2021 27th International Conference on Mechatronics and Machine Vision in Practice (M2VIP) | 978-1-6654-3153-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/M2VIP49856.2021.9665066

Feedforward Neural Network Assisted Configuration Transition Control of Continuum Surgical Manipulators

Yifan Wang, Chuanxiang Zhu, Yue Ding, Bo Feng and Kai Xu, *Member, IEEE*

***Abstract*—Continuum manipulators have been increasingly used in surgical applications recently. A continuum surgical manipulator can be controlled to perform useful tasks when partially inserted into a patient's cavity, enlarging the workspace of the manipulator. During the insertion and retraction, the inserted portion of the manipulator achieves different configurations, and a configuration transition control framework was proposed based on the resolved motion rate control. However, the inverse kinematics (IK) under the configuration transition framework suffers from failures from time to time due to the reduced motion capability during configuration transition. This paper hence proposes to improve the IK performance of the configuration transition control with the assistance of a feedforward neural network (FNN). During the iteration to solve the IK, the priority and the value of linear and angular velocities are adjusted by the FNN which takes the current and target end effector poses as inputs. By properly adjusting the task priority and the values of the end effector twists, the converging trajectory of the manipulator is regulated to reduce the cases of being trapped in local minima. Numerical simulations were carried out to validate the proposed method. The results show that the proposed method achieves a higher success rate than the original configuration transition method.**

I. Introduction

Minimally Invasive Surgery (MIS) offers benefits such as less pain, quicker recovery, and lower postoperative complication risks [1], while also posing challenges to the surgeons. To address the operative challenges of MIS, numerous robot-assisted surgical platforms have been developed [2, 3]. Apart from articulated structures, continuum mechanisms are promising candidates for robotic surgical platforms [3] and are adopted in the designs of several surgical manipulators [4-7], due to their dexterity, structural compliance, and design compactness.

During surgical operations, a multi-segment continuum surgical manipulator is usually fully inserted through a trocar into a patient's abdominal cavity, as shown in Fig. 1(d). In this working pattern, an unreachable volume exists inside the translational workspace of the manipulator, as investigated in [8]. Nevertheless, a multi-segment continuum manipulator is capable of performing surgical tasks even when partially inserted, as shown in Fig. 1(a)-(c). Teleoperation in these partially inserted configurations can extend the manipulator's overall motion capability, even though the motion capability might be reduced in these configurations.

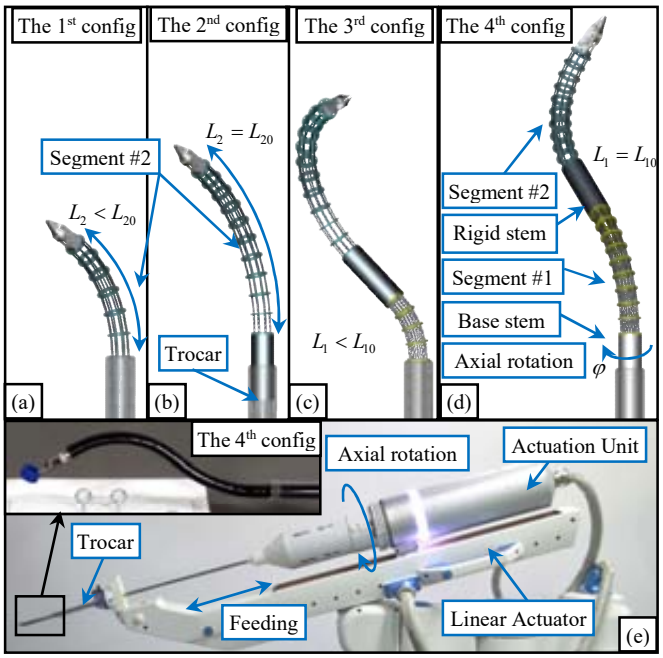

Figure 1. Different configurations of the multi-segment continuum surgical manipulator: (a) the 1st configuration, (b) the 2nd configuration, (c) the 3rd configuration, and (d) the 4th (fully inserted) configuration; (e) the manipulator's actuation scheme.

To realize the configuration transition control of a continuum manipulator, a kinematics framework was proposed previously [9]. Based on the resolved motion rate control [10], this framework solves the inverse kinematics (IK) under different configurations in a unified way. However, this framework can still fail when the manipulator is controlled among the partially inserted configurations, due to the reduced kinematic capability in such configurations.

Feedforward neural network (FNN), as a widely adopted supervised learning architecture, has been used in solving the IK and inverse statics problems of continuum manipulators [11-13]. The FNN usually consists of neurons that are connected by weights and biases and map the inputs to the outputs by nonlinear functions. By training the FNN using labeled data sets, the weights and biases are adjusted such that the FNN can approximate complex functions. Taking advantage of the universal approximation capability of the FNN, the nonlinear IK problem of the continuum manipulator can be quickly solved by the FNN after proper training. On the

*This work was supported in part by the National Key R&D Program of China (Grant No. 2017YFC0110800), in part by the National Natural Science Foundation of China (Grant No. 51722507), and in part by the Foundation of National Facility for Translational Medicine (Shanghai) (Grant No. TMSK-2021-505).

Yifan Wang, Chuanxiang Zhu, Yue Ding and Kai Xu are with the State Key Laboratory of Mechanical System and Vibration, School of Mechanical Engineering, Shanghai Jiao Tong University, Shanghai, 200240, China (e-mails: fan_tasy@sjtu.edu.cn, zcx_sjtu@sjtu.edu.cn, dingyue2020@sjtu.edu.cn, and k.xu@sjtu.edu.cn; corresponding author: Kai Xu).

Bo Feng is with Department of Surgery, Affiliated Ruijin Hospital, Shanghai Jiao Tong University, Shanghai, China, 200025 (e-mail: fengbo2022@163.com).

other hand, FNN can be used to construct nonlinear motion controllers and assist in the control of multi-segment continuum manipulators, accounting for uncertainties in the robot dynamics and the external disturbance [14, 15].

Inspired by the applications of the FNN in robot control, this paper proposes to improve the success rate of the configuration transition IK of a 2-segment continuum surgical manipulator. An FNN is used to predict the iteration result of the resolved motion rate method under different priorities between the linear and angular velocity. The task priority and the value of the end effector twists (i.e., linear and angular velocities) are adjusted according to the FNN prediction during iterations. This helps to regulate the converging trajectory and reduce the cases of being trapped in local minima. The effectiveness of the proposed method is validated using numerical simulations.

The rest of this paper is organized as follows. Section II defines the configurations of the manipulator, while the neural network assisted control framework is elaborated in Section III. Numerical simulation results are presented in Section IV with the conclusions summarized in Section V.

II. Kinematics Nomenclature and Configuration Transition Control

A. Kinematics Nomenclature

Based on the constant curvature assumption, the modeling of a single continuum segment is shown in Fig. 2, while the nomenclatures are listed in Table I. The kinematics of the manipulator can be referred to [9].

TABLE I. Nomenclature Used in the Kinematics Model

Symbol	Definition
t	Index of the segments. t = 1, 2.
j	Index of the configurations. j = 1, 2, 3, 4.
L_t, L_{t0}	Inserted length and the full length of the t^{th} segment.
L_r, L_{r0}	Inserted length and the full length of the rigid stem.
L_s, L_{s0}	Inserted length and the full length of the base stem.
θ_t	Bending angle of the t^{th} continuum segment.
δ_t	Bending direction angle of the t^{th} continuum segment.
φ	Axial rotation realized by the actuation unit.
$\mathbf{v}$, $\boldsymbol{\omega}$	Desired linear and angular velocity of the end effector.
$\boldsymbol{\psi}_j$	The configuration variable vector of the manipulator in the j^{th} configuration.
$\mathbf{J}_j$	The manipulator's Jacobian matrix in its j^{th} configuration.
$\mathbf{J}_{jv}$, $\mathbf{J}_{j\omega}$	Jacobian matrices of the tip velocity and angular velocity in the j^{th} configuration.

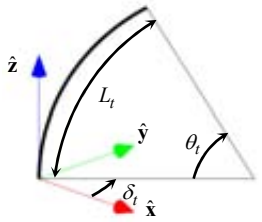

Figure 2. Kinematics modeling of the t^{th} bending segment.

B. Configuration Definition

The manipulator has two inextensible continuum segments with a rigid straight stem in between. The segment #1 is stacked on a base stem, as shown in Fig. 1(d). The manipulator is mounted onto an actuation unit, which provides rotation about and translation along its axis, as shown in Fig. 1(e).

When an inextensible bending segment is partially inserted, the inserted portion is kinematically treated as possessing 3 DoFs: 2-DoF bending and 1-DoF length changing. This insertion process hence leads to 4 configurations as follows.

- The 1st Configuration (C1): the segment #2 is partially inserted as shown in Fig. 1(a). The manipulator has 4 DoFs: axial rotation, length changing, and 2-DoF bending of the segment #2. The configuration verctor is $\boldsymbol{\psi}_1 = [\varphi\ \theta_2\ L_2\ \delta_2]^T$.
- The 2nd Configuration (C2): the rigid stem is partially inserted as in Fig. 1(b). The manipulator has 4 DoFs: axial rotation and feeding of the rigid stem, as well as 2-DoF bending of the segment #2. The configuration vector is $\boldsymbol{\psi}_2 = [\varphi\ L_r\ \theta_2\ \delta_2]^T$.
- The 3rd Configuration (C3): the segment #1 is partially inserted as in Fig. 1(c). The manipulator has 6 DoFs: 2-DoF bending of the segment #2, as well as axial rotation, length changing and 2-DoF bending of the segment #1. The configuration vector is $\boldsymbol{\psi}_3 = [\varphi\ \theta_1\ L_1\ \delta_1\ \theta_2\ \delta_2]^T$.
- The 4th Configuration (C4): the base stem is partially inserted as in Fig. 1(d). The manipulator has 6 DoFs: 2-DoF bending of for each of the segment #1 and #2, as well as axial rotation and feeding of the base stem. The configuration vector is $\boldsymbol{\psi}_4 = [\varphi\ L_s\ \theta_1\ \delta_1\ \theta_2\ \delta_2]^T$.

C. Configuration Transition

The configuration transition is triggered when a length variable exceeds its limit during the insertion of the manipulator. The triggering variables for the configuration transition are summarized in Fig. 3, while the strategies are explained as follows.

- Between C1 and C2: the manipulator configuration with $L_2 = L_{20}$ in C1 is identical to that with $L_r = 0$ in C2. The transition from C1 to C2 is triggered when L_2 is updated longer than L_{20}. The overshoot of L_2 (excess value after one update) in C1 is set to L_r in C2:

$$(L_r)_{C2} = (L_2)_{C1} - L_{20}\,. \tag{1}$$

The transition from C2 to C1 is triggered when L_r is updated less than 0. The overshoot of L_r (a negative value) in C2 is set to L_2 in C1:

$$(L_2)_{C1} = (L_r)_{C2} + L_{20}\,. \tag{2}$$

- Between C2 and C3: the manipulator configuration with $L_r = L_{r0}$ in C2 is identical to that with $L_1 = 0$ in C3. The transition from C2 to C3 is triggered when L_r is updated longer than L_{r0}. The overshoot of L_r in C2 is set to L_1 in C3:

$$(L_1)_{C3} = (L_r)_{C2} - L_{r0}\,. \tag{3}$$

The transition from C3 to C2 is triggered when L_1 is updated less than 0. The overshoot of L_1 (a negative value) in C3 is set to L_r in C2:

$$(L_r)_{C2} = (L_1)_{C3} + L_{r0}\,. \tag{4}$$

- Between C3 and C4: the manipulator configuration with $L_1 = L_{10}$ in C3 is identical to that with $L_s = 0$ in C4. The transition from C3 to C4 is triggered when L_1 is updated longer than L_{10}. The overshoot of L_1 in C3 is set to L_s in C4:

$$(L_s)_{C4} = (L_1)_{C3} - L_{10}\,. \tag{5}$$

The transition from C4 to C3 is triggered when L_s is updated less than 0. The overshoot of L_s (a negative value) in C4 is set to L_1 in C3:

$$(L_1)_{C3} = (L_s)_{C4} + L_{10}\,. \tag{6}$$

After a configuration transition, the configuration variables other than the length variables are inherited from the previous configuration.

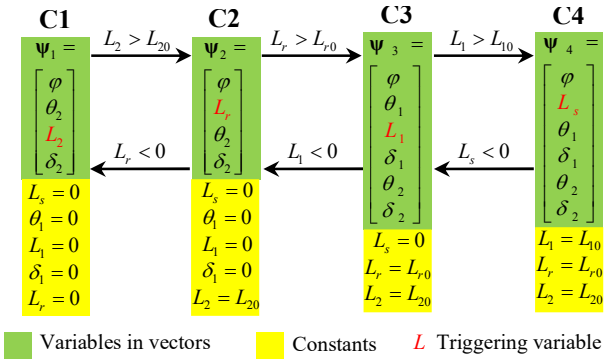

Figure 3. The variables and constants of the manipulator and the transition conditions between the adjacent configurations.

D. Prioritized Resolved Motion Rate Control

The resolved motion rate control [10] iteratively updates the configuration variables by using the Jacobian to achieve the desired end effector twist, which is obtained as:

$$\dot{\mathbf{x}} = \begin{bmatrix} \mathbf{v} \\ \boldsymbol{\omega} \end{bmatrix} = \begin{bmatrix} k_v \mathbf{e}_P \\ k_w \mathbf{e}_R \end{bmatrix}, \tag{7}$$

where $\mathbf{e}_P$ and $\mathbf{e}_R$ denote the position and orientation error to the target pose, respectively, while k_v and k_w are proportional gains. In Configuration C1 and C2, the manipulator only possesses 4 DoFs and may not always be able to reach a desired position and orientation at the same time. Therefore, the linear and angular velocities are separately treated, with one of them chosen as the primary task and the other as the secondary task. The priority is handled by the null space projection method as in (8) [16]:

$$\dot{\boldsymbol{\psi}}_j = \mathbf{J}_{jp}^{+}\dot{\mathbf{x}}_p + (\mathbf{J}_{js}(\mathbf{I} - \mathbf{J}_{jp}^{+}\mathbf{J}_{jp}))^{+}(\dot{\mathbf{x}}_s - \mathbf{J}_{js}\mathbf{J}_{jp}^{+}\dot{\mathbf{x}}_p)\,, \tag{8}$$

where $\dot{\mathbf{x}}_p$ and $\dot{\mathbf{x}}_s$ denote the primary and secondary tasks (selecting from $\mathbf{v}$ and $\boldsymbol{\omega}$), respectively; $\mathbf{J}_{jp}$ and $\mathbf{J}_{js}$ denote the Jacobians of the primary and secondary tasks in the j^{th} configuration (selecting from $\mathbf{J}_{jv}$ and $\mathbf{J}_{j\omega}$), respectively. $\mathbf{J}^{+}$ represents the Moore-Penrose pseudo-inverse of $\mathbf{J}$.

III. Feedforward Neural Network Assisted Resolved Motion Rate Control

While solving the IK problem, the resolved motion rate control method relies on the proper setting of the task priority and the end effector twists to successfully converge. Improper task priority or end effector twists can lead to local minima while traversing multiple configurations, such as converging to the target position with a different orientation [9]. To alleviate this problem, an FNN is used to adaptively adjust the priority and values of the end effector twist based on the current and target end effector poses.

A. Feedforward Neural Network

An FNN consists of several layers of processing units called neurons, including an input layer, several hidden layers, and an output layer. The input layer takes an input vector into the neural network, with the number of neurons equal to the dimension of the input vector. In hidden layers, each neuron is fully connected to the neurons in the previous layer by weights and biases. The output of a neuron in the hidden layer is the weighted and biased sum of the output of the previous layer processed by a non-linear activation function $\sigma(\cdot)$. For the i^{th} hidden layer, its output vector $\mathbf{u}^i$ is calculated as

$$\mathbf{u}^i = \sigma(\mathbf{W}^i\mathbf{u}^{i-1} + \mathbf{b}^i)\,, \tag{9}$$

where $\mathbf{u}^{i-1}$ is the output of the $i-1^{\text{th}}$ layer. $\mathbf{W}^i$ is the weight matrix of the i^{th} layer with its elements w_{lk} being the weight connecting the k^{th} neuron in the $i-1^{\text{th}}$ layer and the l^{th} neuron in the i^{th} layer. $\mathbf{b}^i$ is the bias vector of the i^{th} layer with its elements b_l being the bias of the l^{th} neuron in the i^{th} layer. The output layer is fully connected to the last hidden layer and generates the output of the whole neural network.

In this study, the FNN is trained as a classifier, taking the current and target end effector poses as the network input and predicting the iteration result of the input case using different task priority settings. The FNN consists of an input layer, three hidden layers each containing 30 neurons with the hyperbolic tangent function as the activation function ($\sigma(\cdot) = \tanh(\cdot)$), and a softmax output layer with four output neurons, corresponding to four categories:

#1) the manipulator can reach the target pose with either linear or angular velocities as the primary task;

#2) the manipulator can reach the target pose only with the linear velocity as the primary task;

#3) the manipulator can reach the target pose only with the angular velocity as the primary task;

#4) the manipulator cannot reach the target pose with either linear or angular velocities as the primary task.

The input consists of two $SE(3)$ poses and typically has 12 elements. However, using kinematic properties of the continuum manipulator, the input dimension can be reduced as described below.

Since configuration variables φ, δ_1 and δ_2 are not limited, the workspace of the manipulator is rotationally symmetric about the axis of the base stem. Moreover, the rotation around the end effector axis can be independently generated by φ, δ_1 and δ_2 without changing the shape of the manipulator, and

does not affect the convergence of the position and pointing direction of the end effector in the resolved motion rate control. Therefore, the FNN only concerns about the 2-dimensional pointing direction of the end effector.

A fixed world coordinate $\{w\}$ is defined such that its origin coincides with the trocar exit and its $\hat{\mathbf{z}}_w$ axis aligns with the trocar axis. The current and target positions with respect to $\{w\}$ are denoted by ${}^w\mathbf{p}_c = [{}^wp_{cx}\ {}^wp_{cy}\ {}^wp_{cz}]^T$ and ${}^w\mathbf{p}_t = [{}^wp_{tx}\ {}^wp_{ty}\ {}^wp_{tz}]^T$, respectively. The directions of the current and target end effector axes with respect to $\{w\}$ are denoted by ${}^w\mathbf{a}_c = [{}^wa_{cx}\ {}^wa_{cy}\ {}^wa_{cz}]^T$ and ${}^w\mathbf{a}_t = [{}^wa_{tx}\ {}^wa_{ty}\ {}^wa_{tz}]^T$, respectively.

Since the workspace is rotationally symmetric, rotating the current and target poses around $\hat{\mathbf{z}}_w$ does not affect the iteration result. A coordinate $\{s\}$ is defined for each target pose by rotating $\{w\}$ around $\hat{\mathbf{z}}_w$ for an angle $\gamma = \arctan({}^wp_{ty}/{}^wp_{tx})$ such that the XZ-plane of $\{s\}$ contains ${}^w\mathbf{p}_t$, as shown in Fig. 4(a). The current and target positions as well as the end effector directions are represented in $\{s\}$ as:

$$\begin{aligned} {}^s\mathbf{p}_c &= \mathrm{Rot}(\hat{\mathbf{z}},-\gamma)\,{}^w\mathbf{p}_c, \quad {}^s\mathbf{a}_c = \mathrm{Rot}(\hat{\mathbf{z}},-\gamma)\,{}^w\mathbf{a}_c, \\ {}^s\mathbf{p}_t &= \mathrm{Rot}(\hat{\mathbf{z}},-\gamma)\,{}^w\mathbf{p}_t, \quad {}^s\mathbf{a}_t = \mathrm{Rot}(\hat{\mathbf{z}},-\gamma)\,{}^w\mathbf{a}_t. \end{aligned} \tag{10}$$

where $\mathrm{Rot}(\hat{\mathbf{z}},-\gamma)$ is the rotation matrix around $\hat{\mathbf{z}}$ for an angle $-\gamma$, and ${}^sp_{ty} = 0$. Then ${}^sp_{ty}$ can be removed from the input vector.

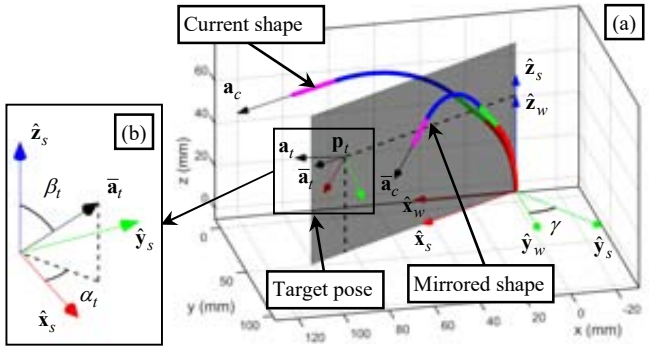

Figure 4. (a) Definition of coordinate $\{s\}$ and the mirrored current pose, and (b) spherical coordinate representation of the end effector direction.

Furthermore, due to the symmetry of the workspace, mirroring the current and target poses about the XZ-plane of $\{s\}$ does not affect the iteration results. If the current position is on the negative side of the $\hat{\mathbf{y}}_s$ axis, the current and target poses are mirrored about the XZ-plane of $\{s\}$ as in (11). This mirroring reduces the range of the current end effector position and facilitates the training of the FNN.

$${}^s\bar{p}_{cy} = \left|{}^sp_{cy}\right|, \quad {}^s\bar{a}_{cy} = \mathrm{sgn}({}^sp_{cy})\,{}^sa_{cy}, \quad {}^s\bar{a}_{ty} = \mathrm{sgn}({}^sp_{cy})\,{}^sa_{ty}. \tag{11}$$

After the mirroring, the end effector directions are represented in spherical coordinates as shown in Fig. 4(b):

$$\begin{aligned} \alpha_c &= \arctan({}^s\bar{a}_{cy}/{}^sa_{cx}), \quad \beta_c = \arccos({}^sa_{cz}), \\ \alpha_t &= \arctan({}^s\bar{a}_{ty}/{}^sa_{tx}), \quad \beta_t = \arccos({}^sa_{tz}). \end{aligned} \tag{12}$$

The input of the neural network $\mathbf{v}_{in}$ is hence reduced to a 9-dimensional vector given as in (13). To avoid saturation of the tanh function, the input is normalized by projecting each element to the interval $[-1,1]$.

$$\mathbf{v}_{in} = \left[{}^sp_{cx}\ \ {}^s\bar{p}_{cy}\ \ {}^sp_{cz}\ \ \alpha_c\ \ \beta_c\ \ {}^sp_{tx}\ \ {}^sp_{tz}\ \ \alpha_t\ \ \beta_t\right]^T. \tag{13}$$

B. Data Generation, Preprocessing, and Network Training

The data set for network training is generated as follows. 125,000 poses are generated for each configuration by randomly assigning values to the configuration variables within the allowed ranges and calculating the forward kinematics. Then a total of 500,000 poses from the four configurations are randomly arranged to generate the initial and the target poses for the training cases. The IK for the target poses is solved twice from the corresponding initial configurations using the prioritized resolved motion rate configuration transition control described in Section II.D, with the linear and the angular velocities as the primary task, respectively. The proportional gains for the end effector twist are set as $k_v = 10$ and $k_w = 8$. The IK successfully converges when the position error is less than 0.1 mm, the orientation error is less than 0.02 rad, and the iteration number is less than 200. The results are classified into four categories described in Section III.A as the desired output of the FNN.

Among the 500,000 cases, category #1 contains 442,673 cases, category #2 contains 29,507 cases, category #3 contains 22,483 cases, and category #4 contains 5,337 cases. To keep the balance of the training data set, 25,995 cases (average number of cases in category #2 and #3) are randomly selected from category #1 and other cases in category #1 are discarded. Also, the cases in category #4 are repeated 5 times (a total of 26,685 cases). Note that this repeat puts larger weights on the accuracy over category #4, which is desirable since it is more important to detect a case that fails in both priority settings and further adjust the end effector twist. After the preprocessing, the data set contains a total of 104,670 cases. The data set is randomly arranged into a training set (70%), a validation set (15%), and a test set (15%). The training set is used to train the FNN, with the validation set for early stop to avoid overfitting and the test set for evaluating the network performance.

The performance of the FNN is assessed by the cross entropy, and the training algorithm is scaled conjugate gradient. The training stops when either the cross entropy on the validation set does not decrease for 30 consecutive epochs or the number of epochs reaches the maximum number of 1,000. The training process eventually stopped at the 1,000th epoch and took 318 s using the Matlab deep learning toolbox on an Nvidia Geforce GTX1050 GPU. The training history is shown in Fig. 5, and the accuracy of the trained FNN on the test set is shown in Table II. The FNN achieved 93.4% overall accuracy in classification on the test set.

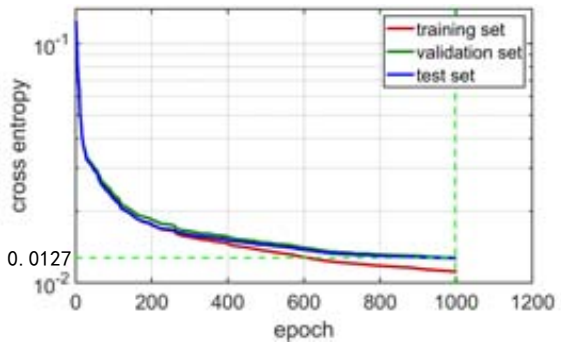

Figure 5. Performance history of the FNN during the training process.

TABLE II. FNN ACCURACY ON THE TEST SET

Number of cases		Ground truth category				Accuracy
		#1	#2	#3	#4	
Output category	#1	**3486**	34	64	13	96.9%
	#2	156	**4134**	20	172	92.2%
	#3	236	4	**3227**	28	92.3%
	#4	19	246	50	**3812**	92.4%
Accuracy		89.5%	93.6%	96.0%	94.7%	**93.4%**

C. *FNN Assisted Resolved Motion Rate Control*

The trained FNN is used to determine the task priority at each iteration during the resolved motion rate control. When the combination of the current and target pose is classified as category #1 or #2, the linear velocity is chosen as the primary task; when the combination is classified as category #3, the angular velocity is chosen as the primary task. Since there are still 5,337 cases in the data set that cannot be solved by altering the task priority, the end effector twist is further adjusted. To determine the proper end effector twists for cases in category #4, different combinations of linear and angular velocities as well as the task priorities were tested on category #4. The highest success rate was achieved by setting $k_v = 10$, $k_w = 32$, with the angular velocity as the primary task, where only 1384 cases fail to converge (0.28% of the 500,000 cases). Therefore, this setting is used for category #4.

The FNN assisted control framework is shown in Fig. 6. From the target pose, the position and orientation errors of the current pose are calculated. The task priority and values of the end effector twist are adjusted according to the classification given by the FNN. Then the configuration velocity is calculated by (7) and (8), and the configuration variables are updated. The configuration transition is conducted if triggered as described in Section II.C. And the current pose is updated by calculating the forward kinematics. The iteration continues until the errors are smaller than the error thresholds or the number of iterations exceeds the allowed number (200).

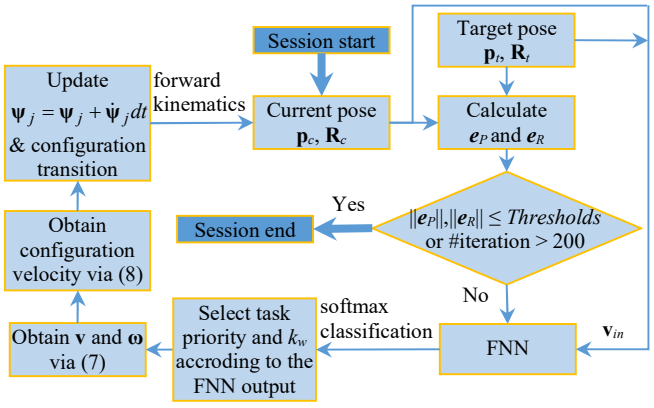

Figure 6. FNN assisted resolved motion rate control framework.

IV. NUMERICAL SIMULATION

The proposed method was tested in numerical simulations and compared to the original configuration transition control. The simulation was conducted on a total of 500,000 test cases generated in the same way as described in Section III.B.

The proposed method achieved 99.18% success rate (4,078 failed cases) on the test cases, showing an improved performance compared to the 94.36% success rate (28,211 failed cases) when the linear velocity is chosen as the primary task using the original method, and 92.34% success rate (38,303 failed cases) when the angular velocity is chosen as the primary task using the original method. Two representative cases are analyzed as follows.

A. *Case Study 1*

This case starts in C2 with the target pose in C3 whose position is lower than the initial position in the **z**-direction. The original method used the linear velocity as the primary task, and was stuck at the boundary between C1 and C2, as indicated by the red arrow in Fig. 7(a). In C1, reducing the position error requires extending and bending the segment #2, which drives the manipulator into C2. However, in C2, the position error cannot be further reduced by bending segment #2, and the manipulator has to retract the rigid stem to reduce the position error, leading the configuration back to C1. By contrast, in the proposed method, the FNN first recognized that the target pose cannot be reached by only altering task priority (category #4 as shown in Fig. 7(b)) and therefore set the angular velocity as the primary task with $k_v = 10$ and $k_w = 32$. The relatively large velocity drives the manipulator to extend the rigid stem and enter C3, as indicated by the dotted arrows in Fig. 7(a). From the configuration in C3, the end effector is more likely to converge to the target pose since the manipulator need not go through C1 or C2 with a reduced motion capability. Then, the FNN classified the current pose as category #2 and set the linear velocity as the primary task.

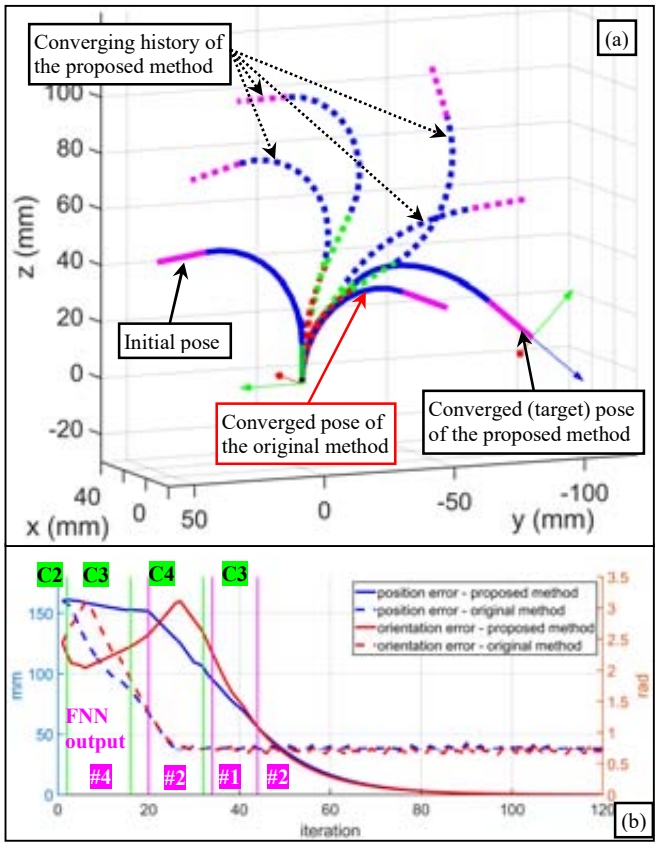

Figure 7. Case 1: (a) Visualization of the converging history. (b) History of the position and orientation errors of the two methods.

B. *Case Study 2*

The second case starts from C2 with the target pose in C3. The original method used the angular velocity as the primary task, and eventually converged to a pose that has the same orientation as the target pose but is still stuck in C2 with a position error around 82 mm, indicated by the red arrow in Fig. 8(a). As the priority of the angular velocity was enforced, the orientation converged before the position. Since the end effector position and orientation of the continuum manipulator are highly coupled, further reducing the position error would increase the orientation error, which is prohibited by the prioritized control scheme. By contrast, in the proposed method, the FNN first classified the case as in category #2, which means that the manipulator can only converge to the target pose by setting the linear velocity as the primary task. Then the task priority was altered and the manipulator converged to the target pose, despite a temporary increase in the orientation error as shown in Fig. 8(b).

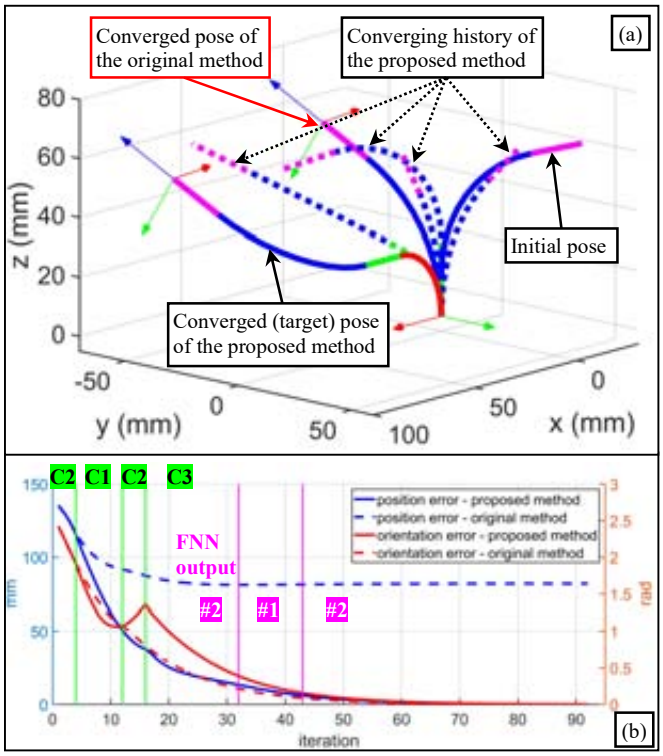

Figure 8. Case 2: (a) Visualization of the converging history. (b) History of the position and orientation errors of the two methods.

V. Conclusion

The configuration transition IK for continuum manipulators often fails in the configurations with reduced kinematic capability. To address this problem, this paper proposes to regulate the converging trajectory of the manipulator by properly setting the priority and value of the linear and angular velocities in real time. A data set is generated by calculating configuration transition IK using either the linear or angular velocity as the primary task. Based on the data set, an FNN is trained as a classifier to determine the task priority and the end effector twist setting during the entire resolved motion rate control process. The proposed method is compared to the original configuration transition control via extensive numerical simulations, showing that the proposed method achieved a higher success rate while solving the IK problem (success rates of 99.18% v.s. 94.36% and 92.34%).

References

[1] A. Cuschieri, "Laparoscopic Surgery: Current Status, Issues and Future Developments," *The Surgeon,* Vol. 3, No.3, pp. 125-138, June 2005.

[2] C. Bergeles and G.-Z. Yang, "From Passive Tool Holders to Microsurgeons: Safer, Smaller, Smarter Surgical Robots," *IEEE Transactions on Biomedical Engineering,* Vol. 61, No.5, pp. 1565-1576, May 2014.

[3] 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, Dec 2015.

[4] J. Ding, K. Xu, R. Goldman, P. K. Allen, D. L. Fowler, and N. Simaan, "Design, Simulation and Evaluation of Kinematic Alternatives for Insertable Robotic Effectors Platforms in Single Port Access Surgery," in *IEEE International Conference on Robotics and Automation (ICRA)*, Anchorage, Alaska, USA, 2010, pp. 1053-1058.

[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, April 2010.

[6] Y.-J. Kim, S. Cheng, S. Kim, and K. Iagnemma, "A Stiffness-Adjustable Hyperredundant Manipulator Using a Variable Neutral-Line Mechanism for Minimally Invasive Surgery," *IEEE Transactions on Robotics,* Vol. 30, No.2, pp. 382-395, April 2014.

[7] K. Xu, J. Zhao, and M. Fu, "Development of the SJTU Unfoldable Robotic System (SURS) for Single Port Laparoscopy," *IEEE/ASME Transactions on Mechatronics,* Vol. 20, No.5, pp. 2133-2145, Oct 2015.

[8] K. Xu, J. Zhao, and X. Zheng, "Configuration Comparison among Kinematically Optimized Continuum Manipulators for Robotic Surgeries through a Single Access Port," *Robotica,* Vol. 33, No.10, pp. 2025-2044, Dec 2015.

[9] S. a. Zhang, Q. Li, H. Yang, J. Zhao, and K. Xu, "Configuration Transition Control of a Continuum Surgical Manipulator for Improved Kinematic Performance," *IEEE Robotics and Automation Letters,* Vol. 4, No.4, pp. 3750-3757, Oct 2019.

[10] D. E. Whitney, "Resolved Motion Rate Control of Manipulators and Human Prostheses," *IEEE Transactions on Man-Machine Systems,* Vol. 10, No.2, pp. 47-53, June 1969.

[11] M. Rolf and J. J. Steil, "Efficient Exploratory Learning of Inverse Kinematics on a Bionic Elephant Trunk," *IEEE Transactions on Neural Networks and Learning Systems,* Vol. 25, No.6, pp. 1147-1160, Jun 2013.

[12] M. Giorelli, F. Renda, M. Calisti, A. Arienti, G. Ferri, and C. Laschi, "Neural Network and Jacobian Method for Solving the Inverse Statics of a Cable-Driven Soft Arm with Nonconstant Curvature," *IEEE Transactions on Robotics,* Vol. 31, No.4, pp. 823-834, 2015.

[13] T. G. Thuruthel, E. Falotico, M. Manti, A. Pratesi, M. Cianchetti, and C. Laschi, "Learning Closed Loop Kinematic Controllers for Continuum Manipulators in Unstructured Environments," *Soft Robotics,* Vol. 4, No.3, pp. 285-296, Sep 2017.

[14] D. Braganza, D. M. Dawson, I. D. Walker, and N. Nath, "A Neural Network Controller for Continuum Robots," *IEEE Transactions on Robotics and Automation,* Vol. 23, No.5, Dec 2007.

[15] Z. Wang, T. Wang, B. Zhao, Y. He, Y. Hu, B. Li, P. Zhang, and M. Q. H. Meng, "Hybrid Adaptive Control Strategy for Continuum Surgical Robot Under External Load," *IEEE Robotics and Automation Letters,* Vol. 6, No.2, pp. 1407-1414, 2021.

[16] D. N. Nenchev, "Restricted Jacobian Matrices of Redundant Manipulators in Constrained Motion Tasks," *The International Journal of Robotics Research,* Vol. 11, No.6, pp. 584-597, 1992.