Learning-based Inverse Kinematics from Shape as Input for Concentric Tube Continuum Robots

Nan Liang, Reinhard M. Grassmann, Sven Lilge and Jessica Burgner-Kahrs, Senior Member, *IEEE*

***Abstract*— We introduce a methodology to compute the inverse kinematics for concentric tube continuum robots from a desired shape as input. We demonstrate that it is possible to accurately learn joint parameters using neural networks for a discrete point-wise shape representation with different discretization. In comparison to a vanilla numerical method, the learning-based method is preferred in terms of accuracy in joint space and computation. Representing the shape with up to 20 equidistant points, a shape-to-joint inverse kinematics with errors of 2.22° and $1.45\,\mathrm{mm}$ is obtained. Further, we extend the shape-to-joint inverse kinematics to image-to-joint inverse kinematics utilizing multi-view images as shape representation. This image-based method achieves errors of 6.02° and $2.76\,\mathrm{mm}$. Both approaches, i.e., shape-to-joint and image-to-joint, result in higher accuracy compared to the learning-based state-of-the-art approach which only considers the tip pose.**

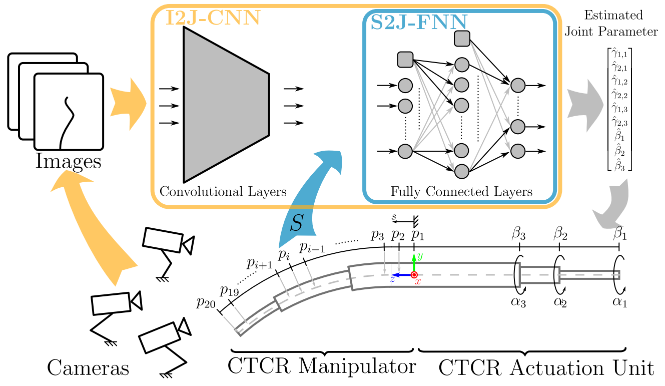


Fig. 1: Principle of our learning-based approaches.

I. Introduction

Concentric tube continuum robots (CTCR) are composed of nested, pre-curved, and super-elastic tubes. The inherent compliance and flexibility alongside a variety of achievable motion profiles makes them well suited to operate in confined and tortuous environments. Thanks to their small scale, with typical diameters $<2.5\,\mathrm{mm}$ and overall length of 200-300 mm, CTCR show great potential for medical applications, such as minimally invasive surgery [1].

A major challenge is modeling the forward kinematics (FK) and inverse kinematics (IK). The highly non-linear kinematics and numerous physical phenomena involved during their motion, such as bending, torsion, shear, extension, and friction, limit the achievable forward kinematics' accuracy as well as computation time. The CTCR research community has converged to a physics-based kinetostatic model for FK. By leveraging the Cosserat theory of elastic rods, the shape of a CTCR can be determined from joint space parameters by solving differential equations numerically [2], [3]. Additional phenomena such as clearance and friction [4], [5], material hysteresis [6], and snapping behaviour [7] have been studied, but integration into a real-time FK for general CTCR has not been achieved. To account for unmodeled effects, online adaptive methods to continuously update model parameters [8] or calibration techniques [9] have been proposed.

IK has been addressed in terms of mapping from a desired tip position or pose of a CTCR to joint parameters. Closed-form, analytical solutions only exist for simple CTCR composed of a stiffness-dominating tube pair [10] or for simplified constant-curvature assumptions [11]. As no analytical solutions exist for general CTCR, the IK relies on numerical approximations [12] or differential inverse kinematics [2], [13], [14], [15].

To overcome limitations of physics-based models, learning-based approaches have been subject to recent research [16], [17], [18], [19]. The idea of using neural network to learn the FK leads to results outperforming the accuracy of physics-based FK [17], [18]. For the IK, a feedforward neural network (FNN) architecture has been proposed which takes the desired pose as input [16], [17]. With data obtained from a CTCR simulator, errors below $0.8\,\mathrm{mm}$ and 0.1° in tube translation and rotation could be achieved [16]. Using data obtained with a physical robot prototype, an FNN can achieve an error of $4.0\,\mathrm{mm}$ and 8.3° for translation and rotation, respectively [17]. Kuntz et al. [20] address learning-based FK with CTCR shape information. They propose a FNN to predict the coefficients of polynomial basis functions as a shape representation from joint parameters as input. To the best of our knowledge, the IK of a CTCR has not been solved from desired shape as input thus far.

In this paper, we introduce a methodology to approximate the IK for CTCR from shape using a learning-based approach. We demonstrate that it is possible to accurately learn CTCR joint parameters using neural networks from shape information. In particular, we represent a desired CTCR shape by discrete curves and by multi-view images.

We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC), [RGPIN-2019-04846].

All authors are with the Continuum Robotics Laboratory, Department of Mathematical & Computational Sciences, University of Toronto Mississauga, Canada. nan.liang@mail.utoronto.ca

TABLE I: CTCR tube parameters

Parameter			Set of tubes		
Term	Symbol	Unit	Tube 1 (inner)	Tube 2 (middle)	Tube 3 (outer)
Length, overall	L	mm	220	165	110
Length, straight	L_{s}	mm	184	107	10
Curvature	κ_x	m^{-1}	28	12.4	4.37
Diameter, outer	D_{o}	mm	0.5	0.9	1.5
Diameter, inner	D_{i}	mm	0.4	0.7	1.2
Young's Modulus	E	GPa	50	50	50
Poisson's ratio	ν	1	0.3	0.3	0.3

II. Concentric Tube Continuum Robots

In this paper, we consider typical CTCR with tubes of total length L_i with a straight segment of length L_s and a curved segment of length L_c with constant precurvature κ_x. In particular, we look at CTCR composed of $n = 3$ tubes, with index $i = 1$ referring to the innermost and $i = 3$ to the outermost tube. As illustrated in the Fig. 1, each tube is actuated with two degrees of freedom (axial translation β_i and rotation α_i) at its base. The joint values are subject to

$$\alpha_i \in [-\pi, \pi) \tag{1}$$

$$\beta_i \in [-L_i, 0) \tag{2}$$

$$\beta_1 \leq \beta_2 \leq \beta_3 \leq 0 \tag{3}$$

$$0 \leq L_3 + \beta_3 \leq L_2 + \beta_2 \leq L_1 + \beta_1 \tag{4}$$

For a given set of joint values, the CTCR's deployed length is determined by $L_1 + \beta_1$ with its maximum at $\beta_1 = 0$, i.e., the inner tube is fully extended. All tubes are free to rotate with respect to each other; however, for translational motion, the inequality constraints (3) and (4) are required to ensure that each tube is always aligned with or extending from its surrounding tube.

Using $\gamma_{1,i} = \cos(\alpha_i)$ and $\gamma_{2,i} = \sin(\alpha_i)$, the joint space is represented as

$$q = [\gamma_{1,1}, \gamma_{2,1}, \gamma_{1,2}, \gamma_{2,2}, \gamma_{1,3}, \gamma_{2,3}, \beta_1, \beta_2, \beta_3]. \tag{5}$$

This representation has been proven to be advantageous for learning-based approaches in [17], [18]. In the following, we use a tube set as specified in Table I.

III. Learning-based Shape-to-joint IK

Here, we investigate the problem of how to determine the IK of a CTCR for a desired robot shape, referred to as shape-to-joint (S2J) inverse kinematics or S2J-IK in short.

A. Problem Definition

We consider a discrete representation of the desired shape. A shape S is described by m equidistant points $p_j = [x_j, y_j, z_j]^\top$ w.r.t. the total robot length. The points are ordered from the base to the tip and defined in a fixed coordinate frame at the robot's base as illustrated in Fig. 1. As the first point $p_1 = [0, 0, 0]^\top$ is constant, the remaining $m - 1$ points from p_2 to p_m are relevant for the S2J-IK:

$$S = [x_2, y_2, z_2, \ldots, x_m, y_m, z_m] \in \mathbb{R}^{3(m-1)} \tag{6}$$

The desired output of our learning-based S2J-IK for any given shape S are the corresponding joint values $q \in \mathbb{R}^9$.

B. Shape-to-Joint Feedforward Neural Network

We propose a shape-to-joint feedforward neural network (S2J-FNN) consisting of fully connected layers with input dimension of $3(m-1)$ and output dimension 9. For selecting a network structure, we conduct a grid search of varying numbers of hidden layers and neurons at each layer. The grid search uses $m = 20$, 200 training epochs, and rectified linear activation function (ReLU) activation functions in the hidden layers. The selected structure in Table II is one hidden layer deeper than the FNN used in [16], [17].

TABLE II: Neural Network Structure of S2J-FNN.

Layer Name	Number of Neurons	Activation Function
Input Layer	$3(m-1)$	None
FC1	1000	ReLU
FC2	500	ReLU
FC3	250	ReLU
Output Layer	9	Linear

In order to learn α_i and β_i simultaneously, we propose a customized loss function that consists of two components. The first component (7) is based on the cosine distance between the estimation $\hat{\alpha}_i$ and real rotational angle α_i, which has a minimal value of zero when the two angles are aligned.

$$D_i = 1 - \frac{\gamma_{1,i}\hat{\gamma}_{1,i} + \gamma_{2,i}\hat{\gamma}_{2,i}}{\sqrt{\gamma_{1,i}^2 + \gamma_{2,i}^2}\sqrt{\hat{\gamma}_{1,i}^2 + \hat{\gamma}_{2,i}^2}} \in [0, 2] \tag{7}$$

Using the cosine distance has the convenience of easily addressing the predicted values of $\hat{\gamma}_{1,i}$ and $\hat{\gamma}_{2,i}$ that exceed the range $[-1, 1]$, considering this constraint is not enforced during the model training. The second component is using the common squared error for the translational component multiplied by a scalar weight constant ω.

The two components are assembled to the loss function

$$\mathcal{L} = \sum_{i=1}^{3} \left(D_i + \omega \left(\beta_i - \hat{\beta}_i \right)^2 \right). \tag{8}$$

We determine ω using another grid search performed on an FNN structure with two hidden layers (800 and 400 neurons), trained for 200 epochs for $m = 20$. The results are shown in Table III. Considering that β_i are expressed in meters in (8) such that their contribution is smaller than the rotational part, magnifying it by a large factor is necessary. We choose $\omega = 200$ as the expected value of position and orientation errors are approximately in the same scale. The grid search results also show that (8) outperforms the commonly used Root Mean Squares Loss (RMS) for the specific task we investigate here.

C. Data Generation

The data set to train S2J-FNN is denoted $\mathbb{Q}_1$ in the following with size of 100 000 samples of q with $\alpha_i \in [-\pi, \pi)$, $\beta_1 \in [-154, 0]$, $\beta_2 \in [-115.5, 0]$, and $\beta_3 \in [-77.5, 0]$,

TABLE III: Grid search for ω. Errors e_{α_i} defined in (10) are in degrees and e_{β_i} defined in (11) are in millimeters.

ω		e_{α_1}	e_{α_2}	e_{α_3}	e_{β_1}	e_{β_2}	e_{β_3}
1	mean	4.90	3.72	**1.62**	1.21	5.32	3.39
	median	3.81	2.22	**0.90**	0.94	4.61	2.59
10	mean	3.99	3.87	1.71	0.77	4.26	2.80
	median	3.01	2.37	1.00	0.59	3.56	2.14
50	mean	3.78	3.86	1.75	0.58	3.11	2.24
	median	2.91	2.34	1.09	0.44	2.51	1.65
100	mean	3.72	3.67	1.86	0.47	2.88	2.13
	median	**2.82**	2.25	1.13	0.34	2.27	1.60
200	mean	**3.70**	**3.46**	1.64	**0.42**	**2.41**	**1.86**
	median	2.86	**2.11**	0.94	**0.31**	**1.90**	**1.35**
400	mean	4.21	3.89	1.73	**0.42**	2.53	1.98
	median	3.17	2.35	0.99	**0.31**	1.96	1.46
RMS	mean	4.91	3.92	1.74	1.19	4.60	2.76
	median	3.52	2.47	0.94	0.89	3.80	2.20

while satisfying the joint space constraints in (3) and (4). We further enforce a minimum distance between the outer and middle tube $(\beta_1 + L_1) - (\beta_2 + L_2) > 25\,\mathrm{mm}$ to ensure that all shapes in $\mathbb{Q}_1$ are sufficiently distinguishable.

For each sample q_k in $\mathbb{Q}_1$ we calculate the corresponding shape S_k using the physic-based FK proposed by Rucker et al. [3]. The set of all shapes is denoted as $\mathbb{S}_1$ hereafter.

D. S2J-FNN Training

The S2J-FNN is trained on data set $\{\mathbb{S}_1, \mathbb{Q}_1\}$ with a training set of size 80 000 and validation set of size 10 000. The remaining 10 000 samples are used for testing. The model is implemented in PyTorch [21] and, after a tuning process, trained by Adam Optimizer [22] with a learning rate $\lambda = 10^{-4}$, decay rates 0.9 and 0.999, and mini batch size of 64, for 1000 epochs. The Xavier initialization [23] is used to initialize S2J-FNN weights. The models are trained with different values of m ranging from 2 to 20. After the training phase, the S2J-FNN performance is evaluated with the testing set.

E. Evaluation

The estimated rotational angles $\hat{\alpha}_i$ and α_i are determined using atan2 leading to

$$\hat{\alpha}_i = \operatorname{atan2}(\hat{\gamma}_{2,i}, \hat{\gamma}_{1,i}) \text{ and } \alpha_i = \operatorname{atan2}(\gamma_{2,i}, \gamma_{1,i}). \quad (9)$$

For each tube, the estimation errors are evaluated in degrees for rotation and in millimeters for translation. We also adapt the definitions from [17] for total errors, defined as following.

$$e_{\alpha_i} = |\alpha_i - \hat{\alpha}_i| \quad (10)$$

$$e_{\beta_i} = \left|\beta_i - \hat{\beta}_i\right| \quad (11)$$

$$e_\alpha = \sqrt{e_{\alpha_1}^2 + e_{\alpha_2}^2 + e_{\alpha_3}^2} \quad (12)$$

$$e_\beta = \sqrt{e_{\beta_1}^2 + e_{\beta_2}^2 + e_{\beta_3}^2} \quad (13)$$

We further explore the relationship between joint parameter estimation error and shape. Since the joint space constraints in (1)-(4) are not enforced during the training process, some outputs $\hat{q}_k$ are not valid. For the outputs $\hat{q}_k$ from S2J-FNN that satisfy (1)-(4), we determine the shape $\hat{S}_k$ for the estimated joint parameters $\hat{q}_k$ using the physics-based FK model [3] with $m = 20$. This reconstructed shape $\hat{p}_j = [\hat{x}_j, \hat{y}_j, \hat{z}_j]^\top$ of $\hat{S}_k$ is then compared to the ground truth values $p_j = [x_j, y_j, z_j]^\top$ of S_k. Both, absolute error in millimeter and relative error in percentage of the robot length s_j are calculated by

$$e_j^{\text{abs}} = \sqrt{(x_j - \hat{x}_j)^2 + (y_j - \hat{y}_j)^2 + (z_j - \hat{z}_j)^2} \quad (14)$$

$$e_j^{\text{rel}} = e_j^{\text{abs}} / s_j \quad (15)$$

The average value of e_j^{abs} and e_j^{rel} averaged from $j = 2$ to $j = 20$ are reported, along with e_{20}^{abs} and e_{20}^{rel}, which are the errors specifically at the CTCR tip.

F. Results

The performance is evaluated for values of m from 2 to 20 summarized in Fig. 2. As we can see from Fig. 2a and Fig. 2b, the errors in joint space e_{α_i} and e_{β_i} are significant for $m < 5$, and remains constant with more input points. A similar trend can be observed for the average absolute and relative reconstruction errors in Fig. 2c and Fig. 2d. With $m = 20$, the median e_{α_i} are 1.13°, 1.01°, 0.62°, and median e_{β_i} are $0.39\,\mathrm{mm}$, $0.82\,\mathrm{mm}$, $0.72\,\mathrm{mm}$, for $i = 1, 2, 3$. Fig. 3 shows violin plots of the error distributions for e_{α_i} and e_{β_i} for values of m from 2 to 20.

Also, for all values of m, the percentage of valid estimations is constantly higher than 95 %. This indicates that our S2J-FNN is able to learn the constraints of the CTCR joint space in most cases, without specifically enforcing these during training. In terms of the estimation time, the average computing time is 0.40 s for 10 000 samples in the test set running on a GeForce RTX 2080 Ti.

IV. Numerical Shape-to-joint IK

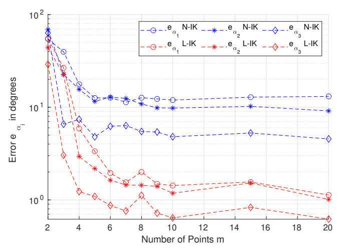
To compare the performance of our learning-based method with the state-of-the-art, we implement a vanilla numerical S2J-IK.

A. Algorithm Description

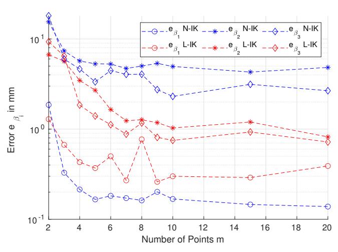
As for the learning-based method, we consider equidistant points along the robot's length (see Sec. III-A). These m discrete points p_j represent the desired target shape and serve as an input to the numerical approach to S2J-IK. Based on this target shape a numerical nonlinear optimization routine is implemented to find joint values q that correspond to the discretely defined target shape. This numerical method aims to minimize the squared residual in the shape error between the discrete target shape p_j and $\hat{p}_j$, computed by the FK [3] for the joint values q, using the same number of m equally distributed points:

$$\min_q \sum_{j=2}^{m} ||p_j - \hat{p}_j||_2^2 . \quad (16)$$

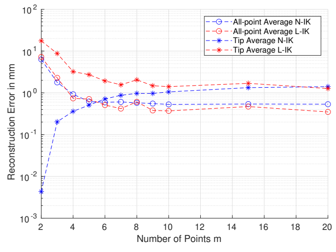
At the same time, the constraints on q in (1)-(4) have to be respected. To solve this constrained nonlinear optimization



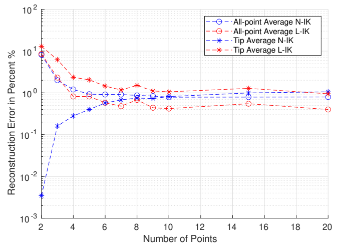
(a) Median e_{α_i}



(b) Median e_{β_i}



(c) Average e^{abs}



(d) Average e^{rel}

Fig. 2: Median and average joint space and reconstruction errors for S2J-IK. L-IK for learning-based S2J-FNN method and N-IK for numerical IK method.

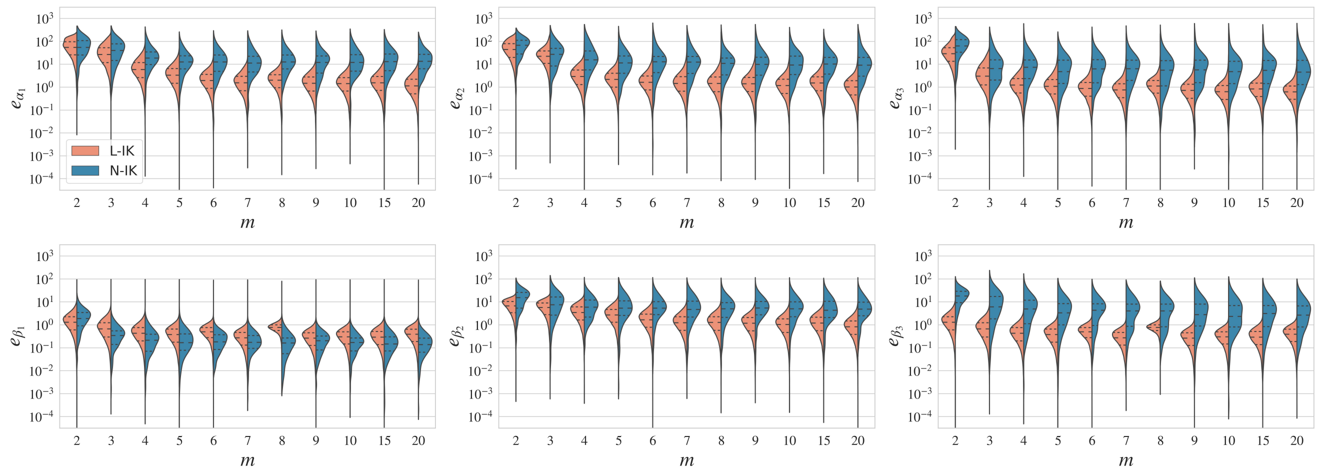


Fig. 3: Violin plots for error distributions over the number of points m for S2J-IK. L-IK for learning-based S2J-FNN method and N-IK for numerical IK method. Errors e_{α_i} and e_{β_i} are in degrees and in millimeters, respectively.

problem, a derivative-free algorithm based on linear approximations as proposed in [24] is used. The choice of an initial guess q_{init} highly affects the performance and convergence of the numerical S2J-IK. To improve the convergence behavior, we run the optimization problem expressed in (16) with 27 different initial guesses and pick the resulting joint values with the lowest remaining residuals. The considered 27 initial guesses use the same values for β_i, so that each tube is approximately retracted by half its length. For the rotation of each tube, we consider three different values $\alpha_i \in \{-2\pi/3, 0, 2\pi/3\}$, distributing them equally in their valid ranges.

B. Results

The numerical approach for S2J-IK is implemented in C++ using the open-source library [25] and is executed using an Intel Core i5-3470 processor with 4 cores at 3.20 GHz. Its computing time per sample is between 2 s to 3 s.

The approach is evaluated with values of m from 2 to 20. For each m, 100 different q_{target} are selected from the test set used for learning-based S2J-IK in Sec. III. The performance is measured by the errors in joint space and task space, as defined in Sec. III-E. The results are shown in Fig. 2 alongside the errors of the learning-based S2J-IK. With $m = 20$, the median e_{α_i} are 13.10°, 9.13°, 4.55°, and median e_{β_i} are 0.14 mm, 4.83 mm, 2.66 mm, for $i = 1, 2, 3$. From the error distributions in Fig. 3 we can observe wider spreads for N-IK for larger m.

C. Comparison S2J N-IK and L-IK

For both learning-based and numerical S2J-IK, we can see that with $m = 2$, which only considers the tip, the joint space errors are high as shown in Fig. 2a and Fig. 2b. As m increases, the errors decrease and remain relatively constant for $m \geq 10$. In general, one can observe that $e_{\alpha_3} < e_{\alpha_2} < e_{\alpha_1}$ and $e_{\beta_1} < e_{\beta_3} < e_{\beta_2}$ for a given m. With an exception for e_{β_1}, the learning-based S2J-IK has lower e_{α_i} and e_{β_i} for $m \geq 4$. More importantly, by examining the error distributions shown in Fig. 3, the learning-based S2J-IK is in general more accurate in terms of the median value and width of the distribution. However, while both S2J-IK approaches generate more than 95 % valid joint configurations, the numerical S2J-IK outperforms learning-based S2J-IK by 2 % to 5 %. Nevertheless, the learning-based S2J-IK is faster to compute.

Regarding the discretization of the shape with equidistant points, errors in the joint space shown in Fig. 2 and the reconstruction errors shown in Fig. 3 converge to relatively constant values for $m \geq 10$. Especially, the appearances of the error distributions remain similar as shown in Fig. 3.

V. Image-based Shape-to-joint IK

Our second approach is to learning-based IK from shape information uses multiple-view images of a CTCR. We refer to this mapping as image-to-joint inverse kinematics (I2J-IK).

A. Problem Definition

We use a triplet of 3 grey scale images $V_k = \{v_{k1}, v_{k2}, v_{k3}\}$ as input. Each image is obtained from a camera fixed in location relative to a base frame and the desired shape is completely depicted in the image. The number of views is chosen to be 3 to avoid possible self-occlusions. The desired output of the I2J-IK for a given triplet V_k is $q \in \mathbb{R}^9$.

B. I2J-CNN Structure

We exploit convolutional layers for processing images from different view points as input. We choose a resolution of 500×500. It is composed of 3 parallel convolutional parts, consisting of convolutional layers and average pooling layers; each part takes images from a fixed view point for input and abstracts relevant features. The outputs from 3 convolutional parts are then concatenated and fed to 2 fully connected layers. The resulted convoluational neural network is called I2J-CNN and its structure is listed in Table IV.

TABLE IV: I2J-CNN structure

Layer Name	Characteristic
Convolutional part $\times$ 3	Input Layer, input dimension $(500, 500, 3)$
	Convolutional layer with 4 filters, kernel size $(2, 2)$, stride (1,1)
	Average pooling layer with kernel size $(2, 2)$, stride=2
	Conv2, 2 filters with kernel size $(2, 2)$, stride (1,1)
	Average pooling layer with kernel size $(2, 2)$
	Convolutional layer with 4 filters, kernel size $(2, 2)$, stride (1,1)
	Average pooling layer with kernel size $(2, 2)$
	Flatten layer with 29 768 neurons
Concatenation	89 304 Neurons
FC1	600 Neurons with ReLU activation function
FC2	300 Neurons with ReLU activation function
Output Layer	9 Neurons with linear activation function

C. Data Generation

Following similar procedure as for S2J-FNN in Sec. III, a data set of size 100 000 with q is generated and referred to as $\mathbb{Q}_2$. Joint values q have identical joint ranges except for $\alpha_i \in [-\pi/3, \pi/3]$. To generate images, we utilize our CTCR 3d simulation environment using the Visualization Toolkit (Kitware, Inc., New York, USA). For each q_k in $\mathbb{Q}_2$ the CTCR is rendered and a triplet of grey scale images V_k is generated from three different but fixed virtual camera locations. The collection of V_k is denoted as $\mathbb{V}_2$.

D. I2J-CNN Training

The I2J-CNN is evaluated on the data set $\{\mathbb{V}_2, \mathbb{Q}_2\}$ with ratio 8:1:1 for training, validation, and testing. I2J-CNN is implemented in Pytorch 1.4.0 trained by Adam Optimizer at learning rate $\lambda = 10^{-4}$, decay rates 0.9 and 0.999, and mini batch size of 50, for 100 epochs. The loss function (8) with $\omega = 200$ is used. This process is repeated for 3 trials with different weight initialization.

E. Evaluation

The estimation errors of I2J-CNN are evaluated both in joint and Cartesian space using e_{α_i} and e_{β_i} as defined in (10) and (11). We also determine the reconstruction error as outlined in Sec. III-E.

F. Results

Figure 4 shows the error distributions for I2J-CNN trial 1. The median errors are $e_{\alpha_1} = 2.87°$, $e_{\alpha_2} = 2.04°$, $e_{\alpha_3} = 0.77°$, $e_{\beta_1} = 1.71\,\text{mm}$, $e_{\beta_2} = 1.23\,\text{mm}$, and $e_{\beta_3} = 0.88\,\text{mm}$. All results are summarized in Table V.

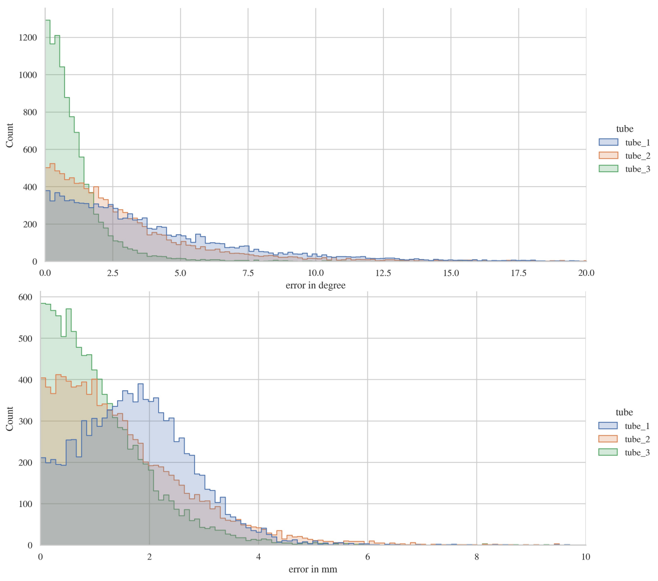


Fig. 4: Estimation error distributions of I2J-CNN for trial 1. For the sake of visualization, the distributions are truncated and, therefore, a few outliers are omitted.

Following the same steps as stated in Sec. III, 28 054 out of 30 000 (93.51 %) predictions of $\hat{q}$ in the testing stage are valid in the joint space. Their corresponding shapes are reconstructed and, afterwards, e_j^{abs} as well as e_j^{rel} are computed. Across all the points, the average errors are $0.72\,\text{mm}$ which is $0.75\,\%$ w.r.t. the corresponding robot length. Whereas the average error at the robot tip is $2.18\,\text{mm}$ which is $1.59\,\%$ w.r.t. the robot length.

VI. Discussion

The performances of both learning-based approaches, i.e. S2J-FNN and I2J-CNN, are summarized in Table V. From the inner tube $i = 1$ to the outer tube $i = 3$, the rotational error e_{α_i} increases, for both S2J-FNN and I2J-CNN. This trend confirms that the outer tube has a larger influence on the overall shape of a CTCR, as it is the stiffest.

For comparison we include the results of our previous work in Table V, where we propose the tip-to-joint T2J-FNN inverse kinematics mapping [17]. The results of our S2J-FNN with $m = 20$ outperform the T2J-FNN by reducing the rotational error by $5.99\,\text{mm}$ and the translational error by $2.55°$. This shows that by considering the whole shape rather than only the tip position, the relationship between shape and

TABLE V: Estimation errors for joints parameters per trial for S2J-FNN ($m = 10, 15, 20$) and I2J-CNN in comparison to tip-to-joint (T2J-FNN) results from [17]. Median values for e_{α_i} and e_{β_i} are reported for both models, as well as averages of 3 trials for I2J-CNN.

		S2J-FNN evaluation trials			I2J-CNN evaluation trials				T2J-FNN [17]
Estimation Errors	Unit	$m = 10$	$m = 15$	$m = 20$	1st	2nd	3rd	Avg	Average error over 10 trials
e_{α_1}	degree	1.43	1.55	1.13	2.87	2.91	3.32	4.02	3.39 ± 0.70
e_{α_2}	degree	1.18	1.52	1.01	2.04	2.07	2.16	3.14	6.17 ± 0.27
e_{α_3}	degree	0.63	0.82	0.62	0.77	0.72	0.99	1.19	2.29 ± 0.21
e_{α}	degree	2.56	3.09	2.22	4.60	4.65	5.24	6.02	8.21 ± 0.28
e_{β_1}	mm	0.29	0.28	0.39	1.71	0.81	1.00	1.32	1.00 ± 0.17
e_{β_2}	mm	1.03	1.20	0.82	1.23	1.19	1.62	1.62	2.00 ± 0.19
e_{β_3}	mm	0.75	0.93	0.72	0.88	1.00	1.05	1.20	2.70 ± 0.78
e_{β}	mm	1.61	1.90	1.45	2.73	2.28	2.74	2.76	4.00 ± 0.60

joint space is more unique and suitable for learning-based methods. Our results also suggest that a CTCR shape can be uniquely represented by 10 or more equidistant points along the shape. Also, notably numerical S2J method achieves smaller errors at the tip with $m < 10$. This indicates that a numerical IK method is preferable for target shape defined by few points. For future work, leveraging the learning-based S2J-IK as a prior for the physic-based S2J-IK is desirable unifying the strengths of both approaches.

With simulated data, our proposed approaches also outperform related work in terms of reconstruction errors. For our S2J-FNN ($m = 20$) the average reconstruction error of $0.35\,\mathrm{mm}$ and maximum reconstruction error e_j^{abs} of $1.27\,\mathrm{mm}$ for the CTCR tip is lower than the learning-based joint-to-shape reconstruction error obtained in [20]. In terms of relative error, the average reconstruction error is $0.40\,\%$ and a maximum relative error e_j^{rel} of $0.95\,\%$ are obtained for the tip. These results are lower than the $3\,\%$ error of the physics-based FK [3] which is most widely used for CTCR. Therefore, we suspect that our methodology may also perform well with a data set of a real CTCR prototype.

There is no canonical choice for the representation of a shape, due to the continuously elastic deformation of the CTCR shape. Existing representations rely on model assumptions and simplifications, e.g. constant-curvature assumption. A discrete representation depends on the number of equidistant points along the shape as shown in Fig. 3 and Fig. 2. A tendency can be observed: the higher the number, the higher the accuracy. Note, that the accuracy not only depends on the number of points, but also on their position and the underlying model [26]. Shape representation based on images, on the other hand, can be considered as a representation with less simplifications, as the continuous curve and all points along the shape can be theoretically captured. As a side note, the output of the convolution layer before the fully connected layer can be interpreted as a shape representation as well, since convolution layers act as an encoder reducing dimensionality of an input image.

One inherent limitation of a learning-based approach for the S2J-IK is that it relies on large amounts of data of known CTCR shapes. A pre-trained S2J-FNN or I2J-FNN with simulation data may be used as a starting point for transfer learning with a real CTCR prototype to reduce the required amount of real data. In this case with real CTCR data available, a learning-based approach has the potential to address unmodeled effects in the selected FK model for the numerical method and may result in improved accuracy.

VII. Conclusion

In this paper, we describe the shape-to-joint inverse kinematics for CTCR and propose two learning-based approaches. The feasibility is demonstrated in simulation. Two approaches are used to describe the shape of a concentric tube continuum robot; as a set of equidistant points along the shape and as an image from different view points. The former approach is compared with a numerical IK approach and shows higher accuracy for different discretization. The latter achieves promising results while bypassing the choice of discretization.

Effectively solving the inverse kinematics for shape input is an important prerequisite for effective open-loop and closed-loop control [27]. Compared to tip position or pose control, we believe that shape-to-joint inverse kinematics and the proposed approaches can enable more advanced controllers and applications.

References

[1] J. Burgner-Kahrs, D. C. Rucker, and H. Choset, "Continuum robots for medical applications: A survey," *IEEE Transactions on Robotics*, vol. 31, no. 6, pp. 1261–1280, 2015.

[2] 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.

[3] 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.

[4] J. Lock and P. E. Dupont, "Friction modeling in concentric tube robots," in *IEEE International Conference on Robotics and Automation*, 2011, pp. 1139–1146.

[5] J. Ha and P. E. Dupont, "Incorporating tube-to-tube clearances in the kinematics of concentric tube robots," in *IEEE International Conference on Robotics and Automation*, 2017, pp. 6730–6736.

[6] C. Greiner-Petter and T. Sattel, "On the influence of pseudoelastic material behaviour in planar shape-memory tubular continuum structures," *Smart Materials and Structures*, vol. 26, no. 12, p. 125024, 2017.

[7] H. B. Gilbert, R. J. Hendrick, and R. J. Webster III, "Elastic stability of concentric tube robots: A stability measure and design test," *IEEE Transactions on Robotics*, vol. 32, no. 1, pp. 20–35, 2015.

[8] C. Kim, S. C. Ryu, and P. E. Dupont, “Real-time adaptive kinematic model estimation of concentric tube robots,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2015, pp. 3214–3219.

[9] V. Modes and J. Burgner-Kahrs, “Calibration of concentric tube continuum robots: Automatic alignment of precurved elastic tubes,” *IEEE Robotics and Automation Letters*, vol. 5, no. 1, pp. 103–110, 2019.

[10] J. Granna, I. S. Godage, R. Wirz, K. D. Weaver, R. J. Webster III, and J. Burgner-Kahrs, “A 3-d volume coverage path planning algorithm with application to intracerebral hemorrhage evacuation,” *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 876–883, 2016.

[11] A. Garriga-Casanovas and F. Rodriguez y Baena, “Kinematics of continuum robots with constant curvature bending and extension capabilities,” *Journal of Mechanisms and Robotics*, vol. 11, no. 1, p. 011010 (12 pages), 2018.

[12] P. Sears and P. E. Dupont, “Inverse kinematics of concentric tube steerable needles,” in *IEEE International Conference on Robotics and Automation*, 2007, pp. 1887–1892.

[13] R. J. Webster, J. P. Swensen, J. M. Romano, and N. J. Cowan, “Closed-form differential kinematics for concentric-tube continuum robots with application to visual servoing,” in *Experimental Robotics*, 2009, pp. 485–494.

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

[15] 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, 2013.

[16] C. Bergeles, F. Lin, and G. Yang, “Concentric tube robot kinematics using neural networks,” in *Hamlyn Symposium on Medical Robotics*, 2015, pp. 1–2.

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

[18] R. M. 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.

[19] K. Iyengar, G. Dwyer, and D. Stoyanov, “Investigating exploration for deep reinforcement learning of concentric tube robot control,” *International Journal of Computer Assisted Radiology and Surgery*, vol. 15, pp. 1157–1165, 2020.

[20] 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.

[21] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, “Automatic differentiation in pytorch,” 2017.

[22] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv:1412.6980*, 2014.

[23] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010, pp. 249–256.

[24] M. J. D. Powell, “A direct search optimization method that models the objective and constraint functions by linear interpolation,” in *Advances in optimization and numerical analysis*. Springer, 1994, pp. 51–67.

[25] S. G. Johnson, “The nlopt nonlinear-optimization package,” 2011.

[26] A. Mahoney, T. Bruns, R. Alterovitz, and R. Webster, “Design, sensing, and planning: Fundamentally coupled problems for continuum robots,” in *Robotics Research*. Springer, 2018, pp. 267–282.

[27] M. T. Chikhaoui and J. Burgner-Kahrs, “Control of continuum robots for medical applications: State of the art,” in *International Conference and Exhibition on New Actuators and Drive Systems*. VDE VERLAG GMBH, 2018, pp. 154–164.