POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

Deep Reinforcement Learning for Concentric Tube Robot Control

Laurea Magistrale in Biomedical Engineering - Ingegneria Biomedica

Author: Lorenzo Valente

Advisor: Prof. Elena De Momi

Co-advisor: Ing. Keshav Iyengar

Academic year: 2021-2022

1. Introduction

The Minimal Invasive Surgery (MIS) technique offers a more efficient and less invasive way to perform many surgical procedures. MIS uses tiny incisions or natural orifices to enter the surgical site, resulting in less pain, less tissue damage, and reduced hospitalization time. However, during complex procedures where the path to follow is deep and tortuous, MIS strategy is more efficient if coupled with Robotic Assisted (RA), dexterous and flexible instrumentation. Multi-segmented arms with high degrees of freedom (DOF) like Concentric Tube Robot (CTR) are one of the most investigated technologies in those cases, they can deform and adapt to the external environment, limiting tissue damage. CTRs consist of concentrically arranged precurved tubes, usually made of Ni-Ti alloy, that can be singularly actuated through axial extension and rotation generating by means of their interactions curvilinear shapes of the robot backbone. With the increased number of DOF and reduced dimensions, CTRs may be beneficial for ablation procedures like Fetoscopic Laser Coagulation for Twin-Twin Transfusion Syndrome. In this procedure no tip forces are involved, however the robot tip position control is essential since a correct distance from the tissue must be maintained in order to deliver the correct laser power. Due to complex tube's interaction, modelling and control of CTRs is challenging. From literature analysis [1, 2], the control strategies based on a kinematic model have some issues regarding the balance between modelled physical phenomena and sufficient performance for real-time integration. Thus, the proposed model-free Deep Reinforcement Learning based controller aims at overcoming model-based issues. The developed controller is intended for a real control loop scenario where the surgeon is performing a procedure through a teleoperated haptic device for Cartesian position control of CTR tip position and the proposed controller calculates the inverse kinematics in order to reach that position or follow that trajectory.

2. Background

Among robots' control techniques, Reinforcement Learning (RL) has had an emergence as popular for solving complex control tasks. RL is a subfield of machine learning (ML) concerned with how an agent can learn to take actions in an environment to maximize a reward signal. The agent is a learner, while the environment is a context in which the agent takes actions. The goal is to optimize the policy of the agent such

that it maximizes the reward it receives from the environment.
Particularly, model-free learning methods became popular among CTRs' control strategies being able to overcome tube interaction complexity and modelling for kinematics. In this work a model-free RL approach is experienced where the agent learns directly from the data, without building a model of the environment. Instead, it learns to associate actions with states based on the reward signals it receives. A typical RL problem is described as a Markov Decision Process (MDP) that is a classical formalization of sequential decision-making, where actions influence not just immediate states but also subsequent situations and it involves: agent, environment, states, actions and rewards.
In this case high-dimensional states and actions are involved, thus is necessary to move to a Deep Reinforcement Learning (DRL) framework, which includes neural networks in the policy optimizing process. In particular two DRL compatible algorithm has been tested, Proximal Policy Gradient (PPO) and Advantage Actor Critic (A2C).

3. Implementation

The agent is represented by the entire robot and it interacts in a simulated environment where the possible actions are extension and rotation of each robot's tube in a free space.
State. The observation state s_t at timestep t is defined by a trigonometric joints representation γ_i of each tube, the Euclidean norm error e_t between current tip position and desired tip position, both defined through a Forward Kinematics (FK) problem solution where the input values are the current and desired joint values respectively, and the current error tolerance $\delta(t)$:

$$\begin{aligned} \gamma_i &= \{cos(\alpha_i), sin(\alpha_i), \beta_i\}, \\ s_t &= \{\gamma_1, \gamma_2, \gamma_3, e_t, \delta(t)\}, \end{aligned} \tag{1}$$

the current error tolerance $\delta(t)$ is a value useful for reward function definition and is updated during training following a constant, linear or exponential decay with respect to timestep.
Action. At each step the agent selects joints values in order to reach the target. Thus, actions are changes in extension and rotation for each tube, these changes are constrained in a range that is $\pm 1mm$ for extension and $\pm 5°$ for rotation. The agent can select any values in the continuous range between the limits, actions can be then defined as:

$$a = (\Delta\beta_1, \Delta\beta_2, \Delta\beta_3, \Delta\alpha_1, \Delta\alpha_2, \Delta\alpha_3) \tag{2}$$

The joint constraints described in eq. 5 are implemented through a check that happens after each action is selected, actions values are clipped in a way that constraints are respected.
Reward. The action taken will lead the agent to a new state with new observations. The policy will be updated according to a reward, which is a scalar value returned by the environment as feedback from the chosen action. Dense and sparse reward function has been tested. The sparse rewards strategy in RL aims to give feedbacks for a small handful of states, in this case is defined as:

$$r = \begin{cases} 0, & e_t \leq \delta \\ -1, & \text{otherwise,} \end{cases} \tag{3}$$

While, with dense reward function the feedback signal is given at each step and defined as:

$$r = \begin{cases} 0, & e_t \leq \delta \\ -e_t, & \text{otherwise,} \end{cases} \tag{4}$$

4. Materials and Methods

4.1. CTR model

The current study will focus on the following CTR system, made of three concentrically arranged tubes with index i going from the innermost to the outermost tube as see Fig. 1, where each tube length L_i is the summation of the curved L_i^c and straight L_i^s section. Each tube has two DOFs, a rotation α_i and an extension β_i; defining in this way a full joint configuration of the entire system in the form of $q = [\beta_1, \beta_2, \beta_3, \alpha_1, \alpha_2, \alpha_3]$. The extension constraints due to actuation limitations are the following:

$$\begin{aligned} &\beta_i \in [-L_i, 0) \\ &\beta_1 \leq \beta_1 \leq \beta_1 \leq 0 \\ &0 \leq L_3 + \beta_3 \leq L_2 + \beta_2 \leq L_1 + \beta_1 \end{aligned} \tag{5}$$

Where the zero position is defined based on the reference frame represented in Fig. 1.

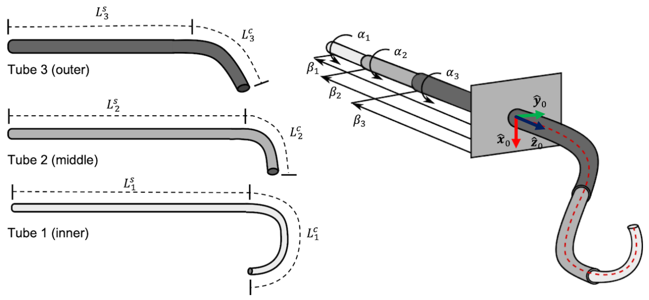

Figure 1: CTR system notation. $alpha_i$ relative rotation, $beta_i$ relative extension, L_i^c precurved tube length, L_i^s straight tube length, $(\hat{x_0}, \hat{y_0}, \hat{z_0})$ reference frame attached to the robot base where tubes can fully retract.

4.2. Simulation environment

The environment that simulates a CTR interacting in a free space has been developed following the OpenAI framework to collect data and experiences useful for the DRL algorithms to train a policy. The environment takes as input the CTR's tube parameters listed in Tab. 1 and the selected actions by the agent to compute the overall CTR's backbone shape.
The training process is divided in episodes that are sets of training steps with a defined maximum length. At the beginning of each episode a target point is selected inside of CTR workspace, through Forward Kinematics(FK) problem solution based on the Constant Curvature CTR kinematic model [4]. Then for each step of the episode, the agent selects an action and the reward feedback is assigned based on the achieved end-effector position computed through FK based on the same kinematic model. The episode terminates if the number of steps overcomes the maximum length or if the current observed error is below the current tolerance. Once an episode is terminated the final robot pose becomes the starting pose for the next episode and another point in the workspace is selected.

4.3. Methods

To train the model has been exploited Stable Baselines 3(SB3) which is a set of reliable implementation of DRL algorithms. Among the types of SB3 policy network, the Multi Layer Perceptron (MLP) one has been selected for this work. Two different network for actor and critic have been selected with two hidden layers of 256 units each.
To the implemented simulation environment the following methods have been applied to compare the control performance:

- PPO, a DRL on-policy policy gradient algorithm
- A2C, a DRL on-policy actor-critic algorithm
- DDPG + HER, in a previous similar work Iyengar et. al [3] applied Deep Deterministic Policy Gradient (DDPG) an off-policy policy gradient algorithm to the same environment with the addition of Hindsight Experience Replay (HER) strategy to improve training convergence.
- Jacobian-based controller that is one of the most common method for CTR control with the following closed form law to steer the CTR:

$$\dot{q_d} = J^{\dagger}[\dot{x_d} + K_p(x_d - x)] \tag{6}$$

  where $J^{\dagger}$ is the pseudoinverse of the robot Jacobian, K_p is a symmetric positive definite matrix, given a desired joint values change $\dot{q_d}$ as control input, $\dot{x_d}$ the desired change in Cartesian space and x_d the desired Cartesian position.

All the DRL algorithms has been tested with both dense and sparse reward functions.

	1st	2nd	3rd
Length	340.36	169.69	72.75
Length Curved	90.00	87.50	61.03
Inn. Diameter	0.51	0.70	1.15
Out. Diameter	0.66	1.00	1.63

Table 1: CTR tube parameters measured in mm (1st is the innermost tube)

4.4. Hyperparameters Tuning setup

An essential step for an RL training process is *Hyperparameters Tuning*, these parameters are a set of values that will heavily impact on training performance. Several automatic hyperparameters optimization methods have been developed, the one exploited in this study is a framework called Optuna. The optimizer runs the environment for a predefined number of trials and steps per trial, for each trial a set of hyperparameters values are sampled from a group of standard values. The result of this research is the best combination of parameters that comes from the best trail which is the trail with the highest reward. In this study a *Random* sampler has been selected which determines the value of a single parameter without considering any relationship between parameters; while a *Median* pruner has been chosen, which will terminate a trial if its best intermediate result is worse than median of intermediate results of previous trials at the same step. The optimization has been performed over 600 trials of 10000 episodes each.

5. Experimental Setup

The tuned hyperparameters have been used as input for the training process. Both PPO and A2C algorithms are compatible with continuous state and action spaces environments, which is a requirement for the reaching task involved in this case. The algorithms have been applied to the environment using SB3 implementations. Each algorithm has been tested with longer and shorter training, three and one million steps respectively. SB3 implementation supports environment parallelization, thus 1 and 8 parallel environments trainings have been tested. All the three goal tolerance decay functions has been evaluated, resulting the linear the one with better performance. During each training every 10000 steps the policy is evaluated on the same environment and the results of this evaluation recorded. The training performance evaluation has been conducted by looking into these metrics:

- *episode reward mean*, the average reward of an episode for all the training's episodes
- *episode length mean*, the average episode length for all the training's episodes
- *error*, distance between desired and achieved end-effector position calculated at the end of each training's episode
- *success rate*, percentage of success among the evaluation episodes. The success of an episode is established if the distance between desired and achieved end-effector position is below the current tolerance.

Each trained policy has been evaluated on the same simulation environment, thus with the same tube parameters as input, through two kind of experiments:

- **targeting**, the trained policy is used to predict joint values in order to reach a target point inside of the task space. The policy has been tested over 500 target points. The trained agent interacts with the environment trying to reach the goal, the episode stops if the error is below 1 mm and then, in the next episode, a new target is selected. The robot starting pose is resampled at each episode.
- **path following**, the trained policy is used to predict joint values in order to follow a predefined path. Two kind of path have been tested: line and circle. The trajectory is built as a series of consecutive points that the robot should reach in order to follow the path. The first target point is sampled inside of the task space and then the following target points form a line or a circle. The path following test is repeated with 10 different starting points for each kind of trajectory and the results averaged.

The quality of the obtained results from these two experiments has been established by looking at the error distance between the achieved Cartesian tip position G_a and the desired one G_d calculated as:

$$E = \sqrt{(G_{dx} - G_{ax})^2 + (G_{dy} - G_{ay})^2 + (G_{dz} - G_{az})^2} \quad (7)$$

and episode length, that defines how fast is the policy to find joint values that perform the required task. These kind of evaluation tests are relevant as similar to a real surgical scenario where a surgeon control the end-effector tip position through a teleoperated haptic device giving Cartesian coordinates as input for the trained policy.

6. Results

6.1. PPO vs A2C

Training results show that PPO outperforms A2C in both three and one million steps training and with both sparse and dense reward functions. PPO converges to a 100% success rate in less than 500000 steps, also reward and episode length show better performance. Fig. 2 demonstrates how PPO outperform A2C also in the targeting evaluation test. The average error distance with its standard deviation and the

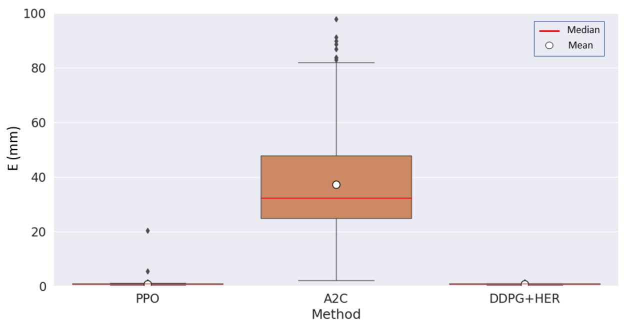

Figure 2: Evaluation error result of the three DRL methods, distance between desired and achieved end-effector position.

episode length mean of the targeting test are described in Tab. 2.

	Av.	St. dev.	Ep. Len.
PPO	0.88 mm	0.90 mm	26.33 steps
A2C	37.2 mm	17.9 mm	50 steps
DDPG+HER	0.83 mm	0.16 mm	25.81 steps

Table 2: Targeting results of the three DRL methods: average and standard deviation of error E, episode length.

6.2. Dense vs sparse reward function

Observing training performance metrics for PPO one million steps training for dense and sparse reward comparison, it can be inferred that dense reward converges to 100% success rate while sparse does not. Both the trained policies has been evaluated with a targeting experiment, the average error distance measured in the experiment is 0.8mm with 0.6mm standard deviation for the dense reward policy, while for sparse reward the mean is 29.3mm and the standard deviation 13.8mm.

6.3. PPO vs Jacobian

A line trajectory path-following test with 10 different starting points has been exploited to evaluate and compare a Jacobian controller, with $K = 2I$, and a PPO policy trained with a dense reward function for one million steps. In Fig. 3 is shown one of the most successful experienced test, while in most of the case the Jacobian controller steered the robot far away from the target

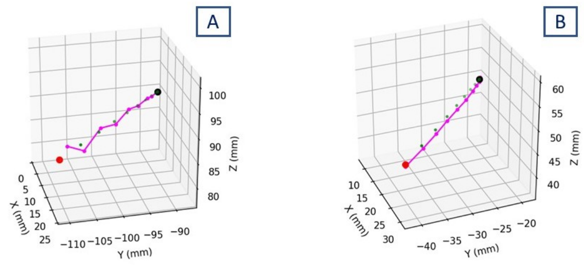

Figure 3: PPO vs Jacobian controller evaluation test, the black dot is the starting point of the trajectory and the red one is the ending point while the green dots describe the trajectory. The pink line is the actual end-effector trajectory. **(A)** is the Jacobian controller trajectory, **(B)** is the PPO trajectory.

trajectory, thus the error metric of this successful test is not a clear evidence of the bad Jacobian performance.

6.4. PPO vs DDPG+HER

In [3] has been demonstrated that DDPG needs 2 million steps and 19 parallel workers to converge, while PPO method is able to converge with 8 parallel workers and 1 million steps training showing on-policy method ability to learn faster for this environment. This sample efficiency is reflected also in the overall training time parameter that settles around 10 hours for DDPG+HER training and around 4 hours for PPO training. DDPG+HER and PPO policies has been evaluated over the same CTR simulation environment with both targeting and path following test. Observing path following results in Fig. 4 and targeting results in Fig. 2 and Tab. 2 the same accuracy for both methods can be inferred.

6.5. Domain Randomization

The last experiment has been conducted as an initial proof of concept of PPO ability to learn a more general policy. The concept of domain randomization is to sample at each episode a set of parameters with a variation in tube's parameters inside of a defined range. The chosen range in this experiment was 10%, at each episode a random value of the parameter P among tube's parameters inside the range between $P + 0.1 * P$

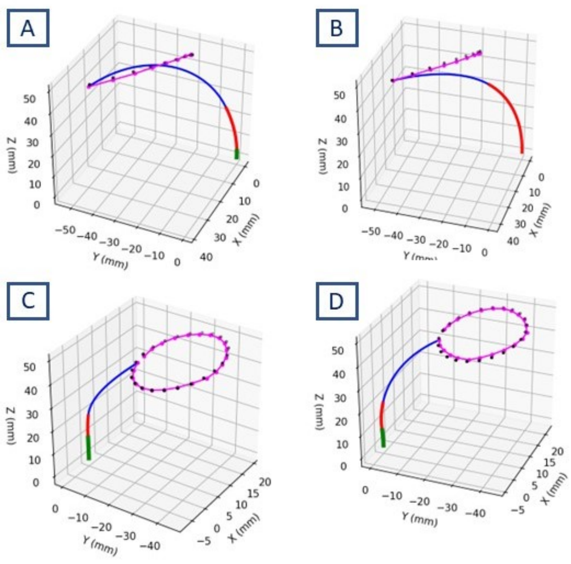

Figure 4: Path following test. **(A),(C)** refers to PPO policy evaluation, **(B)(D)** refers to DDPG policy evaluation

and $P - 0.1 * P$ is selected and the episode executed with the selected parameters. This kind of environment has been trained with PPO for one million steps. The trained policy has been evaluated with a targeting task for 200 points in 10%, 20%, 30% and 40% randomization environment and the average and standard deviation of the distance error with the episode length are listed in Tab.3.

	Av.	**St. dev.**	**Ep. Len.**
10% Rand.	1.62mm	6.8mm	28.6 steps
20% Rand.	1.28mm	2.99mm	26.82 steps
30% Rand.	2.74mm	10.9mm	30.45 steps
40% Rand.	2.39mm	6.4mm	31.31 steps

Table 3: Targeting results from Randomization: average and standard deviation of error E, episode length

7. Conclusion

Analysing the results it is evident that PPO outperform A2C, this behaviour is justifiable by the clipped feature of PPO's loss function that keeps the policy changes in a limited range increasing the convergence to an optimal solution probability. Moreover, being PPO an on-policy method is reasonable that a dense reward function is a better fit because the policy is updated online based on current experiences, thus a more frequent reward signal is necessary to push the agent in the right direction. In addition, the trained PPO policy can follow both a linear and circular path with a 0.7mm average error, while Jacobian controller most of the time fails moving far away from the target. This is mainly due to the absence of joint limits in Jacobian formulation that steers the robot to configuration it cannot recover from. While the advantage of PPO with respect to the previous work DDPG method sample efficiency can be useful in a real scenario where a procedure-specific control strategy could be necessary, thus PPO's ability to learn a good policy with a smaller amount of collected experiences is helpful. On-policy nature of PPO is also useful to learn a policy in a dynamic environment where parameters change during training as demonstrated in section 6.5. In conclusion it has been demonstrated that PPO is a valid method for CTR control, improving both accuracy with respect to standard methods and sample efficiency with respect to previous work method, some initial proof of concept for PPO policy generalization has been also proved.

References

[1] Hessa Alfalahi, Federico Renda, and Cesare Stefanini. Concentric tube robots for minimally invasive surgery: Current applications and future opportunities. *IEEE Transactions on Medical Robotics and Bionics*, 2(3):410–424, 2020.

[2] Jessica Burgner-Kahrs, D. Caleb Rucker, and Howie Choset. Continuum robots for medical applications: A survey. *IEEE Transactions on Robotics*, 31(6):1261–1280, 2015.

[3] Keshav Iyengar, Sarah Spurgeon, and Danail Stoyanov. Deep reinforcement learning for concentric tube robot path planning. *arXiv preprint arXiv:2301.09162*, 2023.

[4] III Robert J. Webster and Bryan A. Jones. Design and kinematic modeling of constant curvature continuum robots: A review. *The International Journal of Robotics Research*, 29(13):1661–1683, 2010.