEE) received the B.E. degree in Mechanical Design, Manufacturing and Automation from the Harbin Institute of Technology, Harbin, China, in 2016, and the M.E. degree in Mechanical Engineering in 2020 from the Harbin Institute of Technology (Shenzhen), Shenzhen, China, where he is currently working toward the Ph.D. degree in Mechanical Engineering with the School of Robotics and Advanced Manufacturing.

His research interests include design and control of medical and surgical robots. Mr. Zhang was the recipient of the finalist of best conference paper of IEEE ROBIO 2021.

Shuang Song (Member, IEEE) received B.S. degree in Computer Science and Technology from North Power Electric University, Beijing, China, in 2007, the M.S. degrees in Computer Architecture from the Chinese Academy of Sciences, Beijing, China, in 2010, and the Ph.D. degree in Computer Application Technology from the University of Chinese Academy of Sciences, Beijing, China, in 2013.

He is currently a Professor with the School of Robotics and Advanced Manufacturing, Harbin Institute of Technology (Shenzhen), Shenzhen, China. His main research interests include magnetic tracking and actuation for Bioengineering applications.

Jiaole Wang (Member, IEEE) received the B.E. degree in Mechanical Engineering from Beijing Information Science and Technology University, Beijing, China, in 2007, the M.E. degree from the Department of Human and Artificial Intelligent Systems, University of Fukui, Fukui, Japan, in 2010, and the Ph.D. degree from the Department of Electronic Engineering, The Chinese University of Hong Kong (CUHK), Hong Kong, in 2016.

He was a Research Fellow with the Pediatric Cardiac Bioengineering Laboratory, Department of Cardiovascular Surgery, Boston Children's Hospital and Harvard Medical School, Boston, MA, USA. He is currently an Associate Professor with the School of Biomedical Engineering and Digital Health, Harbin Institute of Technology (Shenzhen), Shenzhen, China. His main research interests include medical and surgical robotics, image-guided surgery, human-robot interaction, and magnetic tracking and actuation for biomedical applications.