Sim2Real Transfer of Reinforcement Learning for Concentric Tube Robots

Keshav Iyengar Student, IEEE, S.M.Hadi Sadati Member, IEEE, Christos Bergeles Senior Member, IEEE, Sarah Spurgeon Fellow, IEEE, Danail Stoyanov Senior Member, IEEE

Abstract—Concentric Tube Robots (CTRs) are promising for minimally invasive interventions due to their miniature diameter, high dexterity, and compliance with soft tissue. CTRs comprise individual pre-curved tubes usually composed of NiTi and are arranged concentrically. As each tube is relatively rotated and translated, the backbone elongates, twists, and bends with a dexterity that is advantageous for confined spaces. Tube interactions, unmodelled phenomena, and inaccurate tube parameter estimation make physical modeling of CTRs challenging, complicating in turn kinematics and control. Deep reinforcement learning (RL) has been investigated as a solution. However, hardware validation has remained a challenge due to differences between the simulation and hardware domains. With simulation-only data, in this work, domain randomization is proposed as a strategy for translation to hardware of a simulation policy with no additionally acquired physical training data. The differences in simulation and hardware forward kinematics accuracy and precision are characterized by errors of 14.74±8.87 mm or 26.61±17.00 % robot length. We showcase that the proposed domain randomization approach reduces errors by 56 % in mean errors as compared to no domain randomization. Furthermore, we demonstrate path following capability in hardware with a line path with resulting errors of 4.37 ± 2.39 mm or 5.61 ± 3.11 % robot length.

Index Terms—Surgical robotics, Robot control, Reinforcement learning, Concentric tube robots

I. Introduction

CONCENTRIC tube robots (CTRs) are a type of continuum robotic system, first introduced by Furusho et al. [2] and later established [3], [4], that consist of a number of concentric tubes arranged such that they can be controlled independently. These tubes are usually made of a composite of nickel and titanium known as nitinol but have also been 3D printed [5], [6]. With multiple tubes of different diameters placed one inside each other to form a telescoping structure as seen in Fig. 1, the overall shape of the system can elongate, bend and twist to navigate through complex and narrow environments encountered in medicine, such as blood vessels, bronchi or other cavities in the body [7]. Furthermore, the working channel can be exploited for optical imaging and ultrasound [8].

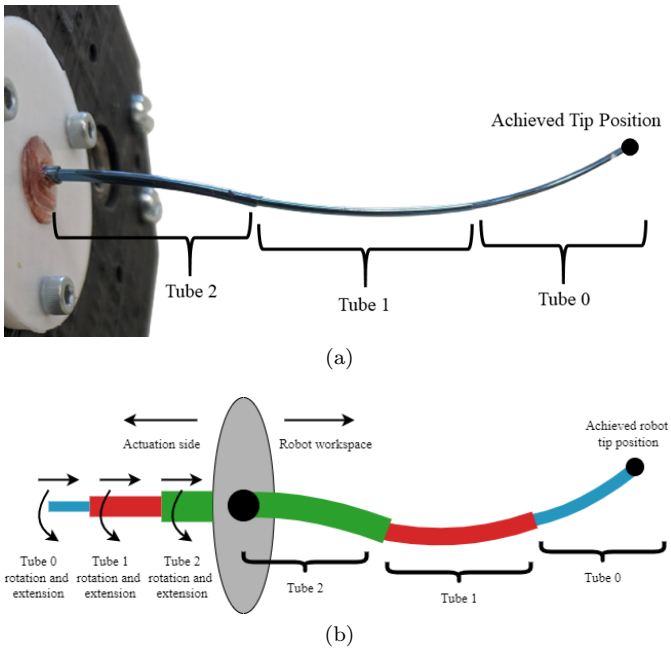


Fig. 1: (a) Real CTR system with tubes adapted from [1] and (b) illustration with tubes, actuation and robot workspace.

Although CTRs have several advantages over traditional rigid robots for minimally invasive surgery (MIS) including increased flexibility and maneuverability, controlling these robotic systems is challenging. Learning-based approaches for kinematics, shape estimation, and dynamics have shown promise and deep learning-based forward kinematics and shape estimation for CTRs have been shown to be more accurate than traditional methods by training on a large data set of training data [9], [10], [11], [12], [13]. In terms of simulation data effectiveness,

Manuscript received: May, 12, 2023; Revised July, 26, 2023; Accepted July, 30, 2023. This paper was recommended for publication by Editor Jens Kober upon evaluation of the Associate Editor and Reviewers' comments.

This work was supported by the Wellcome/EPSRC Centre for Interventional and Surgical Sciences (WEISS) at UCL (203145Z/16/Z), EPSRC (EP/P027938/1, EP/R004080/1), by the Wellcome/EPSRC Centre for Medical Engineering at KCL (WT 203148/Z/16/Z), an ERC Starting Grant (714562), and an NIHR Cardiovascular MIC Grant. For the purpose of open access, the author has applied a CC BY public copyright license to any author-accepted manuscript version arising from this submission

Danail Stoyanov is supported by a Royal Academy of Engineering Chair in Emerging Technologies and an EPSRC Early Career Research Fellowship (EP/P012841/1). Keshav Iyengar and Danail Stoyanov are with the Wellcome/ EPSRC Centre for Interventional and Surgical Sciences (WEISS), University College London, London W1W 7EJ, UK. Sarah Spurgeon is with the Department of Electronic and Electrical Engineering, University College London, London W1W 7EJ, UK. Christos Bergeles and S.M.Hadi Sadati are with the School of Biomedical Engineering and Imaging Sciences, King's College London, London WC2R 2LS, U.K.

Corresponding author: Keshav Iyengar, keshav.iyengar@ucl.ac.uk.

Digital Object Identifier (DOI): see top of this page.

Sim2Real learning-based methods [14] can aid neural networks and reinforcement learning (RL) policies to adapt-on-the-fly to changing CTR hardware systems. Domain randomization has been shown to be able to adapt to situations not seen in training to solve the Rubik's cube with a dexterous shadow hand [15]. In the context of CTRs, as tube parameters due to manufacturing tolerances are often inaccurate, and friction and permanent strain are disregarded in kinematics models, adapting to new tube parameters, and therefore new CTRs is crucial. Furthermore, fabrication challenges for both 3D printing and heat setting exist such that the results from fabrication can be imprecise [16] which domain randomization may be able to account for. Evidently, only using simulation data with no domain transfer methods proves to be very inadequate. To this end, in this work, we investigate the transfer of previous deep reinforcement learning (DRL) work for inverse kinematics and path following of concentric tube robots in simulation to a hardware system. Specifically, we consider domain randomization as a method of Sim2Real transfer where the hardware CTR system is out-of-distribution as compared to the simulation CTR system for inverse kinematics and path following. The work aims to showcase that simulation data with domain randomization can lead to successful deployment of RL techniques on hardware, without additional hardware data collection. Building on our previous simulation work [17], we aim to generalize to hardware with domain randomization [18] by perturbing tube characteristics and sensor noise to expose the agent to various CTR systems in simulation, thereby bridging the gap to the hardware. The method is evaluated with error metrics on hardware inverse kinematics and path following, with no additional hardware data used for training. Summarizing the main contributions:

- Transfer a simulation policy to hardware with domain randomization.
- Evaluations of error metrics on hardware for simulation-only data with and without domain randomization for inverse kinematics and path following.
- To our knowledge, the first work on Sim2Real for CTRs.

II. Related prior work

To our knowledge, all previous work in the literature concerning DRL for CTRs has been in simulation. However, other deep learning-based approaches have conducted hardware experiments successfully providing better accuracy metrics than model-based methods. Grassmann et al. [9] demonstrated hardware experiments for kinematics with state-of-the-art forward kinematics estimation. Kuntz et al. [12] also demonstrate state-of-the-art hardware experiments for shape estimation using a deep learning-based method. A takeaway was that simulation data plus a large set of hardware data was needed for the best performance, with simulation-only data performing the worst. Furthermore, Liang et al. [13] demonstrated forward kinematic joint-to-shape as well as inverse kinematic shape-to-joint estimation from images using deep learning, however only for simulation.

We have previously investigated the exploration problem associated with the joint sampling of CTRs in simulation [19]. As each joint is composed of a rotation and extension degree of freedom, the action exploration noise must consider the difference in units. In subsequent work, a more accurate kinematics model was used in simulation as well as a novel curriculum training methodology and joint representation to improve sample efficiency. Finally, in [17], we investigated rotational constraints and their effects on training and error metrics and the first proof-of-concept towards generalizing a single policy for a multitude of CTR systems of various workspace sizes. Notably, our prior work has all been in simulation, and hardware validation and verification for deep learning-based approaches for inverse kinematics has remained a challenge.

In another work by Kuntz et al, [12], deep learning is used in forward kinematics shape estimation of the backbone of CTRs. A deep learning approach where data of both simulation and hardware as able to outperform physics-based approaches in error metrics. However, simulation-only data performed the worst. The mean square error along the shaft was 12.40 ± 16.40 mm with only simulation data and 3.03 ± 4.84 mm with both simulation and real data. In this work, we aim to maximize the potential of simulation data by utilizing domain randomization. In terms of cost, simulation data is almost always cheaper to collect than hardware data, and therefore should be exploited as much as possible.

III. Methods

To utilize RL in simulation or hardware, the Markov Decision Process (MDP) of the system must be fully defined for the agent to collect data to complete the given task.

A. Markov Decision Process

The MDP has three main components: the state, reward, and action. In the following section, each component is defined in relation to the task and how the elements of the robot setup relate to the MDP.

State (s_t): The state at timestep t, is defined as the concatenation of the trigonometric joint representation, Cartesian goal error between the current achieved position and desired position, and current goal tolerance. As shown in Fig. 1b, rotation and extension of tube i (ordered innermost to outermost) are α_i and β_i with L_i representing the full length. First, the trigonometric representation [9], γ_i, of tube i is defined as:

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

The rotation can be retrieved by taking the arc-tangent

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

The extension joint β_i can be retrieved directly and has constraints

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

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

due to extension actuation constraints. The Cartesian goal error is the current error between the achieved end-effector position G_a, and desired end-effector position G_d. Finally, the current goal tolerance, $\delta(t)$ is included in the state where t is the current timestep of training [20]. The full state, s_t, can then be defined as:

$$s_t = \{\gamma_1, \gamma_2, \gamma_3, G_a - G_d, \delta(t)\}. \tag{5}$$

In the work by Grassmann et al. [10], a transformation matrix was used to decorrelate the interdependence of the extension joints and normalize extension to 0 to 1. The matrix defined as

$$\boldsymbol{M}_{\mathcal{B}} = \begin{bmatrix} -L_3 & 0 & 0 \\ -L_3 & L_3 - L_2 & 0 \\ -L_3 & L_3 - L_2 & L_2 - L_1 \end{bmatrix} \tag{6}$$

can transform extensions to the interval $[0, 1]$. Extensions can be further transformed to the interval $[-1, 1]$ with a system of linear equations [11]

$$\begin{bmatrix} \beta_3 \\ \beta_2 \\ \beta_1 \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{2}\boldsymbol{M}_{\mathcal{B}} & \frac{1}{2}\boldsymbol{M}_{\mathcal{B}} \cdot 1_{3\times 1} \\ 0_{1\times 3} & 1 \end{bmatrix} \begin{bmatrix} \beta_{3,\mathcal{U}} \\ \beta_{2,\mathcal{U}} \\ \beta_{1,\mathcal{U}} \\ 1 \end{bmatrix}. \tag{7}$$

With this transformation from extensions to normalized extensions, sampling valid extension joints can be done by uniform sampling from the $[-1, 1]$ range. Moreover, at each step during operation, the extension joints can be constrained by verifying all joints are within the interval. Action (a_t): Actions are defined as a change in rotation and extension joint positions, [19]:

$$a_t = \{\Delta\beta_1, \Delta\beta_2, \Delta\beta_3, \Delta\alpha_1, \Delta\alpha_2, \Delta\alpha_3\}. \tag{8}$$

The maximum action in extension and rotation is set to 1.0 mm and 5°.

Goals (G_a, G_d): Goals are the Cartesian positions in the Cartesian workspace of the robot. G_a is the current achieved position of the robot, either in simulation or hardware. In simulation, a forward kinematics model is used to compute the current achieved position. After each step, where new joints are actuated, the current achieved position is recomputed. In hardware, a tracker is used to track the position of the end-effector. Similarly, G_d is the desired goal of the robot end-effector. In simulation, joint positions are sampled using Eqn. 7 at the start of every episode and forward kinematics is applied to determine the desired end-effector position for that episode as seen in Section III. In hardware, however, workspace exploration is performed to determine possible end-effector positions to sample from directly as differences in simulation and hardware may result in some positions not in the hardware workspace.

Rewards (r_t): Rewards are the scalar values returned as feedback by the environment to the agent at the current step. Therefore, at each step, a tuple of state, action, reward, and next state experiences are collected and stored. Sparse rewards are used as they have been shown to be more effective than dense rewards when using hindsight experience replay (HER) [21]. The reward function is defined as:

$$r_t = \begin{cases} 0 & \text{if } e_t \leq \delta(t) \\ -1 & \text{otherwise} \end{cases} \tag{9}$$

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.

To collect data as well as execute actions from the policy, the openAI gym framework [22] was used to develop a simulation and hardware environment. The framework requires the implementation of a reset and step method, with returned values of state, rewards, and termination, allowing for RL algorithm agnostic environments.

B. Simulation System

To collect experiences in simulation, the geometrically exact kinematics model was used to create a simulation environment. The model was first presented in [23] and for externally loaded systems with point and distributed forces in [24], [25]. The model ignores friction, permanent strain, and forces along the backbone of the robot. The environment takes tube parameters describing a CTR system as shown in Table I and computes the current achieved position, G_a, and if the desired goal, G_d has been achieved. The tube parameters were found by photocopying each tube separately on an A4 sheet of paper, then using CAD software to determine the parameters. In previous work, successful inverse kinematics and path following of various CTR systems have been demonstrated, but not transferred to hardware, thus a Sim2Real strategy like domain randomization [18] is required.

1) Domain Randomization: In domain randomization, the aim is to transfer a policy from a simulation (source domain) to a similar but different target domain (physical hardware). In simulation, a set of randomization parameters is selected. As a policy is trained in the source domain, experiences are collected with these randomizations applied, exposing the policy to a variety of environments. In this work, we apply uniform domain randomization, where for each selected randomization parameter, a bounded interval is defined where each parameter is sampled uniformly at the start of each episode. The selected parameters for each tube were curved length, inner diameter, outer diameter, stiffness, torsional stiffness, and curvature. A randomization percentage, ϵ of the nominal value of each parameter was used to determine the upper and lower bounds such that for parameter x, the interval was $[x - \epsilon x, x + \epsilon x]$.

Furthermore, noise was added to the observation through the joint values and end-effector position or

Parameter	Inner	Middle	Outer
Length (mm)	340.36	169.69	72.75
Straight Length (mm)[1]	250.36	82.19	11.72
Curved Length (mm)[2]	90.00	87.50	61.03
Inner Diameter (mm)	0.51	0.70	1.15
Outer Diameter (mm)	0.66	1.00	1.63
Stiffness (GPa)	7.50	7.50	7.50
Torsional Stiffness (GPa)	2.50	2.50	2.50
Pre-curvature (mm^{-1})[2]	24.61	19.12	14.04
Home offset (mm)[3]	24.61	19.12	14.04

[1] There is a straight section of 96.41 mm steel and of 153.95 mm nitinol. [2] These values were calculated by photocopying each tube and measuring manually via CAD software (Autodesk AutoCAD). [3] Home offset is measured from the base plane to the limit switch of each extension block.

TABLE I: CTR Tube Parameters

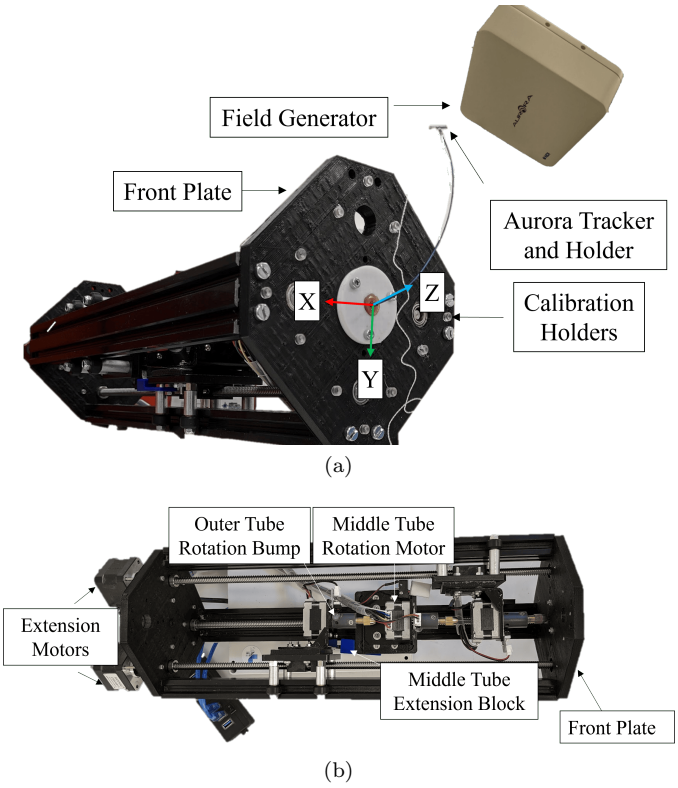


Fig. 2: Hardware experimental setup in (a) the robot workspace with tracking system and (b) the actuation system.

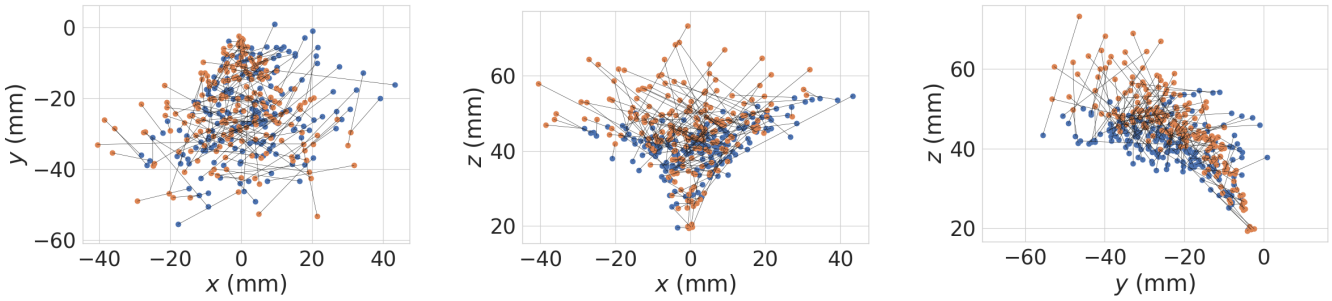
achieved goal tracking. The motor joint positions were perturbed by $\omega = 5°$ resulting in a $5°$ perturbation in rotation positions and 0.005 mm in extension positions. The achieved goal position or tracking of the end-effector was perturbed by $\zeta = 1$ mm as documented by Aurora NDI documentation.

2) Training Methodology: In simulation training, a server cluster with Intel Gold 6130 18C 140W 2.3 GHz with 19 parallel [21] workers for 500,000 training steps with stable baselines [26] was used. A goal-based curriculum with an egocentric joint representation as presented in [20], [17] was employed.

C. Hardware System

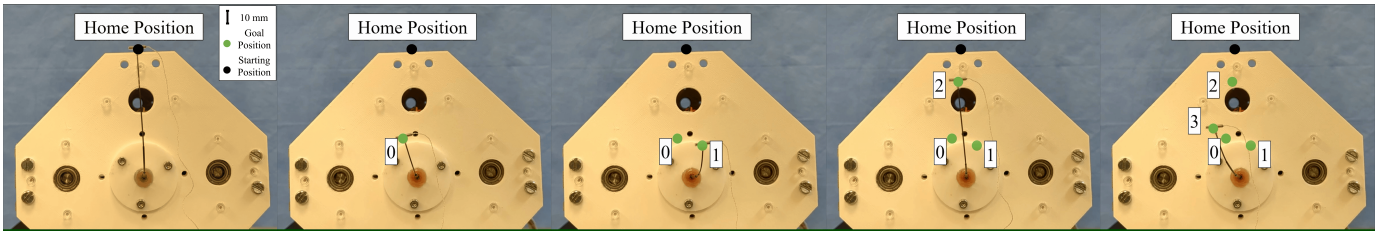
The CTR hardware system consisted of two full nitinol tubes and a third nitinol tube with a partial steel section along the straight length. Three brass collets and chucks were used with custom 3D-printed motor couplings. The robot chassis had three main aluminum extrusions of 200 mm in length with two 3D-printed plates for motor mounts and tube entry. To actuate the tubes in rotation three stepper motors (200 steps per revolution) with a setting of 4 micro-step and for translation, three stepper motors with a 16 micro-step setting were used. The linear rails were of 200 mm stroke and 1 mm lead. For rotation and translation motors, limit switches were incorporated for the initial homing operation. For rotation, 3D-printed rotation bumps were used to set the limit switch, and 3D-printed limit blocks were used for translation. To home the motors, limit switches set the zero position for each rotation and translation motor. In the initial calibration of rotation, each tube is rotated individually with others retracted until the tracker is aligned to the Y-axis. For translation, each motor is set to move forward until a limit switch is hit, starting from outer to inner tube. For motor control, two Arduino microcontrollers with GRBL CNC shields for rotation and translation separately were used. To track the end effector position, the Aurora tracking system was used with a 0.6 mm sensor. A custom tracker holder was designed and 3D printed. To calibrate the base of the robot, calibration holders were designed to be used with the Aurora NDI 6 DOF probe to determine the base plane and base frame of the CTR system.

1) Workspace Analysis: To analyze the characteristics of the CTR hardware system, with respect to simulation, a workspace collection trial was completed whereby joint commands were sent to the system, and once the end-effector was in a steady state, the position of the end-effector with respect to the base frame was collected. Simultaneously, the forward kinematics was computed using the simulation framework and commanded joint positions. This produced the simulation end-effector positions. By comparing the errors, a baseline difference of simulation to hardware was determined which the Sim2Real method will need to overcome. We present errors in two ways, the first is in millimeters and the second is in percentage robot length. In the second, the error is divided by the current extension of the innermost tube as done previously in [10]. The error metrics of simulation to hardware for the workspace were as follows: mean error and standard deviation in (mm) is 14.5 ± 8.2 mm and for percentage robot length, the mean error and standard deviation are 26.24 ± 16.41 %. To examine how well the RL agent can adapt to the hardware system via domain randomization,

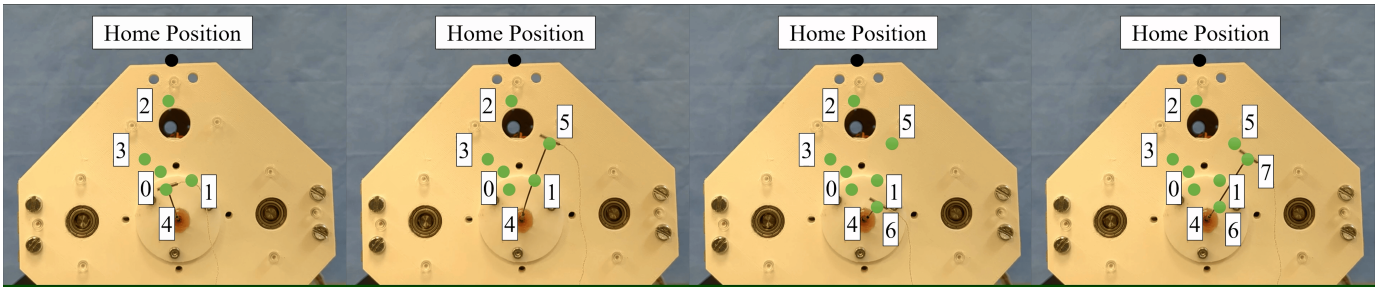


(a) (b) (c)

Fig. 3: Visualization of hardware workspace Cartesian points. The simulation framework with the joint values commanded is used to compute the simulation points (blue) and the actual hardware points tracked are in orange. An (a) $x-y$, (b) $x-z$ and (c) $y-z$ views are shown.



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



(f) (g) (h) (i)

Fig. 4: Home and goal positions used to evaluate each domain randomization method.

no system identification was performed to improve the gap between the simulation and hardware system.

To examine and differentiate the different elements of domain randomization, being that of the sensor noise and model parameter randomization, combinations, and values of randomization elements were experimented with. Once these agents were trained in simulation, each was tasked with solving inverse kinematics tasks in simulation and hardware. Finally, after testing each relevant experiment in a line following task, we discuss the results of domain randomization on error metrics with the hardware system, acknowledging the initial large system deviation from the workspace analysis.

IV. Experiments and Results

To compare these experiments, a baseline experiment with no additional Sim2Real strategies where the learned policy is directly trained with the given CTR parameters for comparison. The effects of domain randomization were explored by performing a set of 10 additional experiments split into two categories. The first category consists of 4 experiments that vary the domain randomization value for 0.1 and 0.2 with the addition of sensor noise. The second category of experiments consists of 6 experiments varying the randomization value for $0.05, 0.08, 0.10, 0.12, 0.15$, and 0.20 with no additional sensor noise. With this set of experiments, the effects of various values of domain randomization with the combination of sensor noise can be presented.

In the following section, first, the simulation training results are analyzed and presented. Success rate and error metrics were recorded throughout simulation training for all experimental policies and inverse kinematics trials were performed in simulation.

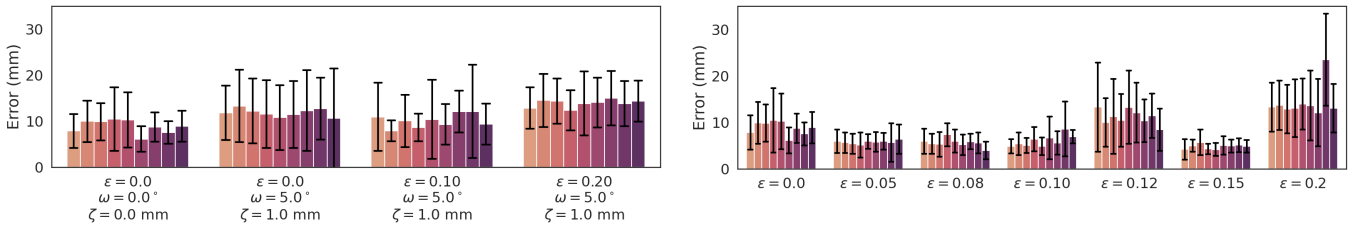


Fig. 5: Experimental errors for each policy with varied domain randomization parameters shown over trials for (a) sensor noise with model randomization and (b) with only model noise.

A. Simulation Training Results

During simulation training of the experimental policies, all were able to converge to below 1.0 mm error by the end of the 2 million steps of training. Success rate, measured as a percentage of successful episodes over total evaluation episodes, was consistently $> 97.5\%$ after the first $50,000$ because of the use of the decay curriculum [20]. Having this curriculum training method greatly improved training times as successful episodes took fewer steps as a result of the larger goal tolerance. Moreover, in the final steps of training, when the policy is already converged, having the required 1.0 mm still results in a high success rate. To evaluate the experimental policies, inverse kinematics was performed for 100 randomly selected points in the robot workspace based on hardware joint constraints. The final errors and the success rate was tabulated. With no domain randomization or sensor noise ($\epsilon = 0$, $\omega = 0°$, $\zeta = 0$ mm) error metrics were 0.88 ± 1.87 mm. For the first category of experiments, where sensor noise of $\omega = 5°$ and $\zeta = 1.0$ mm was incorporated, the simulation results are as follows. With no randomization $\epsilon = 0.0$ errors were 0.76 ± 0.73 mm. With domain randomization of $\epsilon = 0.1$ and $\epsilon = 0.2$ errors were 0.81 ± 1.31 mm and 0.77 ± 0.27 mm. In the second category without sensor noise and domain randomization only varying from 0.05, 0.08, 0.10, 0.12, 0.15, and 0.20, the error metrics are presented below. In order, errors were 0.50 ± 0.82, 0.59 ± 1.05, 0.60 ± 0.92 mm, 0.90 ± 1.85, 0.60 ± 0.90 and 0.73 ± 0.99 mm.

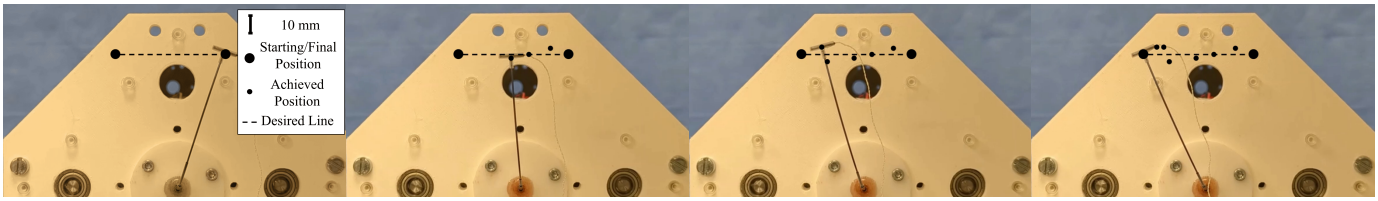
B. Hardware Results

The inverse kinematics hardware experiments consisted of the robot end-effector starting at the home position, and attempting to reach each desired goal in sequence, before returning to the home position. This is performed for each experiment, and repeated for 9 trials. The desired goals are 9 randomly selected positions in the robot workspace with the sampling method presented in [11] and remain the same for all experiments. First, the effects of sensor noise are illustrated in Fig. 5a over trials where each trial is shaded. In Fig. 5b, the error metrics are shown to illustrate the improvements with different values of domain randomization with no sensor noise. To summarize inverse kinematics results, we present error metrics per experiment in mm as well as percentage robot length over trials as follows. We first present errors with respect to adding sensor noise and domain randomization as shown in Fig. 5a. With no domain randomization or sensor noise ($\epsilon = 0.0$, $\omega = 0.0°$, $\zeta = 0.0$ mm), error metrics were 8.56 ± 4.58 mm or 19.86 ± 11.29 %. Adding only sensor noise ($\epsilon = 0.0$, $\omega = 5.0°$, $\zeta = 1.0$ mm) errors were 13.91 ± 5.39 mm or 27.86 ± 13.93 %. With sensor noise and domain randomization of 0.10 and 0.20, error metrics were 11.86 ± 7.87 mm or 23.25 ± 15.48 % for ($\epsilon = 0.10$, $\omega = 5.0°$, $\zeta = 1.0$ mm) and 10.10 ± 6.34 mm or 19.43 ± 11.75 % for ($\epsilon = 0.20$, $\omega = 5.0°$, $\zeta = 1.0$ mm). Removing sensor noise and evaluating increments of domain randomization of 0.05, 0.08, 0.10,0.12, 0.15, and 0.20 the following metrics were found as seen in Fig. 5b. The error metrics for each increment were 5.78 ± 2.63 mm or 12.14 ± 5.70 %, 5.63 ± 2.45 mm or 12.19 ± 5.53 %, 6.05 ± 3.31 mm or 13.35 ± 8.34 %, 11.20 ± 6.74 mm or 22.69 ± 11.8 %, 4.83 ± 1.18 mm or 11.06 ± 6.22 % and 14.37 ± 7.38 mm or 28.39 ± 13.29 %.

In path following, a line across the workspace discretized into 8 individual points was used to evaluate various domain randomization increments including $\epsilon = 0.0$, $\epsilon = 0.05$, $\epsilon = 0.08$, $\epsilon = 0.10$, $\epsilon = 0.12$, $\epsilon = 0.15$ and $\epsilon = 0.20$. For each respective randomization increment, the errors were 8.95 ± 3.90 mm or 11.49 ± 4.96 %, 4.37 ± 2.39 mm or 5.61 ± 3.11 %, 4.45 ± 2.19 mm or 5.41 ± 2.65 %, 4.45 ± 2.57 mm or 5.59 ± 3.21 %, 9.49 ± 4.34 mm or 11.53 ± 5.82 %, 8.08 ± 4.36 mm or 10.59 ± 6.15 % and 6.46 ± 4.22 mm or 8.09 ± 5.28 %. The error metrics are summarized in Fig. 7.

V. Discussion

Despite the large errors found when comparing the simulation modeling to the hardware implementation, the policy is able to approximately reach desired goals and follow a discretized line path with domain randomization. For inverse kinematics, the best domain randomization found was 0.15 although 0.05, 0.08, and 0.10 performed adequately. 0.12, however, did not perform well as compared to neighboring values. No domain randomization and 0.20 performed poorly as no randomization does not help the Sim2Real gap and too much randomization does not collect enough data on the correct hardware



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

Fig. 6: Line path experiments for $\epsilon = 0.05$.

Domain Randomization Parameters			Inverse Kinematics			Path Following	
ϵ (%)	ω (°)	ζ (mm)	Simulation Errors (mm)	Hardware Errors (mm)	Hardware Errors (%)	Hardware Errors (mm)	Hardware Errors (%)
0	0	0	0.88 ± 1.87	8.56 ± 4.58	19.86 ± 11.29	8.95 ± 3.90	11.49 ± 4.96
0	5	1	0.76 ± 0.73	13.91 ± 5.39	27.86 ± 13.93	–	–
0.1	5	1	0.81 ± 1.31	11.86 ± 7.87	23.25 ± 15.48	–	–
0.2	5	1	0.77 ± 0.27	10.10 ± 6.34	19.43 ± 11.75	–	–
0.05	0	0	0.50 ± 0.82	5.78 ± 2.63	12.14 ± 5.70	4.37 ± 2.39	5.61 ± 3.11
0.08	0	0	0.59 ± 1.05	5.63 ± 2.45	12.19 ± 5.53	4.45 ± 2.19	5.41 ± 2.65
0.10	0	0	0.60 ± 0.92	6.05 ± 3.31	13.35 ± 8.34	4.45 ± 2.57	5.59 ± 3.21
0.12	0	0	0.90 ± 1.85	11.20 ± 6.74	22.69 ± 11.8	9.49 ± 4.34	11.53 ± 5.82
0.15	0	0	0.60 ± 0.90	4.83 ± 1.18	11.06 ± 6.22	8.08 ± 4.36	10.59 ± 6.15
0.20	0	0	0.73 ± 0.99	14.37 ± 7.38	28.39 ± 13.29	6.46 ± 4.22	8.09 ± 5.28

TABLE II: A summary of experimental results for various domain randomization parameters in simulation and hardware for the tasks of inverse kinematics and path following. Inverse kinematics simulation results are over 100 trials of randomly selected desired goal points, however, inverse kinematics hardware results are a sequence of 9 desired goal points repeated over 9 trials. Path following results is for a straight line along the workplace. Hardware errors are given in millimeters and % of robot length.

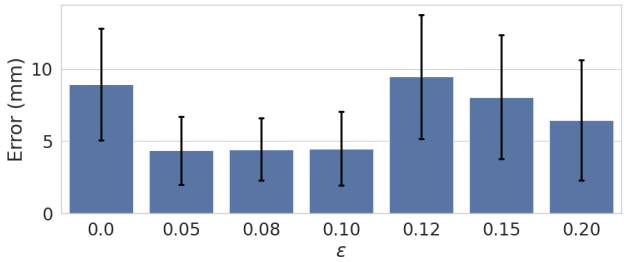


Fig. 7: Path following error metrics for each domain randomization over 3 trials.

parameters. During simulation training, the joint rotation of the tube remains unconstrained for the agent to explore the entirety of the workspace. However, in hardware, the robot rotation joints are constrained to $\pm 60°$. For $\epsilon = 0.12$, this causes issues when reaching desired goals. In analyzing the rotational joint commands by the agent in $\epsilon = 0.12$, it is evident the agent is aiming to rotate joints past the constraints, and therefore, unable to progress. The mean and standard deviation of final rotational joints in order of innermost to outermost for $\epsilon = 0.12$ was $9.44° \pm 38.64°$, $-66.80° \pm 78.25°$ and $198.93° \pm 218.11°$ whereas for $\epsilon = 0.10$ the values were $-2.03° \pm 25.15°$, $19.25° \pm 34.81°$ and $-6.53° \pm 24.34°$. When training with constrained rotation, we found results remained similar in simulation inverse kinematics results.

In the line following task, similar to the inverse kinematics task, 0.05, 0.08, and 0.10 performed well over the trials, with no randomization and 0.20 performing not as well. Other factors may be affecting the performance of certain policies depending on the region where the desired goals are set. For example, although 0.15 performed well in inverse kinematics, it does not perform as well in path following. Finally, similar to inverse kinematics, 0.12 does not perform well in path following due to large tube rotations. The calibration procedure for the tracking system, inverse kinematics experimental results, and line following paths are attached in the associated video.

As mentioned in Section II, the work by Kuntz et al. [12] compares accuracy results for different sources of data (simulation, real, and simulation + real) in the task for shape estimation. Looking at mean squared error along the shaft metrics for the best network parameters for

each were 12.1 ± 16.5 mm, 3.49 ± 4.72 and, 3.03 ± 4.84 mm using simulation, real and, simulation + real data respectively. Although errors are difficult to compare, as errors are presented in mm rather than % robot length, and a different learning task, it is evident that simulation-only data performs the worst when evaluated on hardware. Further study will be required to see the effects of domain randomization with simulation data, with the addition of hardware data.

VI. Conclusion

In this paper, the effects of domain randomization in simulation training and the resulting error metrics are evaluated on a CTR hardware system in inverse kinematics and a line-following task. The benefits are demonstrated by comparing error metric results to the workspace characterization metrics comparing simulation and hardware. Moreover, with the availability of more accurate real-time CTR dynamics [27] and Finite Element [28] models, better simulation data will further improve the error metrics achievable on hardware. More work is needed to understand the peak behavior as related to ϵ. It could be possible the performance was linked to the path chosen and future studies may focus on varying these paths, in a more accurate dynamics model, to understand the effects of randomization on policy convergence.

References

[1] 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.

[2] J. Furusho, T. Ono, R. Murai, T. Fujimoto, Y. Chiba, and H. Horio, "Development of a curved multi-tube (cmt) catheter for percutaneous umbilical blood sampling and control methods of cmt catheters for solid organs," in IEEE International Conference Mechatronics and Automation, 2005, vol. 1, 2005, pp. 410–415 Vol. 1.

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

[4] R. J. Webster, A. M. Okamura, and N. J. Cowan, "Toward active cannulas: Miniature snake-like surgical robots," in 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2006, pp. 2857–2863.

[5] T. K. Morimoto and A. M. Okamura, "Design of 3-d printed concentric tube robots," IEEE Transactions on Robotics, vol. 32, no. 6, pp. 1419–1430, 2016.

[6] K. Picho, B. Persons, J. F. d'Almeida, N. E. Pacheco, C. Reynolds, and L. Fichera, "Multi jet fusion of nylon-12: a viable method to 3d-print concentric tube robots?" arXiv preprint arXiv:2204.00505, 2022.

[7] Z. Mitros, S. Sadati, R. Henry, L. D. Cruz, and C. Bergeles, "From Theoretical Work to Clinical Translation: Progress in Concen-tric Tube Robots," Annual Review of Control, Robotics, and Autonomous Systems, 2021.

[8] G. Dwyer, E. J. Alles, R. J. Colchester, K. Iyengar, A. E. Desjardins, and D. Stoyanov, "Robot-assisted optical ultrasound scanning," IEEE Transactions on Medical Robotics and Bionics, vol. 3, no. 4, pp. 948–958, 2021.

[9] 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.

[10] R. M. Grassmann and J. Burgner-Kahrs, "On the merits of joint space and orientation representations in learning the forward kinematics in se(3)," Robotics: Science and Systems XV, 2019.

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

[12] 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.

[13] 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.

[14] W. Zhao, J. P. Queralta, and T. Westerlund, "Sim-to-real transfer in deep reinforcement learning for robotics: a survey," in 2020 IEEE Symposium Series on Computational Intelligence (SSCI), 2020, pp. 737–744.

[15] I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas et al., "Solving rubik's cube with a robot hand," arXiv preprint arXiv:1910.07113, 2019.

[16] C. J. Nwafor, C. Girerd, G. J. Laurent, T. K. Morimoto, and K. Rabenorosoa, "Design and fabrication of concentric tube robots: A survey," IEEE Transactions on Robotics, 2023.

[17] K. Iyengar, S. Spurgeon, and D. Stoyanov, "Deep reinforcement learning for concentric tube robot path planning," 2023. [Online]. Available: https://arxiv.org/abs/2301.09162

[18] 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.

[19] K. Iyengar, G. Dwyer, and D. Stoyanov, "Investigating exploration for deep reinforcement learning of concentric tube robot control," International Journal of Computer Assisted Radiology and Surgery, vol. 15, no. 7, pp. 1157–1165, 2020.

[20] 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.

[21] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. Pieter Abbeel, and W. Zaremba, "Hindsight experience replay," Advances in neural information processing systems, vol. 30, 2017.

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

[23] 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.

[24] 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.

[25] 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.

[26] A. Hill, A. Raffin, M. Ernestus, A. Gleave, A. Kanervisto, R. Traore, P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, and Y. Wu, "Stable baselines," https://github.com/hill-a/stable-baselines, 2018.

[27] S. M. H. Sadati, Z. Mitros, R. Henry, L. Zeng, L. d. 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.

[28] K. Zuo, B. Jackson, R. Henry, C. Bergeles, and S. M. H. Sadati, "Finite Element Dynamics of a Concentric Tube Robot Motion and Interaction with Environment Using SOFA-framework," in Hamlyn Symposium on Medical Robotics (HSMR), 2022, p. 2.