# Deep Reinforcement Learning for Concentric Tube Robot Path Following

Keshav Iyengar, Sarah Spurgeon, *Fellow, IEEE* Danail Stoyanov *Senior Member, IEEE*

*Abstract*—As surgical interventions trend towards minimally invasive approaches, Concentric Tube Robots (CTRs) have been explored for various interventions such as brain, eye, fetoscopic, lung, cardiac, and prostate surgeries. Arranged concentrically, each tube is rotated and translated independently to move the robot end-effector position, making kinematics and control challenging. Classical model-based approaches have been previously investigated with developments in deep learning-based approaches outperforming more classical approaches in both forward kinematics and shape estimation. We propose a deep reinforcement learning approach to control where we generalize across two to four systems, an element not yet achieved in any other deep learning approach for CTRs. In this way, we explore the likely robustness of the control approach. Also investigated is the impact of rotational constraints applied on tube actuation and the effects on error metrics. We evaluate inverse kinematics errors and tracking errors for path-following tasks and compare the results to those achieved using state-of-the-art methods. Additionally, as current results are performed in simulation, we also investigate a domain transfer approach known as domain randomization and evaluate error metrics as an initial step toward hardware implementation. Finally, we compare our method to a Jacobian approach found in the literature.

*Index Terms*—Kinematics, Reinforcement Learning, Concentric Tube Robots
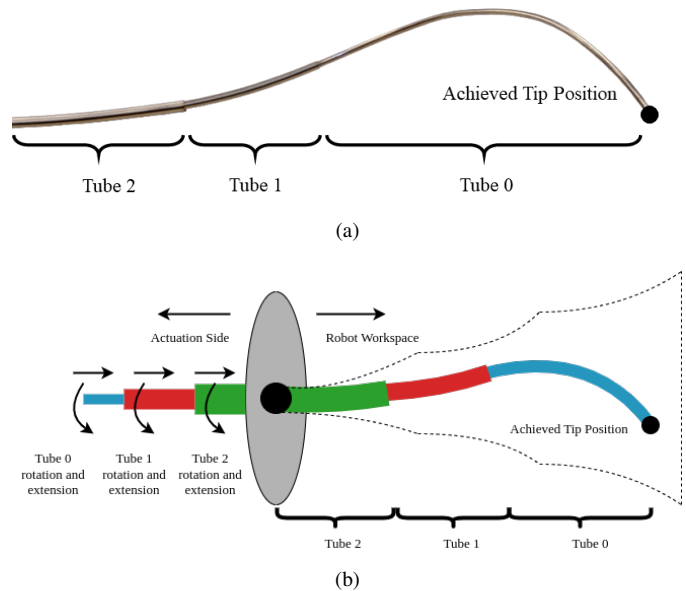


Fig. 1. A real set of concentric tubes [5] (a) and an equivalent illustration with tube actuation, workspace, and tip position.

## I. INTRODUCTION

CONCENTRIC tube robots (CTRs) are a class of continuum robots that depend on the interactions between neighboring, concentrically aligned tubes to produce the curvilinear shapes of the robot backbone [1]. The main application of these unique robots is that of minimally invasive surgery (MIS), where most of the developments for CTRs have been focused. MIS has trended towards semi-autonomous and autonomous robotic surgery to improve surgical outcomes [2], [3]. Due to the confined workspaces and resulting extended learning times for surgeons in MIS, dexterous, compliant continuum robots such as CTRs have been under development in preference to the mechanically rigid and limited degrees-of-freedom (DOF) robots used in interventional medicine

today. The pre-curved tubes of CTRs, which are sometimes referred to as active cannulas or catheters, are manufactured from super-elastic materials such as Nickel-Titanium alloys with each tube nested concentrically [4]. From the base, the individual tubes can be actuated through extension, as seen in Fig. 1, which results in the bending and twisting of the backbone as well as access to the surgical site through the channel and robot end-effector.

CTRs are motivated clinically for use in the brain, lung, cardiac, gastric, and other surgical procedures where tool miniaturization and non-linear tool path trajectories are beneficial [6]. Particularly, they have been investigated as steerable needles and surgical manipulators. As steerable needles, they substitute traditional steerable needles with higher precurvature and dexterity. They are also actuated with a follow-the-leader approach, where every point along the robot's backbone traces the same path as the tip. As surgical manipulators, the benefit of using CTRs is the large number of design parameters that allow for patient-specific and procedure-specific CTR designs through optimization for surgical requirements and design heuristics [7], [8]. In general, however, CTRs, with their increased DOFs and miniaturization potential, may be beneficial with their flexibility to reach a larger section of a surgical site with a working channel through the tubes

for irrigation, ablation, or other tools. Thus, it is key to control the robot tip position to desired Cartesian points in the robot workspace accurately. However, due to tube interactions, modeling, and control are challenging. Position control for CTRs has relied on model development, and although a balance between computation and accuracy has been reached in the literature [9], there remain issues such as performance in the presence of tube parameter discrepancies and the impact of unmodelled physical phenomena such as friction and permanent plastic deformation. This motivates the development of an end-to-end model-free control framework for CTRs.

One such model-free framework for robotic control that is gaining popularity is reinforcement learning (RL), a paradigm of machine learning that deploys an agent to output an action that interacts with an environment [10]. The environment then processes this action and returns a new state and, depending on the task, a reward signal. One parallel of RL is that of a control system. The agent is equivalent to the controller, the actions are equivalent to the control actions, the state is equivalent to any measurable signal from the plant, the reward is equivalent to a performance metric (eg. minimize steady-state error), and lastly, the learning algorithm is equivalent to the adaptive mechanism of the controller. Deep reinforcement learning (DeepRL) combines deep learning and RL and allows for high-dimensional states and actions traditionally not available to RL algorithms or classical control. In this work, we expand on the previously published literature on utilizing DeepRL to control CTRs [11]. Specifically, the aim is to control the end-effector Cartesian robot tip position with a DeepRL agent by means of actions that represent changes in joint values. The state includes the desired goal at the current step, allowing for more complex control tasks such as path following as well as inverse kinematics.

In §II, we review the Jacobian approach [12] as well as the state-of-the-art inverse kinematics error metrics. The model reviewed is also used in simulation for training and evaluation for our DeepRL method. In §III, the main components of our DeepRL approach for CTRs are described. Improvements such as exploring constrained and constraint-free tube rotation as well as generalizing the policy to accommodate multiple CTR systems are introduced. For tube rotation, we investigate how constraining the rotational degree of freedom for CTRs affects overall error metrics. In other deep learning methods, tube rotations are often constrained during data collection. For example, in [13], a restricted rotation joint space of $-\pi/3$ to $\pi/3$. For a neural network approach, this may not cause issues however, in a timestep-based exploration method like DeepRL, such constraints may hinder exploration of the workspace. Also introduced is a novel end-to-end CTR generalized DeepRL policy, which to our knowledge, is the first work in generalizing over tube parameters with a model-free framework for CTRs. Thus far, in deep learning approaches for CTRs, training occurs with data from a single selected CTR system, in simulation, hardware, or a combination. However, considering that hardware systems often differ from simulation due to manufacturing inconsistencies, and the large design space of CTRs, this type of training

overfits the single CTR system. To demonstrate generalization to multiple CTR systems, the work investigates the scalability and accuracy of generalization over 4 CTR systems of varying workspace sizes. In §IV, results validating these improvements as well as error comparisons to previous tip-tracking and inverse kinematics methods are presented. Finally, we discuss strategies for translation to hardware including domain transfer and initial experiments for a domain transfer strategy known as domain randomization. The contribution evolved from our previous work [11] may be summarised as:

1) Investigating constrained and constraint-free rotation on tube rotation.
2) Development of an initial proof-of-concept CTR system generic policy for CTRs.
3) Details for a pathway and initial simulation results for strategies to hardware translation.

## II. RELATED WORK

Over the last few years, deep learning approaches have become popular for control as well as kinematic and dynamic estimation of CTRs. The first deep learning approach was by Bergeles et al. [14] for forward and inverse kinematics, performed in a simulation where a simple extension and rotation representation was used. However, tube rotation representation in the network resulted in ambiguity in inverse kinematics solutions. In later work by Grassman et al. [13], by improving the joint representation, errors of $2.8\%$ of robot length were achieved, albeit in a limited region of the workspace and in simulation. More recently, work on shape estimation and shape-to-joint input estimation has been investigated [15], [16] using deep neural networks. Finally, Donat et al. [17] introduced tip contact force estimation based on backbone deflection using a data-driven approach via deep direct cascade learning (DDCL). With tip error represented as a percentage of robot backbone length, more traditional optimization-based methods and inverse Jacobian methods have been found to have a tip tracking error of $3.2\%$ and $2.5\%$. State-of-the-art closed-loop control methods can achieve errors of $0.9\%$ and $0.5\%$ [18]. With active constraints and the use of a model-predictive closed loop controller, tip errors of $0.3-0.5\%$ [19] have also been reported.

The Jacobian-based controllers which are common in literature [18], [20] can be used in a closed-form fashion to perform path following with a CTR. A similar comparison was done in [19]. Given a control input or desired change in joint values $\dot{q}_d$, the desired change in Cartesian space $\dot{x}_d$ and desired position in Cartesian space $x_d$, a positive semi-definite matrix $K_p$ and the pseudo-inverse Jacobian $J^\dagger$, we can define a control law such that

$$\dot{q}_d = J^\dagger \left[ \dot{x}_d + K_p \left( x_d - x \right) \right]. \tag{1}$$

Moreover, the pseudo-inverse can become very sensitive to singularities so a damping factor $\Lambda$ is added such that $J^\dagger = (J^T J + \Lambda^2 I)^{-1} J^T$. As shown in [19], this method does not account for joint limits resulting in failed trajectories. Although learning-based approaches have been well developed and have had success for forward kinematics, force estimation and shape

estimation, inverse kinematics and control using deep learning has remained an open problem for CTRs. Given that the deep learning-based forward kinematics [13] and shape estimation [15] report better error metrics than their physics-based model comparisons, investigating a deep learning-based approach for inverse kinematics and control could be beneficial and advantageous. To this end, we have investigated the use of DeepRL for CTRs. Our initial work [21] investigated the exploration problem for CTRs with simpler constant curvature dominant stiffness kinematics for simulation. The exploration problem stems from previous analysis for workspace characterization [22] that has shown the bias associated with uniform joint sampling for CTR workspaces. Due to the constraints to extension from the actuation perspective, full extension and retraction are less likely to be sampled. Since DeepRL methods rely on the experiences collected during training episodes as determined by the agent's actions if full extension or retraction joint values are not sampled, kinematics and control in those areas of the workspace will not be accurate. To mitigate this, noise is usually added to the selected actions or policy network to explore the state space. Our initial work determined that applying separate noise to the rotation and extension joints was crucial in acquiring a policy with accurate control.

As this work builds on our previous DeepRL literature, a brief overview of key concepts is introduced. In our last work [11] we improved our initial DeepRL approach [21] by using a more accurate geometrically exact kinematics model [9] for simulation, investigating joint representations, and applying a reward curriculum to improve sample efficiency (faster policy convergence with fewer data) for training. Two joint representations and three curricula functions were evaluated. The curricula were used to determine the goal tolerance during training steps, a novel approach for DeepRL to our best knowledge. The egocentric representation with the decay curriculum performed best overall in terms of sample efficiency and error metrics. To demonstrate policy robustness, a second noise-induced simulation was created where Gaussian noise was added to the join values as encoder noise and end-effector position as tracking noise. The training was then performed on the noise-induced and noise-free simulation. Performing evaluations on a noise-induced simulation, a slight improvement was seen in policy trained with the noise-induced simulation. The main takeaway from these experimental results was that the policy learned can incorporate some amount of noise in the state, and still perform adequately.

In §III, we first formulate the Markov Decision Process (MDP) consisting of state, action, and rewards. Next in subsection III-D, we review the results of our previous work, specifically joint representations of egocentric and proprioceptive, reward curricula, and combinations of representations and curricula. We then expand to constraint-free rotation in III-E to significantly improve error metrics and finally introduce and evaluate the system identifier to generalize results across multiple CTR systems. Finally, in §IV we provide an initial experiment for domain transfer to hardware using domain randomization in simulation to motivate transfer to hardware.
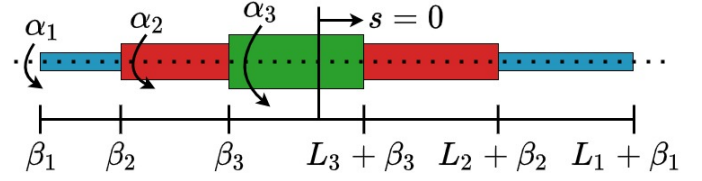


Fig. 2. Extension $\beta_i$ and rotation $\alpha_i$ for tube $i$ of a $i = 3$ 3-tube tube CTR system where $L_i$ is the overall length. $s$ is the arc length or axis along the backbone.

## III. METHODS

In RL, Markov Decision Processes (MDPs) define mathematically the agent's task. Importantly, it defines the key elements for changing the state, the associated rewards, and the actions that affect the state to achieve the task.

### A. Markov Decision Process Formulation

In the following, the state, action, reward, and goals are defined.

State ($s_t$): The state at timestep $t$, is defined as the concatenation of the trigonometric joint representation [13], Euclidean norm between the current desired position and desired position and current goal tolerance. As shown in Fig. 2, rotation and extension of tube $i$ (ordered innermost to outermost) are $\alpha_i$ and $\beta_i$ with $L_i$ representing the full length. First, the trigonometric representation, $\gamma_i$, of tube $i$ is defined as:

$$\gamma_i = \{\gamma_{1,i}, \gamma_{2,i}, \gamma_{3,i}\} = \{\cos(\alpha_i), \sin(\alpha_i), \beta_i\}. \quad (2)$$

The rotation can be retrieved by taking the arc-tangent

$$\alpha_i = \arctan2(\gamma_{2,i}, \gamma_{1,i}). \quad (3)$$

The extension joint $\beta_i$ can be retrieved directly and has constraints

$$0 \geq \beta_3 \geq \beta_2 \geq \beta_1 \quad (4)$$

$$0 \leq L_3 + \beta_3 \leq L_2 + \beta_2 \leq L_1 + \beta_1 \quad (5)$$

from the actuation side. In our previous work, the rotation was constrained from $[-180°, 180°]$, which was not required in the trigonometric representation, as will be shown with constraint-free rotation. The Cartesian goal error is the current error of the achieved end-effector position $G_a$, and desired end-effector position $G_d$. Lastly, the current goal tolerance, $\delta(t)$ is included in the state where $t$ is the current timestep $t$ of training. The full state, $s_t$, can then be defined as:

$$s_t = \{\gamma_1, \gamma_2, \gamma_3, G_a - G_d, \delta(t)\}. \quad (6)$$

Action ($a_t$): Actions are defined as a change in rotation and extension joint positions:

$$a_t = \{\Delta\beta_1, \Delta\beta_2, \Delta\beta_3, \Delta\alpha_1, \Delta\alpha_2, \Delta\alpha_3\}. \quad (7)$$

The maximum actions in extension and rotation are set to 1.0 mm and $5°$.

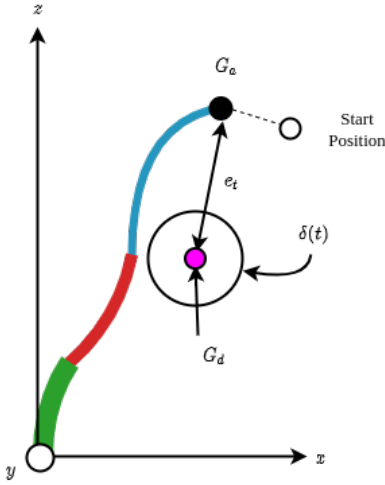Goals ($G_a$, $G_d$): Goals are defined as Cartesian points within

Fig. 3. Illustration of the CTR system with different components of the MDP state definition including the starting position (white), achieved goal ($G_a$) (black), desired goal ($G_d$) (magenta), goal tolerance ( $\delta(t)$). The outer tube (green), middle tube (red), and inner tube (blue).

the workspace of the robot. There is the achieved goal, $G_a$, and desired goal, $G_d$ where the achieved goal is determined with the forward kinematics of the kinematics model used and is recomputed at each timestep as the joint configuration changes from the selected actions from the policy. The desired goal updates at the start of every episode where a desired goal is found by sampling valid joint configurations in the workspace and applying forward kinematics of the model. However, this is not uniform in Cartesian space and requires action exploration.

Rewards ($r_t$): The reward is a scalar value returned by the environment as feedback for the chosen action by the agent at the current timestep. In this work, sparse rewards are used as they have been shown to be more effective than dense rewards when using hindsight experience replay (HER) [23]. The reward function used in this work is defined as:

$$r_t = \begin{cases} 0 & \text{if } e_t \leq \delta(t) \\ -1 & \text{otherwise} \end{cases} \quad (8)$$

where $e_t$ is the Euclidean distance $||G_a - G_d||$ at timestep $t$ and $\delta(t)$ is the goal-based curriculum function that determines the goal tolerance at training timestep $t$. The workspace and various state and reward elements are illustrated in Fig. 3.

### B. Goal-Based Curriculum

In our previous work, we introduced a novel goal-based curriculum that reduces goal tolerance through training steps to improve error convergence and overall error convergence. Linear and exponential decay goal-based curriculum along with a baseline constant curriculum function were compared with combinations of proprioceptive and egocentric joint representations. Each curriculum reduces the goal tolerance as a function of timestep $t$, the number of timesteps to apply the

function, $N_{ts}$, the initial tolerance, $\delta_{initial}$ and final tolerance, $\delta_{final}$. The linear curriculum is defined as

$$\delta_{linear}(t) = at + b$$
$$a = \frac{\delta_{final} - \delta_{initial}}{N_{ts}} \quad (9)$$
$$b = \delta_{initial},$$

and the exponential decay curriculum is defined as

$$\delta_{decay}(t) = a(1 - r)^t$$
$$a = \delta_{initial} \quad (10)$$
$$r = 1 - \left(\frac{\delta_{final}}{\delta_{initial}}\right)^{\frac{1}{N_{ts}}}.$$

The values used for these various parameters can be found in [11].

### C. Joint Representation

To improve learning sample efficiency, joint representations were investigated. Specifically, proprioceptive (absolute) and egocentric (relative) joint representations where the reference of measure for each joint position is changed respectively. Although proprioceptive representations are used often in classical control for robotics, egocentric joint representations are utilized heavily in reinforcement learning control simulation environments like the DeepMind control suite [24]. In proprioceptive or absolute joint representation all the joints are referenced from a common base reference. This is illustrated for rotations in Fig. 4a. However, in egocentric or relative joint representations, only the inner tube is referenced from the base shown in Fig. 4b. The next outer tube is referenced from the previous inner tube and so forth. This can be used for both rotation and extension joints.

$$\alpha_{ego} = \{\alpha_1, \alpha_2 - \alpha_1, \alpha_3 - \alpha_2\}$$
$$= \{\alpha_1, \Delta\alpha_{2-1}, \Delta\alpha_{3-2}\} \quad (11)$$

and extensions

$$\beta_{ego} = \{\beta_1, \beta_2 - \beta_1, \beta_3 - \beta_2\} \quad (12)$$

To retrieve the absolute joint representation, the cumulative sum is taken as shown below:

$$\alpha_{prop} = \{\alpha_1, \Delta\alpha_{2-1} + \alpha_1, \Delta\alpha_{3-2} + \Delta\alpha_{2-1} + \alpha_1\} \quad (13)$$

### D. CTR Simulation Environment

To collect data and experiences for the deepRL algorithm to learn a policy, a simulation environment for kinematics was developed following the openAI gym framework [25]. With this gym framework, any compatible DeepRL algorithm can be utilized to learn a policy for the given task. The environment takes a set of tube parameters describing a CTR system, joint configuration, and selected actions by the agent to determine the overall shape of the CTR. For DeepRL, a large number of experiences are needed to train a policy, so a "sweet-spot" or relatively fast computationally and relatively accurate kinematics model was used which was first presented in [26] and later presented for externally loaded CTRs with
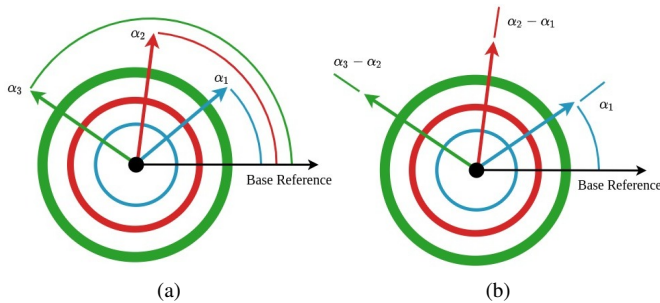
Fig. 4. Illustration of rotational (a) proprioceptive and (b) egocentric joint representation for rotational joints. Inner (blue), middle (red), and outer (green) tubes' rotation representation with respect to the base (black).

**TABLE I**
**JOINT REPRESENTATION WITH CURRICULUM RESULTS.**

| Joint Representation | Curriculum | Error (mm) | Success rate |
|---|---|---|---|
| **Proprioceptive** | Constant | $2.43 \pm 0.09$ | 0.85 |
| | Linear | $3.31 \pm 0.15$ | 0.87 |
| | Decay | $3.16 \pm 0.12$ | 0.80 |
| **Egocentric** | Constant | $1.94 \pm 0.05$ | 0.87 |
| | Linear | $3.38 \pm 0.15$ | 0.89 |
| | Decay | $1.29 \pm 0.03$ | 0.93 |

point and distributed forces and moments in [1], [12]. This model ignores friction, permanent strain, and forces along the backbone of the robot. The gym framework is then used for experiments including inverse kinematics, path following, and domain randomization.

To summarize the results of our previous work, investigating combinations of reward curriculum and joint representations, the egocentric, decay curriculum was found to perform best in evaluation. Training was done with deep deterministic policy gradient (DDPG) [27] algorithm with hindsight experience replay (HER) [23]. Evaluation was done with 1000 evaluation episodes with results shown in Table I. The success rate is defined as the number of successful trajectories over the total number of trajectories, where success is achieving a trajectory with below 1 mm error. In the simulation environment for these experiments, tube rotations were constrained from $-\pi$ to $\pi$. Unconstraining the tube rotation will be investigated in the next section. With learned policy, since the desired goal is within the state as shown in equation (6), a policy controller was created where the desired goal was updated via a trajectory generator. This type of controller is normally not available for other deep neural network approaches as they are not inherently timestep based whereas DeepRL aims to optimize actions at each timestep. Following various paths including circular and straight-line, the mean tracking error was $0.58$ mm using the egocentric decay curriculum. To demonstrate robustness, tracking and joint noise of $0.8$ mm and $1°$ was induced into the simulation resulting in $1.37$ mm mean tracking error. However, with the tube rotation constraints, exploration may not be exploring the workspace in its entirety. Another factor is workspace size which depends on the CTR system used for training. It is evident larger workspaces will take longer computation time to train. First, performing a workspace and joint error analysis, we found removing constraints on tube rotations provided a significant performance improvement, particularly in larger CTR systems. Second, we found generalization possible with the MDP formulation by appending a system identifier to the state. This generic policy would be useful as only a single policy would be needed for multiple systems and would be the first step towards full generalization for deep learning-based CTR kinematics. Moreover, this generalization motivates experiments using domain randomization, for domain transfer from simulation

to hardware. In the following, we describe the methods for the two main experiments, improvement through constraint-free rotation and generalization.

### E. Improvements with constraint-free tube rotation

With the best policy training method, egocentric decay from the previous work, state information such as the achieved goal, desired goal, Cartesian error, and joint error at the end of each episode from 1000 evaluation episodes were tabulated after training the policy on the larger CTR system 0. Plotting the Cartesian points of the achieved goal with RGB values corresponding to Cartesian error to the desired goal results in Fig. 6a. Furthermore, thresholding achieved goal points with Cartesian errors to the desired goal greater than 2 mm, and regions of larger errors can be isolated. As seen, there is a large standard deviation in errors greater than 2 mm with constrained rotation. This motivates unconstraining tube rotations during DeepRL training for exploration. Shown in Figure 6b are constraint-free rotation training results in no errors greater than 2 mm, thus the standard deviation of errors is greatly reduced.

In order to investigate the joint values associated with these large errors, Figure 6 shows that the Cartesian achieved goals for the innermost tube or $\alpha_1$ at the end of each episode and the associated errors in the robot workspace. As seen, there is a large number of errors greater than 2 mm with some points up to 30 mm in error. In Figs. 7a, 7b and 7c, the constraints causing the large errors at the boundaries of $-180$ and $+180$ where the largest outliers for errors are confirmed with a polar plot for each tube rotation. In previous work, this rotation constraint is to limit the joint space sampling during the generation of new desired goals, starting joint values and data collection as has been implemented in previous CTR deep learning work [13]. However, this constraint through timesteps is non-essential in the trigonometric representation. Training a policy using the egocentric decay curriculum without constraining the rotations of the tubes considerably improved the error metrics from the previously constrained egocentric curriculum from a mean error of $4.05$ mm to $0.68$ mm for the largest system 0.

To further analyze the behavior of the agent with respect to error, a goal distance to Cartesian error analysis was performed from the 1000 evaluation episodes, and state information was tabulated. This analysis reveals the relationship between distance to the desired goal and the associated final error.
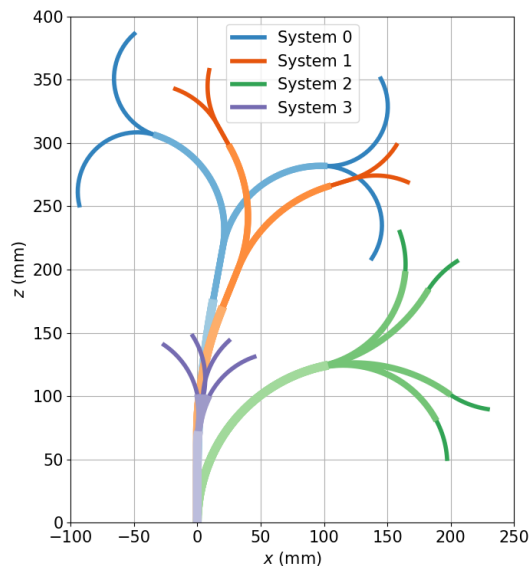
Fig. 5. Visualizing the different CTR systems used with the geometrically exact modeling. Chosen were 4 different CTR systems, each composed of 3 tubes of various parameters resulting in different workspaces. The longest system is CTR system 0 (blue), with the shortest CTR system being system 3 (purple). System 2 (green) has the largest curvature resulting in a wide workspace. Finally, the CTR system 1 is in orange. The systems are ordered from longest to shortest. Each system in the figure is fully extended with the middle and inner tube rotated at $0°$ and $180°$ for a total of 4 configurations per system.

TABLE II
CONSTRAINED AND UNCONSTRAINED ROTATION EXPERIMENTS FOR
VARIOUS CTR SYSTEMS.

| Experiment | System | Error (mm) | Success rate |
|---|---|---|---|
| **Constrained** | 0 | $4.05 \pm 18.18$ | 77.1 |
| | 1 | $2.53 \pm 12.40$ | 92.5 |
| | 2 | $3.13 \pm 16.08$ | 92.8 |
| | 3 | $0.52 \pm 1.49$ | 99.6 |
| **Constraint free** | 0 | $0.68 \pm 0.28$ | 98.3 |
| | 1 | $0.64 \pm 0.67$ | 99.3 |
| | 2 | $0.61 \pm 0.24$ | 99.0 |
| | 3 | $0.45 \pm 0.20$ | 99.9 |

Generally, a closer goal, ie. a low initial goal distance, would be expected to have smaller errors overall, with farther goals having larger errors. For this analysis, all 1000 data points were used to determine the linear relationship between the initial goal distance and final Cartesian error. For constrained rotation with system 0, the slope was found to be 3.27 mm with a $y$-intercept of 0.8 mm. This suggests a poor inverse kinematics solver as errors become very large with higher goal distances. With constraint-free rotation, the slope was found to be 0.6 mm with a $y$-intercept of 0.66 mm, a reasonable slope for such a relationship.

As an inverse kinematics solver, our DeepRL method performs adequately when rotations are unconstrained. However, an important note is only one solution is provided as it is an iterative solver and is dependent on the initial

condition of the joint configuration at the start of each episode. However, this can be remedied with multiple episodes with different initial joint configurations. Visualized in Fig. 8 is an example of the same desired end-effector position with two different initial joint configurations, resulting in two different final inverse kinematics solutions. Using the constraint-free egocentric decay method for system 0, the desired goal position was $(0, 50, 150)$ mm. The final joint configuration in Fig. 8a was $\beta = [-15.35, -12.59, -4.01]$ mm, $\alpha = [-3.73, -183.07, -25.54]°$ with a tip error of 0.99 mm and for Figure 8b $\beta = [-14.77, -9.12, -5.12]$ mm and $\alpha = [-179.19, -180.32, 1.93]°$ with a tip error of 0.36 mm.

To verify the constraint-free results and to demonstrate training and evaluation of our DeepRL method, we apply it to three other CTR systems from various sources in the literature and trained each with constrained and constraint-free rotation, and performed 1000 evaluation episodes. The main changes to hyperparameters were 3 million training timesteps with 1.5 million steps for the curriculum, a buffer size of $500,000$, and neural network layer size of 3 hidden units with 256 neurons each. The results for all four CTR systems are presented in Table II. Of note is the increased standard deviation in constrained rotation. However, with smaller systems such as system 3, this is not as pronounced due to the smaller robot workspace. From our previous work, there has been a large reduction in mean and standard deviation of errors as shown in the largest CTR system.

### F. Generic Policy

To further motivate the utility of DeepRL for CTRs, we introduce an initial proof of concept for a CTR system generic policy. A hurdle currently with using deep learning approaches for CTRs is the limitation of CTR system generalization. Because deep learning relies solely on the data collected, if only one CTR system is used for training, the learned policy will accurately control that system alone. Moreover, we aim to demonstrate that using our egocentric decay goal-based curriculum, with constraint-free rotation has improved error metrics when compared with the one that does not employ these extensions for such a generic policy.

This CTR generic policy will seek to generalize over four CTR systems that have different tube parameters. The objective is to obtain good performance across the CTR systems with a single control policy. For generalization, a system specifier, $\psi = \{0, 1, 2, 3\}$, was appended to the state, $s_t$, for the agent to differentiate the CTR systems. The state is now defined as

$$s_t = \{\gamma_1, \gamma_2, \gamma_3, G_a - G_d, \delta(t), \psi\} \quad (14)$$

At the start of each episode, a discrete uniform distribution is sampled to determine the value of $\psi$, or the CTR system parameters. These parameters are then set in the simulation environment and for that episode, the selected system is the one simulated for the agent's task. Once the episode is reset, a new $\psi$ is sampled. Thus, over timesteps, all systems should be sampled with the agent collecting experiences from all systems uniformly. We acknowledge this is not a true generalization,
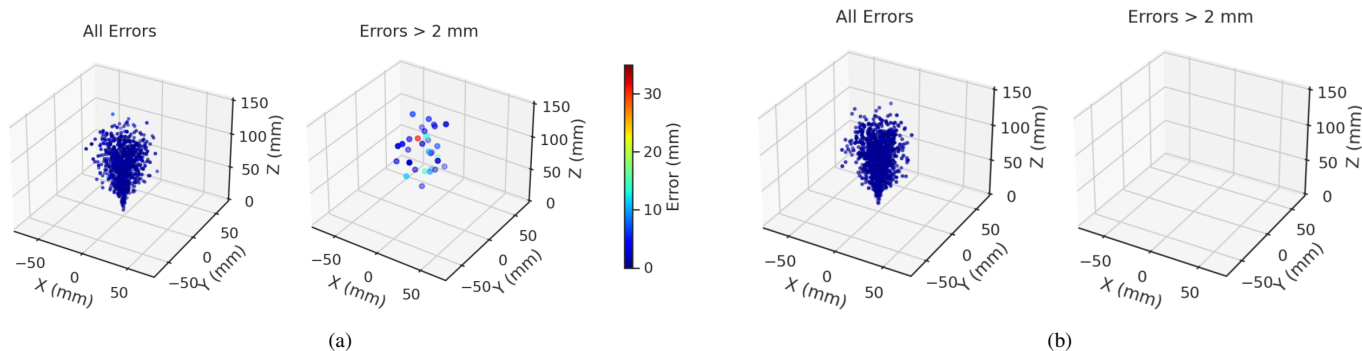
(a)　　　　　　　　　　　　　　　　　　　　　　　　　　(b)

Fig. 6. Plotting achieved goal positions of evaluations of a policy trained in a simulation environment with constrained tube rotation (a) and (b) constraint-free tube rotations. Furthermore, each plot is filtered to errors greater than 2 mm to the desired goal of the episode to illustrate large errors. In (b), no errors are greater than 2 mm demonstrating a low error variance.

TABLE III
ERROR METRICS FOR SYSTEM GENERALIZATION FOR THE VARIOUS NUMBER OF SYSTEMS AND COMBINATION OF SYSTEMS. ERRORS PRESENTED IN MILLIMETERS MEAN $\pm$ STANDARD DEVIATION.

| Number of Systems | Systems | First System | Second System | Third System | Fourth System |
|---|---|---|---|---|---|
| **Two Systems** | [0,1] | $0.715 \pm 0.057$ | $0.710 \pm 0.054$ | – | – |
| | [0,2] | $0.663 \pm 0.069$ | $0.650 \pm 0.063$ | – | – |
| | [0,3] | $0.660 \pm 0.056$ | $0.593 \pm 0.052$ | – | – |
| | [1,2] | $0.606 \pm 0.056$ | $0.623 \pm 0.064$ | – | – |
| | [2,3] | $0.655 \pm 0.057$ | $0.602 \pm 0.046$ | – | – |
| | [1,3] | $0.636 \pm 0.0533$ | $0.582 \pm 0.052$ | – | – |
| **Three Systems** | [0,1,2] | $0.726 \pm 0.092$ | $0.690 \pm 0.057$ | $0.749 \pm 0.087$ | – |
| | [0,1,3] | $0.826 \pm 0.102$ | $0.764 \pm 0.081$ | $0.713 \pm 0.056$ | – |
| | [0,2,3] | $0.901 \pm 4.170$ | $0.964 \pm 0.217$ | $0.791 \pm 0.053$ | – |
| | [1,2,3] | $0.675 \pm 0.059$ | $0.647 \pm 0.074$ | $0.636 \pm 0.071$ | – |
| **Four Systems** | [0,1,2,3] | $0.754 \pm 0.083$ | $0.723 \pm 0.061$ | $0.727 \pm 0.069$ | $0.629 \pm 0.052$ |

and the policy only learns the systems given, however, this initial proof-of-concept demonstrates some form of generalization is possible for CTRs using DeepRL, an attribute not shown in any previous work. Given the right network parameters, all attributes defining a CTR system could be included in the state to generalize fully. To demonstrate initial generalization, we train a single policy to generalize over two, three, and four systems including different combinations. The systems are ordered 0 to 3, from the shortest overall system to the longest overall system. First, to validate our egocentric decay constraint-free method, we compared generalization with our constraint-free egocentric decay method to the constraint-free proprioceptive constant method and constrained proprioceptive constant policy. We aim to demonstrate that our constraint-free egocentric curriculum is key to policy convergence for generalization. A full set of results for all combinations of generalization are provided for the different systems. Then, to demonstrate the learned policy we present path following task results generalized over four systems, with and without sensor and encoder noise to display robustness.

We make our gym environment and the code to reproduce

these results available online. [1]

## IV. EXPERIMENTS AND RESULTS

Thus far, we have verified that constrained tube rotation causes large errors at the boundary of the constraints. This is due to constraints on rotation resulting in inadequate exploration. Furthermore, we verified the method as an inverse kinematics solver by solving for a desired goal position with two different initial starting positions. The result was two different inverse kinematics solutions given the two different initial starting positions. As mentioned prior, the deepRL method can also perform path following. To validate this, experiments in simulation were performed following various paths with the constraint-free egocentric decay DeepRL method. Next, we investigate the state identifier for generalization, and present error metrics and path following results.

In a surgical scenario, the surgeon would be controlling the end-effector tip position with a haptic device similar to a Phantom Omni from Sensable Technologies. Thus, the inputs are Cartesian coordinates of the end effector tip position and fit well with the state description of our DeepRL method.

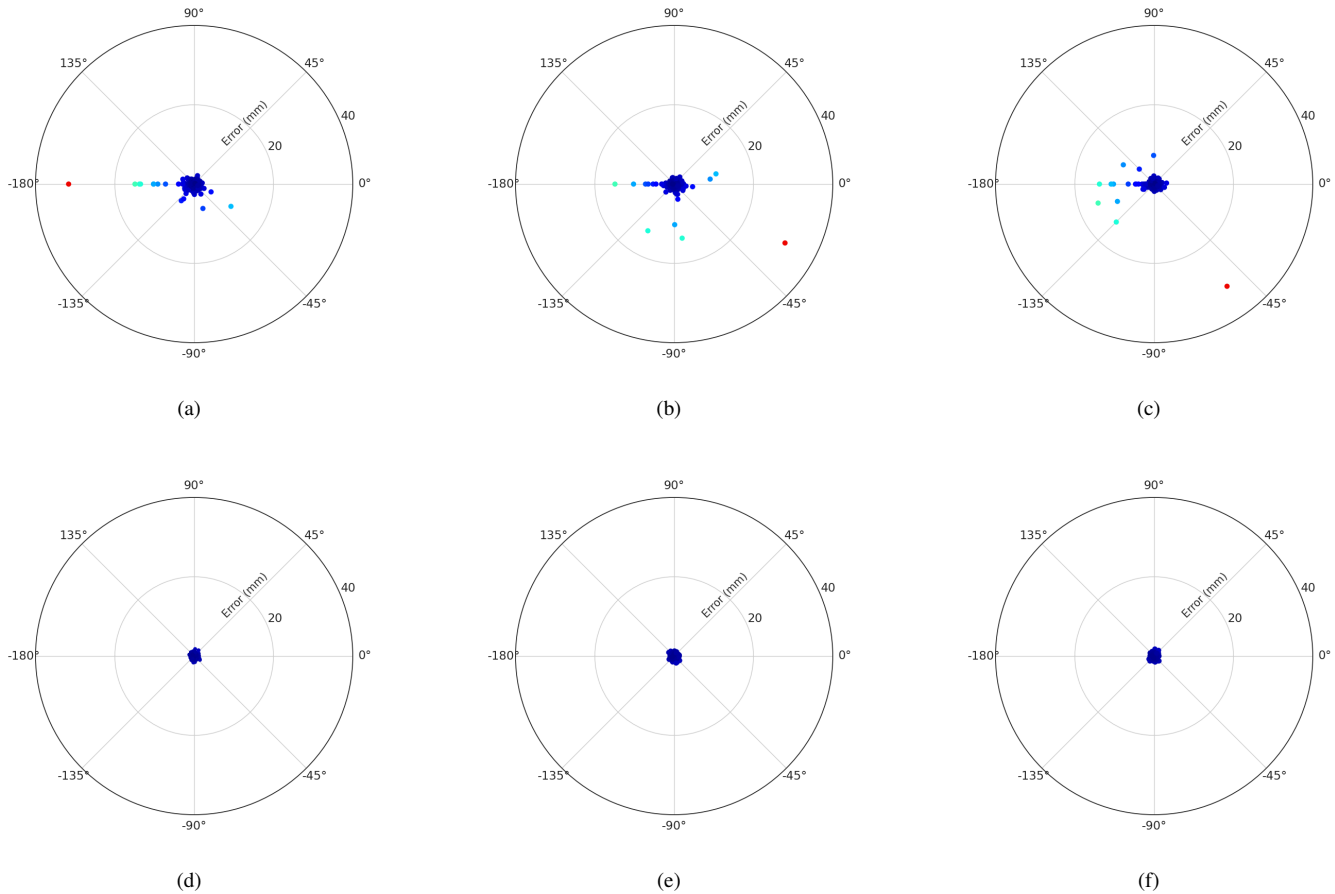[1][Online]. Available: https://github.com/keshaviyengar/gym-ctr-reach

Fig. 7. From evaluations, plotting tube rotations and associated errors in RGB. Constrained rotation errors with respect to tube rotation angle for (a) $\alpha_1$, (b) $\alpha_2$ and (c) $\alpha_3$. Constraint-free rotation errors with respect to tube rotation angle for (d) $\alpha_1$, (e) $\alpha_2$ and (f) $\alpha_3$. See Fig. 6 for RGB color legend.
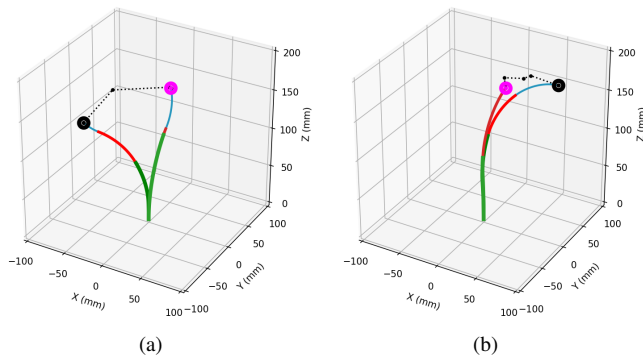


Fig. 8. Two inverse kinematics solutions with different starting joint configurations given the same desired goal. The black dot indicates starting position and the magenta dot indicates the desired position with the smaller black dots indicating intermediary reached points.

Therefore, there is the single-point inverse kinematics task as seen previously, and the path-following tasks described here. The experimental framework for path-following tasks is as follows. First, $x, y, z$ desired goal positions are generated by a path generator component to substitute control inputs from a user. This component takes as input path shape parameters and the discretization parameter. Shapes include polygons, circular

and helix paths. The generator outputs a series of $x, y, z$ desired goal positions of the path with the path parameters given. The next component is the policy controller which takes two inputs, the desired goal positions and the initial joint configuration. The controller is described in detail in the next section. The controller outputs the changes in joint positions to achieve the path by reaching each of the desired goals given by the path generator. The controller is open-loop since information about whether the goal is reached is not relayed back. There are a number of steps given to reach the goal, and even if the goal is not reached, the next goal is given. Finally, the last component is the CTR simulation. The simulation takes an input of changes in joint positions, performs kinematics for each step, and returns the achieved goal positions of the end-effector as well as the full CTR backbone shape, with which the resulting path followed can be visualized. The entirety of the validation path following framework can be found in Fig. 9.

### A. Policy Controller

The policy controller component acts as an open loop controller that takes as input a series of desired goal positions. The main control occurs while iterating over the given desired goal positions. Iterating through desired goals, first, the
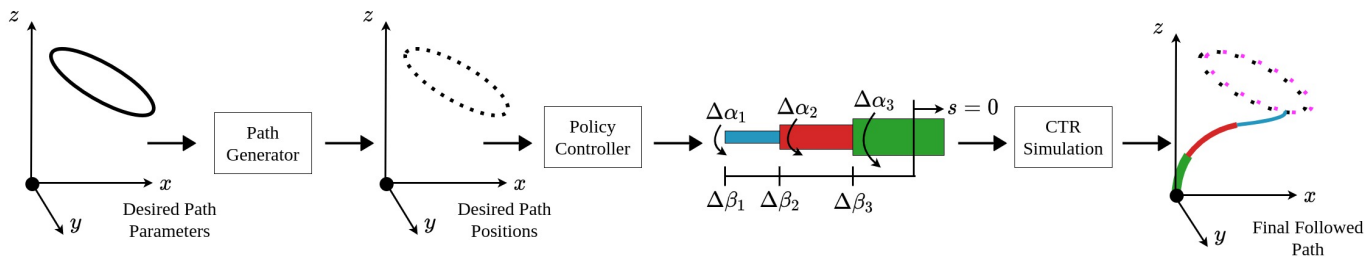
Fig. 9. Illustration of the process by which paths are generated, control actions are determined, and finally, paths followed.
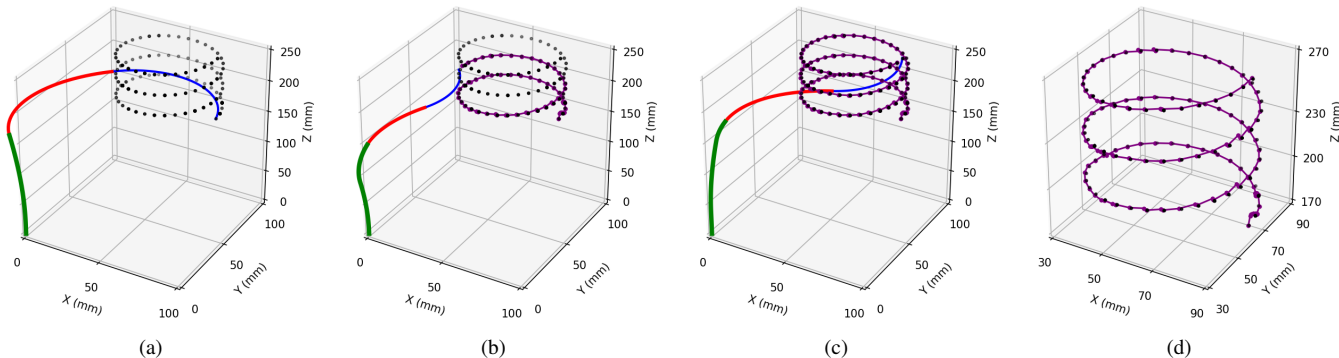


Fig. 10. CTR system 0 following a helix path. (a) starting point of path, (b) midway through path, (c) final point of path and (d) full path following results.
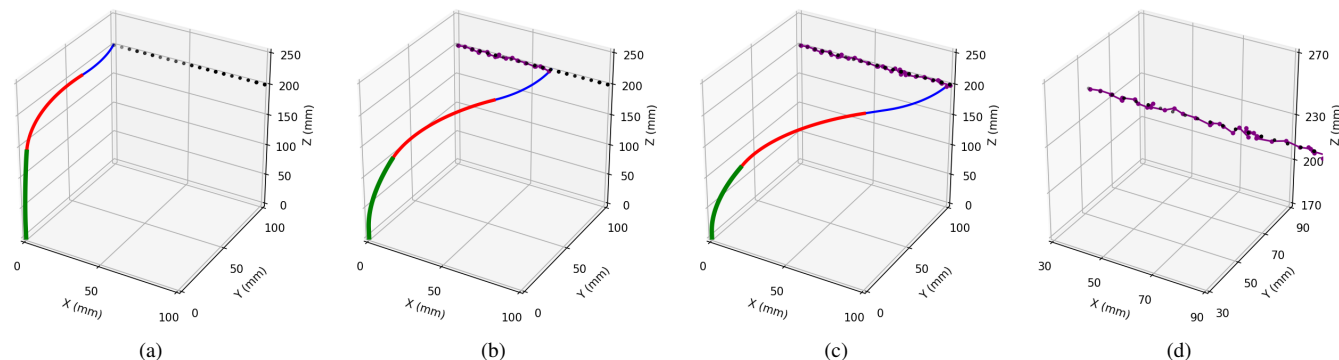


Fig. 11. System 1 following a straight path. (a) starting point of path, (b) midway through path, (c) final point of path and (d) full path following results.

environment is initialized with a reset function that returns the first state. The reset function is used as an input to the system identifier and desired goal. Next, in the main for loop of the controller, the policy is given 20 timesteps to achieve the current desired goal with actions from the policy function. If the agent has achieved the current desired goal within the 1 mm tolerance before 20 timesteps, then a break is initiated and the next desired goal is set. This is outlined in Algorithm 1.

### B. Single System Validation

To validate our constraint-free, egocentric decay curriculum DeepRL method, we train different CTR systems and present error metrics for inverse kinematics and path following for each system. This was done to verify that the same method can

be applied to various systems, resulting in an accurate learned policy. Additionally, we add state noise in the form of encoder and end-effector position noise. This is to demonstrate the learned policy is somewhat resilient to noise in the state. For simplicity, a $1°$ standard deviation was selected. To determine the extension joint noise, a gear ratio of $0.001$ was used. For achieved goal tracking noise, a standard deviation $0.8$ mm based on an EM tracker (Aurora, NDI Inc., CA) precision data found in the documentation.

Secondly, to provide a pathway to hardware translation, we include initial results for domain randomization. In domain randomization, to transfer the policy from the source domain (simulation) to the target domain (hardware), a series of environmental parameters in the source domain is sampled from a randomized space. During training, the episodes are collected with the source domain with randomization sampling

---

**Algorithm 1** Policy Controller

---

**Input:** $G_d^0 \ldots G_d^N$
**Output:** $A[\,]$ (Array of actions to achieve path)

1: **function** CONTROLLER($G_d[\,]$)
2:     **for** $i \leftarrow 1$ to $N$ **do**
3:         $G_{current} \leftarrow G_d[i]$
4:         $s_{current} \leftarrow Reset(G_d)$
5:         **for** $t \leftarrow 0$ to $20$ **do**
6:             $a_{current} \leftarrow Policy(s_{current})$
7:             $s_{next}, r_{next}, terminal \leftarrow Step(a_{current})$
8:             $A \leftarrow A.append(a_{current})$
9:             **if** $terminal = True$ **then** $break$
10:            **end if**
11:         **end for**
12:     **end for**
13:     **return** $A[\,]$
14: **end function**

---

applied. This allows for the policy to be exposed to a variety of environmental parameters to generalize. In this way, the policy is trained to maximize the expected reward over a distribution of CTR configurations around the desired CTR configuration. Specifically, uniform domain randomization [28], is implemented for the tube parameters specified in the simulation. These tube parameters include length, curved length, inner diameter, outer diameter, stiffness, torsional stiffness, and precurvature. In uniform domain randomization, an interval range from which the tube parameters are uniformly sampled should be defined. For example, for a randomization value of 5 %, the inner diameter sampling of system 0, the lower range would be $0.7 - 0.05 \times 0.7 = 0.665$ and the upper range would be $0.7 + 0.05 \times 0.7 = 0.735$. The chosen CTR configuration for the domain randomization experiment was that of system 2 with a domain randomization value of $\pm 5\%$ of each parameter.

To evaluate this translated domain policy, we performed path-following tasks and inverse kinematics on system 2 tube parameters. Although there exist more sophisticated methods of domain translation, as an initial work, we believe this demonstrates the feasibility to translate the policies trained to hardware. In 1000 evaluation episodes, a mean error of $0.86$ mm was found with a standard deviation of $0.64$ mm. Using a straight-line path for testing for path following, the mean tracking error was $1.10$ mm with a standard deviation of $0.15$ mm. We compare these results to state-of-the-art in the next section. Without domain randomization, training results for inverse kinematics were summarized in Table II under constraint-free experiments. For the path following, in a noise-free simulation, system 0 had error metrics of $0.66 \pm 0.28$ mm for a helix path and $1.74 \pm 0.72$ mm in a noise-induced simulation. System 0 is the longest system, with the highest errors, and was chosen for results, evaluation, and comparisons.

### C. Generic Policy Validation

To validate our generic policy method, we trained a generic policy for different combinations of two, three, and four CTR

systems. For example, to generalize over two CTR systems, because we have four different CTR systems available to train on, we trained on all combinations of two systems resulting in a total of six experiments. Similarly, generalizing over three systems results in four experiments and a single experiment to generalize over all four CTR systems. Performing 1000 evaluation episodes and summarizing the error metrics, the proposed method is able to generalize over multiple systems. The full error metrics are shown in Table III. Looking at the error metrics with respect to systems, there is a correlation between the length of the system and higher errors similar to the previous constrained and constraint-free experiments. System 0 is the longest length, and consistently has the largest error metrics. We believe this is because of the workspace size, as overall length increases, the agent will require more training steps and experiences. Another factor is that in general, longer CTR systems have larger errors, and thus comparisons are done with the percentage of robot length. To mitigate this effect, a sampling strategy is used where the sampling of the system used in the environment is based on the lengths of the systems. In this length-based sampling strategy, the categorical distribution is proportionate to the length of each system. Each system probability is the system length divided by the sum of the system lengths being generalized. In this manner, systems that are longer and that have larger workspaces are sampled more during training and have more experiences for the policy to train. Evaluating this sampling strategy for generalizing over four CTR systems, the error metrics were improved from the previous uniform sampling. To validate the generalization, we performed a helix path following task with system 0 as seen in Fig. 10. The error metrics were a mean tracking error of $1.01$ mm and a standard deviation of $0.41$ mm with $50$ desired goal points in the path. Performing a noise-induced experiment the error metrics were a mean tracking error of $1.86$ mm with a standard deviation of $0.8$ mm. When the number of points was increased to $100$, error metrics were a mean tracking error of $0.91$ mm with a standard deviation of $0.41$ mm. In a noise-induced simulation, the mean tracking error was $1.91$ mm with a standard deviation of $0.78$ mm.

To compare our generic egocentric constraint-free method, we also train using proprioceptive representation for a four-system generic policy with constrained rotation and one with constraint-free rotation to compare to our egocentric constraint-free policy. We summarize the results in Table V. To note is the importance of removing rotation constraints as seen in the mean and standard deviation of errors from constrained proprioceptive to constraint-free proprioceptive. Error metrics are greatly reduced with this improvement. Finally, using an egocentric representation does improve metrics but has less of a significant impact as compared to rotational constraints.

### D. Comparisons to the State of the Art

To compare our inverse kinematics and path following results to a previous state-of-the-art classical methods and deep learning methods, we convert errors to a percentage of robot length. First, we present inverse kinematics results for our constraint-free egocentric decay for each system when

TABLE IV
FOUR CTR SYSTEMS WITH VARIED PHYSICAL PARAMETERS. THIS TABLE INCLUDES THE LENGTHS AND CURVATURES.

| System | Tube | Length (mm) | Curved Length (mm) | Inner Diameter (mm) | Outer Diameter (mm) | Youngs Modulus (GPa) | Shear Modulus (GPa) | Precurvature $(m^{-1})$ |
|---|---|---|---|---|---|---|---|---|
| System 0 [19] | Inner | 431 | 103 | 0.7 | 1.1 | 102.5 | 187.9 | 21.3 |
| | Middle | 332 | 113 | 1.4 | 1.8 | 685 | 115.3 | 13.1 |
| | Outer | 174 | 134 | 2.0 | 2.4 | 169.6 | 142.5 | 3.5 |
| System 1 [13] | Inner | 370 | 45 | 0.3 | 0.4 | 500 | 230 | 15.8 |
| | Middle | 305 | 100 | 0.7 | 0.9 | 500 | 230 | 9.27 |
| | Outer | 170 | 100 | 1.2 | 1.5 | 500 | 230 | 4.37 |
| System 2 | Inner | 309 | 145 | 0.7 | 1.1 | 75 | 25 | 1.68 |
| | Middle | 275 | 114 | 1.4 | 1.8 | 75 | 25 | 11.6 |
| | Outer | 173 | 173 | 1.83 | 2.39 | 75 | 25 | 10.8 |
| System 3 [21] | Inner | 150 | 100 | 1.0 | 2.4 | 50 | 23 | 15.82 |
| | Middle | 100 | 21.6 | 3.0 | 3.8 | 50 | 23 | 11.8 |
| | Outer | 70 | 8.8 | 4.4 | 5.4 | 50 | 23 | 20.04 |

TABLE V
EXPERIMENTAL COMPARISONS OF ROTATIONAL AND JOINT
REPRESENTATION PER SYSTEM.

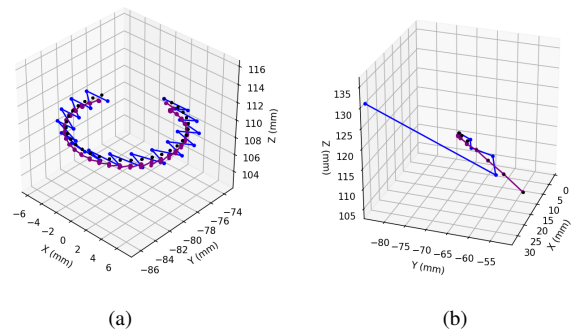| Experiment | First System | Second System | Third System | Fourth System |
|---|---|---|---|---|
| Constrained Proprio-ceptive | 4.68± 15.96 | 1.98± 8.45 | 4.91± 24.05 | 0.87± 1.56 |
| Constraint-free Proprio-ceptive | 0.77± 0.37 | 0.80± 2.63 | 0.78± 0.34 | 0.73± 0.31 |
| Constraint-free Egocentric | 0.75± 0.29 | 0.72± 0.26 | 0.73± 0.26 | 0.63± 0.23 |



Fig. 12. System 0 following a circular path (a) and linear path (b) comparing Jacobian (blue) methods to our Deep RL method (purple) for path following.

trained separately ie. not the generic policy. Mean and standard deviation as percentage for each system was $0.16\% \pm 0.065\%$, $0.17\% \pm 0.18\%$, $0.20\% \pm 0.08\%$ and $0.3\% \pm 0.13\%$. When taken as a percentage of the robot length, the similarity of the error metrics is noteworthy. In our generalization method, for the four system generalization inverse kinematics for each system is $0.18\% \pm 0.02\%$, $0.19\% \pm 0.02\%$, $0.24\% \pm 0.02\%$ and $0.42\% \pm 0.04\%$. For tip tracking with the more complex helix path errors were $0.23\% \pm 0.095\%$ for system 0 with 50 goal points. Increasing the number of desired goal points to 100, the following mean tracking errors of $0.20\% \pm 0.08\%$ as seen in Fig. 10. As reported in §II, Jacobian-based methods can achieve errors of $0.5\%$ to $0.9\%$. As shown in Fig. 12, our DeepRL method is able to avoid joint limits, especially in extension, whereas the Jacobian approach does not include joint limits in the linearization. This comparison was done on system 0 with $K_p = 2I$ for a linear and circular path. The error metrics found were 1.15 mm $\pm$ 0.32 for the Jacobian method and 0.62 mm $\pm$ 0.07 for our DeepRL method in Fig 12a. More importantly, the Jacobian-based method was unable to follow some circular, linear, and helical trajectories that our DeepRL method successfully completed due to the Jacobian not including joint limits in the formulation, even if the damped-least squares method was used with $\Lambda = 0.45$. as seen in Fig. 12b. The advanced MPC method can achieve tip errors of $0.3\%$ to $0.5\%$, however, we were unable to perform

comparisons in simulation as the open-source simulation code was for a two-tube system. Our method does perform comparably to the reported state-of-the-art in simulation, however, this work is only in simulation and does not include constraints for snapping conditions in the model used. The approach will need to be validated in hardware. One possible way to consider snapping and singularity is to design a dense reward function that includes the minimization of elastic energy, hence avoiding snapping conditions. For our domain randomization results, as a percentage of robot length, the inverse kinematics errors were $0.86\% \pm 0.21\%$. Following a straight-line path, the errors were $0.36\% \pm 0.05\%$. The domain randomization metrics are higher, however, the aim is to demonstrate the feasibility of the transfer method. To validate, we will need to compare transfer to hardware with and without domain randomization or other transfer methods.

## V. CONCLUSION

In this work, we investigate DeepRL, an end-to-end method for kinematic control of CTRs. Specifically, we aim to explore constraints on tube rotation and the impact on error metrics. Furthermore, the first proof-of-concept in system generalization is developed, not yet done for any deep learning method

for CTRs. Finally, to provide initial work for hardware, we provided an initial pathway for domain transfer or policy transfer from simulation to hardware using Sim2Real domain randomization. Our method does demonstrate error metrics that perform well, however, validation in hardware is needed. Moreover, other domain transfer methods should be explored. Unaddressed in this work is the issue of snapping, whereby torsion in the tubes causes a rapid transition to a lower energy state resulting in a snapping motion. By creating simulation frameworks that utilize Dynamics models [29], snapping configurations could be avoided through rewards. We believe, with this work, we have demonstrated that deepRL methods may be able to outperform model-based methods for inverse kinematics and control for CTRs, similar to deep learning methods for forward kinematics and shape estimation.

## REFERENCES

[1] P. Dupont, A. Gosline, N. Vasilyev, J. Lock, E. Butler, C. Folk, A. Cohen, R. Chen, G. Schmitz, H. Ren *et al.*, "Concentric tube robots for minimally invasive surgery," in *hamlyn symposium on medical robotics*, vol. 7. Citeseer, 2012, p. 8.

[2] E. De Momi and A. Segato, "Autonomous robotic surgery makes light work of anastomosis," *Science Robotics*, vol. 7, no. 62, p. eabn6522, 2022.

[3] C. D'Ettorre, A. Stilli, G. Dwyer, M. Tran, and D. Stoyanov, "Autonomous pick-and-place using the dvrk," *International Journal of Computer Assisted Radiology and Surgery*, vol. 16, no. 7, pp. 1141–1149, 2021.

[4] 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, 2009.

[5] H. B. Gilbert, J. Neimat, and R. J. Webster, "Concentric tube robots as steerable needles: Achieving follow-the-leader deployment," *IEEE Transactions on Robotics*, vol. 31, no. 2, pp. 246–258, 2015.

[6] H. Alfalahi, F. Renda, and C. Stefanini, "Concentric tube robots for minimally invasive surgery: Current applications and future opportunities," *IEEE Transactions on Medical Robotics and Bionics*, vol. 2, no. 3, pp. 410–424, 2020.

[7] C. Bedell, J. Lock, A. Gosline, and P. E. Dupont, "Design optimization of concentric tube robots based on task and anatomical constraints," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 398–403.

[8] J. Burgner, H. B. Gilbert, and R. J. Webster, "On the computational design of concentric tube robots: Incorporating volume-based objectives," in *2013 IEEE International Conference on Robotics and Automation*, 2013, pp. 1193–1198.

[9] H. B. Gilbert, D. C. Rucker, and R. J. Webster III, "Concentric tube robots: The state of the art and future directions," *Robotics Research*, pp. 253–269, 2016.

[10] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[11] 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.

[12] 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.

[13] 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.

[14] C. Bergeles, F. Y. Lin, and G. Z. Yang, "Concentric tube robot kinematics using neural networks," in *Hamlyn Symposium on Medical Robotics*, 2015, pp. 13–14.

[15] 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.

[16] 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.

[17] H. Donat, S. Lilge, J. Burgner-Kahrs, and J. J. Steil, "Estimating tip contact forces for concentric tube continuum robots based on backbone deflection," *IEEE Transactions on Medical Robotics and Bionics*, vol. 2, no. 4, pp. 619–630, 2020.

[18] R. Xu, A. Asadian, S. F. Atashzar, and R. V. Patel, "Real-time trajectory tracking for externally loaded concentric-tube robots," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 4374–4379.

[19] M. Khadem, J. O'Neill, Z. Mitros, L. Da Cruz, and C. Bergeles, "Autonomous steering of concentric tube robots via nonlinear model predictive control," *IEEE Transactions on Robotics*, vol. 36, no. 5, pp. 1595–1602, 2020.

[20] 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*. IEEE, 2011, pp. 945–950.

[21] 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.

[22] J. Burgner-Kahrs, H. B. Gilbert, J. Granna, P. J. Swaney, and R. J. Webster, "Workspace characterization for concentric tube continuum robots," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 1269–1275.

[23] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba, "Hindsight experience replay," in *Advances in neural information processing systems*, 2017, pp. 5048–5058.

[24] Y. Tassa, S. T. A. Muldal, Y. Doron, S. Liu, S. Bohez, T. E. J. Merel, T. Lillicrap, and N. Heess, "dm_control: Software and tasks for continuous control," 2020.

[25] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[26] R. J. Webster, J. M. Romano, and N. J. Cowan, "Mechanics of precurved-tube continuum robots," *IEEE Transactions on Robotics*, vol. 25, no. 1, pp. 67–78, 2008.

[27] T. Lillicrap, J. Hunt, A. Pritzel, N. Heess, N. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[28] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2017, pp. 23–30.

[29] S. H. Sadati, Z. Mitros, R. Henry, L. Zeng, L. Da Cruz, and C. Bergeles, "Real-time dynamics of concentric tube robots with reduced-order kinematics based on shape interpolation," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 5671–5678, 2022.

**Keshav Iyengar** Keshav Iyengar (Student Member, IEEE) is currently a Ph.D. student at University College London in Computer Science under the Welcome / EPSRC Centre for Interventional and Surgical Sciences (WEISS). He received his BASc in Mechanical Engineering at the University of Waterloo, Canada in 2012 and his MSc in Robotics and Computation at University College London in 2018. His research focus is on model-free control of continuum robots for minimally invasive surgery.

**Sarah Spurgeon** Sarah Spurgeon (F'20) received her B.Sc. and D.Phil. degrees from the University of York, UK in 1985 and 1988, respectively, and is now Professor of Control Engineering at UCL, UK. Sarah's research interests are in the area of systems modeling and analysis, robust control, and estimation in which areas she has published over 300 research papers. She was awarded the Honeywell International Medal for 'distinguished contribution as a control and measurement technologist to developing the theory of control' in 2010 and an IEEE Millennium Medal in 2000. She is currently Editor in Chief of IEEE Press and Vice President of Publications of IFAC.

**Danail Stoyanov** Dan Stoyanov is a Professor at UCL Computer Science holding a Royal Academy of Engineering Chair in Emerging Technologies and serving as Director of the Wellcome/EPSRC Centre for Interventional and Surgical Sciences (WEISS). He graduated from King's College London before completing his PhD at Imperial College London. His research interests are focused on surgical robotics, surgical data science, and the development of surgical AI systems for clinical use.