POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Deep Reinforcement Learning for Concentric Tube Robot Control

Tesi di Laurea Magistrale in
Biomedical Engineering - Ingegneria Biomedica

Author: **Lorenzo Valente**

Student ID: 963989
Advisor: Prof. Elena De Momi
Co-advisors: Ing. Keshav Iyengar
Academic Year: 2021-22

Abstract

Minimal Invasive Surgery (MIS) has introduced a new and more efficient way of performing many surgical procedures; accessing the surgical site through tiny incisions leads to less pain, less tissue damage, and reduced hospitalization time. MIS has faced skepticism from clinicians due to confined workspace, compromised hand-eye coordination, extended learning curves, and procedure duration. The introduction of Robotic-Assisted instrumentation has helped with instrument manipulation. However, when dealing with complex procedures where the interested area is very deep and the path to follow is tortuous instrumentation with higher dexterity and flexibility is needed. Multi-segmented arms with high degrees of freedom (DOF) like Concentric Tube Robot(CTR) are one of the most investigated technologies to solve those issues. CTRs are made of multiple pre-curved telescopic tubes, which actuation mechanism is based on independent axial translation and rotation of each tube making kinematics and control challenging. Ablation procedures like Fetoscopic Laser Coagulation for Twin Twin Transfusion Syndrome are eligible for CTRs employment. CTRs learning-based control strategies have been demonstrated to outperform the classical model-based approach. In this thesis, a model-free Deep Reinforcement Learning(DRL) method has been investigated as a control strategy to be involved in a loop control where the surgeon selects a Cartesian point or trajectory through a haptic device and the controller computes the inverse kinematics to achieve the target. The controller is represented by the trained policy obtained from a DRL problem solution where an agent interacts with a CTR simulation environment learning to choose the correct joint values to perform a targeting task. Two DRL algorithm has been tested, PPO and A2C, and compared with previous work DDPG algorithm and a Jacobian-based model. Observing training metrics as well as policy evaluation test, performed through targeting and path-following tasks, PPO turns out to outperform both A2C and Jacobian-based methods for tracking error and computational time, while reaching the same DDPG tracking error in a significantly inferior number of training steps.

Keywords: Concentric Tube Robot, Control, Kinematics, Deep Reinforcement Learning

Abstract in lingua italiana

La Chirurgia Mininvasiva (CM) ha introdotto una nuova e più efficiente strategia per svolgere molte procedure chirurgiche, accedendo alla zona da operare tramite piccole incisioni cutanee che riducono il danno ai tessuti, provocano meno dolore e minimizzano il tempo di ospedalizzazione.La CM ha generato alcuni scetticismi nel mondo della medicina a causa di un ridotto spazio di lavoro, una compromessa cordinazione occhio mano, estese curve di apprendimento e per la durata delle procedure. L'introduzione di strumentazione robotica ha aiutato nella manipolazione degli strumenti, ma comunque in caso di procedure complesse, dove l'area interessata è molto in profondità e il percorso per raggiungerla tortuoso, una strumentazione con maggiore destrezza e flessibilità è necessaria. Bracci robotici multisegmentati con elevati gradi di libertà (GDL) come i Robot a Tubi Concentrici(RTC) sono tra le tecnologie più esaminate per risolvere queste problematiche. I RTC sono composti da multipli tubi precurvati e disposti telescopicamente, il cui meccanismo di attuazione è basato sulla traslazione e rotazione assiale relativa di ogni tubo, rendendo impegnativi la formulazione della cinematica e il controllo. I RTC possono essere sfruttati per procedure di ablazione come la Coagulazione Fetoscopica con Laser per curare la sindrome da trasfusione feto-fetale. Le strategie di controllo dei RTC basate su insiemi di dati risultano essere più performanti rispetto a quelle basate su modelli cinematici. In questa tesi, una strategia model-free Deep Reinforcement Learning (DRL) è stata studiata come metodo da includere in un processo di controllo in cui idealmente un chirurgo seleziona un punto o una traiettoria Cartesiani attraverso un dispositivo aptico e il controllore risolve la cinematica inversa per raggiungere il target. Il controllore in questo caso è rappresentato dalla rete neurale allenata tramite DRL. La fase di allenamento prevede l'interazione di un agente con un ambiente di simulazione del RTC, e la selezione dello stesso di valori dei giunti necessari per raggiungere il punto selezionato. Due algoritmi DRL sono stati testati, PPO e A2C, e comparati con il metodo del precendente lavoro, DDPG, e con un modello di controllo. Osservando i risultati degli allenamenti così come quelli dei test di valutazione, PPO risulta avere performance migliori rispetto a A2C e al modello Jacobiano per quanto riguarda l'errore di tracciamento e il tempo computazionale, mentre raggiunge lo stesso errore di tracciamento di DDPG ma in un numero significativamente

inferiore di passi durante la fase di allenamento.

Parole chiave: Robot a Tubi Concentrici, Controllo, Cinematica, Deep Reinforcement Learning

ABBREVIATIONS

Minimal Invasive Surgery **(MIS)**
Concentric Tube Robots **(CTR)**
Twin-Twin Transfusion Syndrome **(TTTS)**
Fetoscopic Laser Coagulation
Inverse Kinematics **(IK)**
Forward Kinematics **(FK)**
Boundary Value Problem **(BVP)**
Model Predictive Controller**(MPC)**
Initial Value Problem **(IVP)**
Multilayer Perceptron
Deep Reinforcement Learning **(DRL)**
Reinforcement Learning **(RL)**
Markov Decision Process **(MDP)**
Actor-Critic **(AC)**
Advantage Actor-Critic **(A2C)**
Proximal Policy Optimization **(PPO)**
Deep Deterministic Policy Gradient **(DDPG)**
Hindsight Experience Replay **(HER)**
Stable Baselines 3 **(SB3)**

Contents

1 | Introduction

1.1. Minimally Invasive and Robotic Assisted Surgery

One of the most important discovery in the surgical field is the Minimal Invasive Surgery (MIS) technique, which has established a new and more efficient way to intend many of the surgical procedures. With respect to open surgery what makes MIS a preferred practice is that the area where the surgeon acts is limited to the damaged area and the surrounding healthy tissues should not be affected. In fact in MIS usually for entry to the surgical site the accesses, single or multiple, are very tiny incisions or the natural orifices are exploited. This kind of approach enables less pain, less tissue damage and reduced hospitalisation time [1]. A large number of clinical trials about short and long-term recovery from numerous kind of surgical procedures comparing laparoscopic and open approaches were conducted [2, 3]. They highlight all the benefits of the avoidance of large wounds which leads to minimal morbidity due to post-operative immobility, shorter convalescence, improved cosmesis and an improved immune response.
However, MIS encountered also some skepticism since its introduction due to several drawbacks addressed by clinicians like confined workspace, lack of stereo vision, loss of depth perception, compromised hand-eye coordination, extended learning curves and procedure duration [4]. Thanks to the technological innovations in the medical field like the introduction of miniaturized high-resolution cameras, stereoscopic systems or force-feedback sensory systems those limits have been minimized. Moreover, the introduction of Robotic-Assisted instrumentation has helped the surgeon overcome the difficulties related to instrument manipulation during MIS procedures. In this way, the tools are not controlled directly by the surgeon but through controllers and with the benefits of articulation, tremor filters and haptic feedback surgeon's dexterity and eye-hand coordination are enhanced, improving surgical performance. One of the first and more successful robotic system for surgery is the da Vinci SP (Intuitive Surgical, Sunnyvale, California, USA)(Fig. 1.1). Its master console is equipped with controllers that the surgeon uses to steer the slave unit, made of up to four arms and a 3D HD Camera. Custom-made tools and articulated end-effectors can be mounted to the robotics arms allowing for MIS

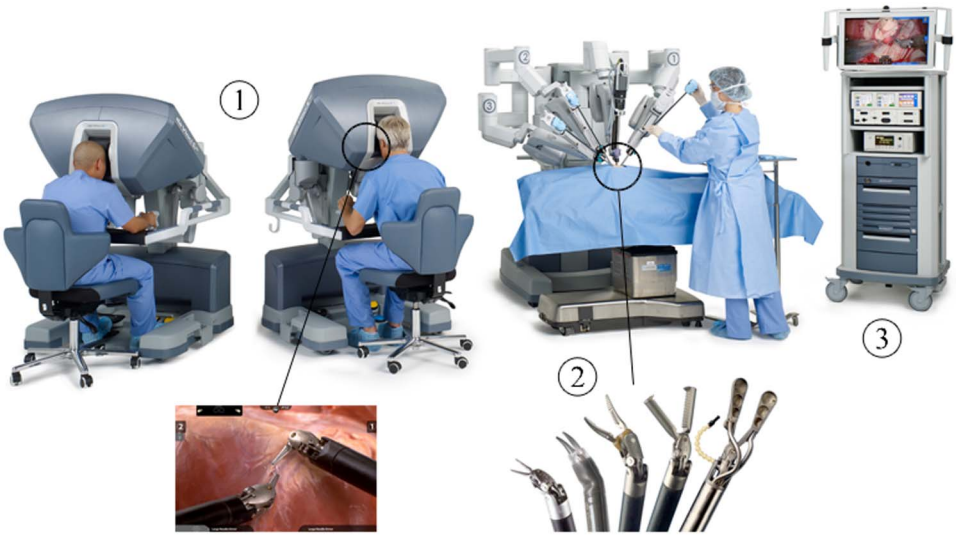


Figure 1.1: The da Vinci surgical platform

procedures.
However, while a surgeon is dealing with difficult procedures like brain and skull-based surgery or eye and deep-orbital intervention, where the interested area is very deep and the path to follow is tortuous, rigid instrumentation even if mounted on robotics arms could be limiting. Instrumentation with higher dexterity and flexibility is needed.
The introduction of multiple segments and flexible kind of instrumentation enables MIS to reach a very high accuracy and resolution. For a single-segment rigid arm is impossible to follow body cavities with different curves without damaging other tissues. Hence, the robotic community has focused its attention on Continuum Robots (CR). CR are flexible multi-segmented arms with high DOF and typically constructed at smaller scale than rigid arms thanks to the simplicity of their structures. This kind of structure makes them useful in procedures where narrow and complex passages inside the body are involved, CR are capable to deform and adapt to the external environment and their limited dimensions can limit tissues damages. Burgner et al. [5] classified CR based on their structural design and actuation strategy as see in Fig. 1.2.

1.2. Concentric Tube Robots

Among CR technologies Concentric Tube Robot (CTR), also known as Active Cannulas(AC), have been the subject of much investigation in the last years [5]. Made of multiple precurved telescopic super-elastic Nickel-Titanium (NiTi) alloy tubes, CTR actuation mechanism is based on axial translation and rotation of each tube with respect

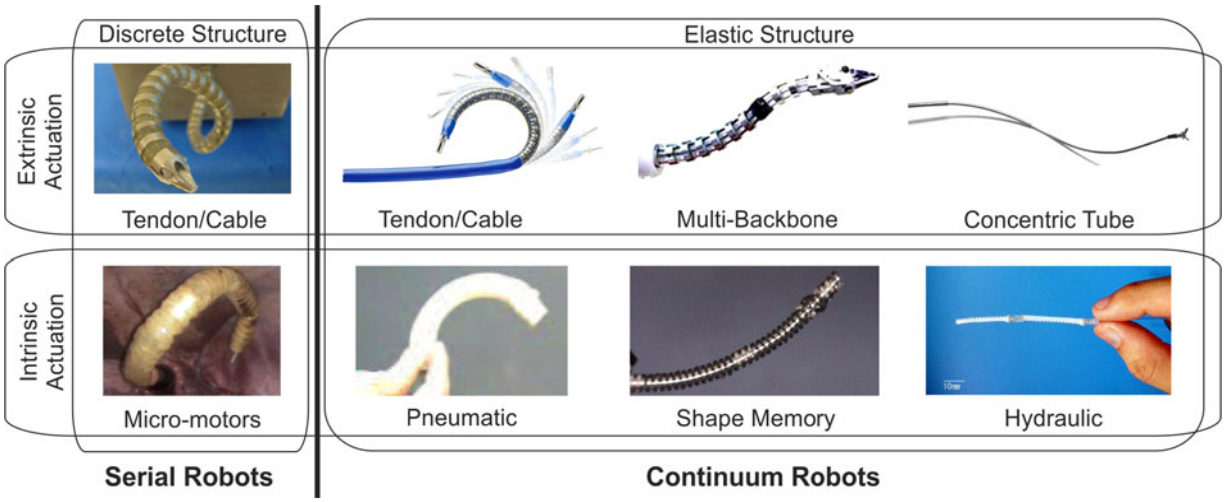


Figure 1.2: CR classification based on [5]

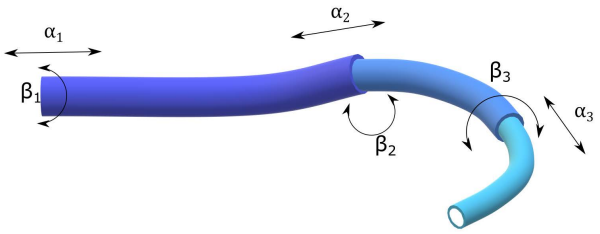
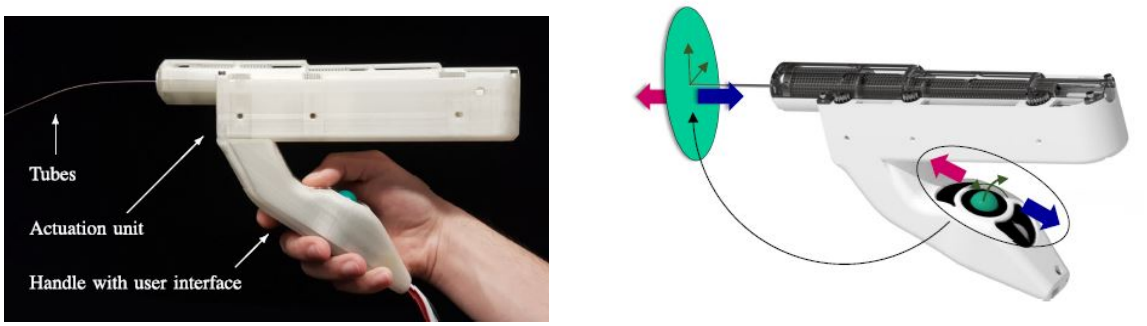


Figure 1.3: Rotation and translation of each tube

to the others as see in Fig. 1.3. Bending actuation arises from tubes elastic interaction resulting in a high curvature steering which enables to overcome complicated passages inside body cavities. Since the actuation method is separate from the backbone itself, the robot diameter can be reduced, essential for narrow paths.

CTRs can be classified based on the stiffness model: Dominant-Stiffness Tube Pair with fixed curves that consequently is more patient-specific, while on the other hand Balanced-Stiffness Tube Pair, with flexible curvatures, where all the tubes have the same stiffness [6].

The first motorized system was the Furusho et al. [7] (2005) curved multi-tube catheter for percutaneous umbilical blood sampling and control methods, where the tip position of the central needle is controlled by rotation and translation of the two outermost tubes. CTRs can also be hand-held like Girerd et al. 6-DOF CTR [8] as see in Fig. 1.4(a) which still offers a higher dexterity with respect to other hand-held rigid tools. The device is equipped with a handle composed of a trackball between two buttons: pressing the button in the front leads the robot's tip moving forward, while pressing the button in the back leads the tip moving backward, the trackball enables the user to control in-plane motions



(a) Hand-held CTR device.

(b) Control units

Figure 1.4

(Fig. 1.4(b)). Experiments, conducted with this device by clinical operators, have been carried out. An operator was asked to use the trackball and buttons to have the CTR tip follow two different paths, repeating each path task three times. Thanks to a 6-DOF sensor attached to CTR tip to sense its position, the tip positioning errors are measured with an average of $0.7mm$ for path 1 and $0.9mm$ for path 2.

CTRs have been primarily designed for diagnostic purposes and here is where we can classify CTRs as steerable needles like Sears et al. [9] and Gilbert et al. [10] solutions. In particular in the latter model a "follow the leader" deployment was applied which was enabling the robot to follow exactly the path described by the tip, very useful to avoid healthy tissue damage.

On the other hand, CTRs as a surgical manipulator needs accurate steering and navigation through body lumens and position and force control of the distal tip because it will interact directly with tissues. This is the case of endonasal skull base surgery, a technically challenging procedure with standard rigid tools, that with a telerobotic system like Burgner et al. [11] one can potentially increase surgical dexterity. Here, two concentric tubes manipulators are teleoperated via haptic devices under endoscopic visualization as see in Fig. 1.5, one manipulator end effector is a curette and the other one is a gripper. They demonstrate how users of widely varying skill levels and backgrounds are able to achieve laparoscopic training tasks.

An improved version of a multi-arm CTR is the Wang et al. system, made of three channels: one is a four-DOF active vision arm, the other two are six-DOF manipulators with scissors or forceps. This active cannula is also intended for transnasal procedures.

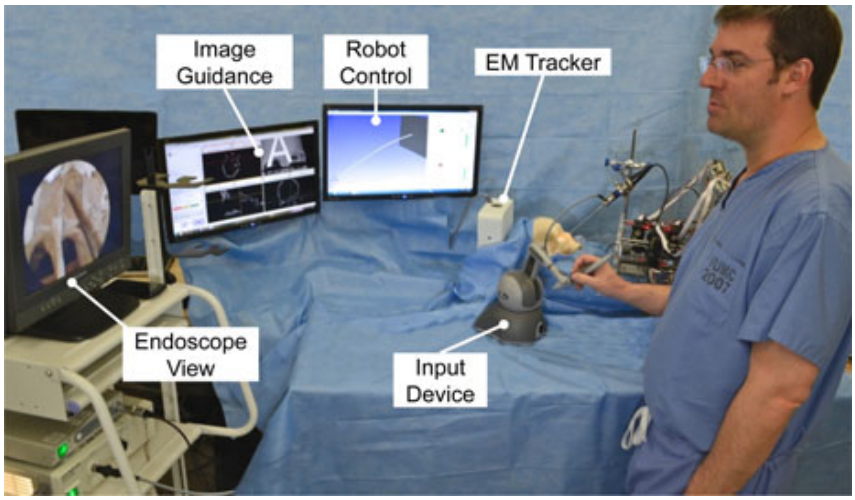


Figure 1.5: Teleoperated CTR for transnasal surgery

1.3. Twin-twin transfusion syndrome treatment

Among all the possible clinical scenarios this study is more addressed to ablation procedures where robot end-effector tip control is crucial with respect to full shape control. Fetoscopic interventions could potentially benefit from CTR technology, as delicate manoeuvres under poor visualisation conditions are required and high accuracy is essential as both mother and fetus wellbeing are involved.

Twin-Twin Transfusion Syndrome (TTTS) is a condition that requires a fetoscopic intervention. This pathology occurs in monochorionic diamniotic (MCDA) twin pregnancy where anastomoses (communications between the two fetal circulations within the placenta) can results in unbalanced blood flow between twins, in this way the two fetuses grow at different rates. The prevalence of TTTS is approximately 1-3 per 10,000 births [12], over three-fourths of diagnosed stage I TTTS remain stable or regress without invasive intervention. Furthermore, TTTS accounts for up to 17 % of total perinatal mortality in twins, and for about half of perinatal mortality in MCDA twins [13].

Fetoscopic Laser Coagulation (FLC) of placental anastomoses is considered to be the best approach for stages II, III and IV TTTS. In fact, according to Simpson et al. [13] trials analysis: perinatal deaths are 44% of the cases after laser photocoagulation while 61% after amnioreduction, and free of neurological complications are the 52% of the cases after FLC and 32% after amnioreduction.

FLC procedure involves coagulating placental anastomoses, so interrupting some links, that give rise to TTTS. Coagulation is performed by using a laser fibre through the working channel of a fetoscope, thus a small skin incision is required, where a small diameter cannula is inserted under ultrasound guidance as see in Fig 1.6. Usually, a rigid fetoscope

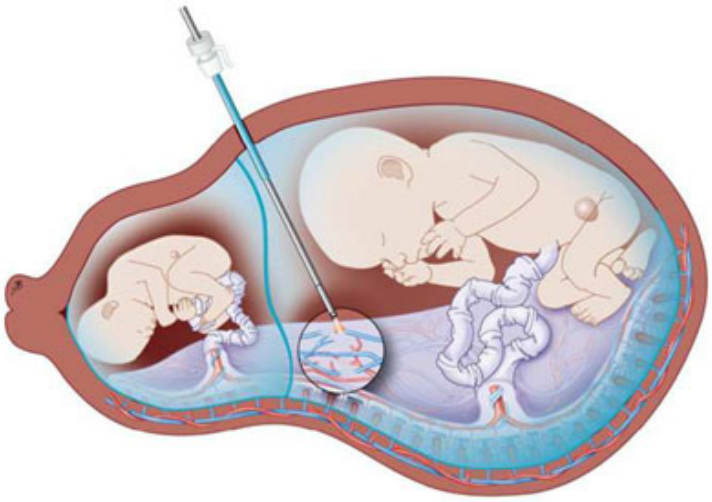


Figure 1.6: FLC procedure for the treatment of TTTS, showing the endoscope positioned to coagulate the placental vessel anastomoses

is used to observe the placenta and coagulate vessels, but due to the small incision motions are constrained and so the treatable area is limited. Moreover, the surgeon must maintain a correct distance from the placenta in order to deliver appropriate laser power to the tissue and also because physical contact with the placenta may cause bleeding and lead to a loss of sight and other complications [14]. A robotic system like CTR can overcome these difficulties: the augmented dexterity of these robots helps in observing and delivering therapy to the placenta; while stability can be reached by controlling the instrumentation with an articulated arm.
CTR should be compatible with laser fibres technology and this kind of ablation procedure facilitates end-effector position control since no forces are involved at the robot tip.

1.4. Aim of the work

To perform complex tasks, like the above-mentioned FLC surgery, accurate robot tip position control is needed. In order to move the robot end-effector the actuation unit imposes some changes on the joint values, the analytical relationship between joint positions and the robot tip's position and orientation is known as *kinematics*. The *forward kinematics* problem involves the determination of a general method to describe the end-effector motion as a function of joint motion. In contrast, the *inverse kinematics* problem is the process to compute the joint values that would enable the robot tip to achieve a specific pose. While, *differential kinematics* describes the analytical relationship between the joint motion and the end-effector motion in terms of velocities, through the manipulator Jacobian.
The availability of a kinematic model for most robotics systems is essential for the resolution of these problems but in the case of CTRs finding this model is not so straightforward mainly because of the complex interactions between tubes which lead to non-linear backbone curvature. Hence, model-based strategies for controlling CTRs do not allow for real-time applications since the complexity of a very accurate kinematic model increases too much the computational cost. For this reason, Iyengar et al. focused on the use of an accurate forward kinematics solver to collect data and experiences for training, then a data-driven method to predict output from the collected data as inputs. In this case the collected data come from a CTR simulated environment and a Reinforcement Learning training is used to learn from those data different tasks like targeting and path following. This method aims to combine accuracy and computational speed of data-driven approach in order to pass physical model-based control limits like the difficulties to model phenomena such as friction, shear, plastic deformation that lead to errors.
The aim of this thesis is to enhance the above mentioned model and the standard control methods in order to:

- improve the accuracy, so minimize the error between desired and achieved end-effector position or trajectory;
- decrease computational cost, so minimize the training time and number of steps;
- optimize generalization, make the model functional for several CTRs systems.
- include joint extension constraints to avoid unstable CTR configurations that cannot be avoided in traditional control methods.

1.5. Thesis structure

In the first chapter of this thesis 2 an overview of the most common CTRs control methods is conducted, starting from model-based methods and going through learning-based and hybrid ones. while a recap about the most important RL methods is coupled with a detailed description of the exploited CTR simulation environment and of the training strategy within the Material and Methods chapter 3. In chapter 4 the performed experiments to validate the proposed method with their results are discussed, while chapter 5 is an analysis of the obtained results. In the final chapter 6 the overall work is resumed and some future developments are discussed.

2 | Background

In this section an overview of the most relevant state of the art CTRs control methods is conducted. We can classify these methods into model-based and learning-based approaches. Then, an hybrid solution between the two approaches is analyzed. In the end of the chapter an overview of Reinforcement Learning theory is conducted.
Accurate control of CTRs is essential as they move in constrained environments, so a good controller has to avoid any possible tissue damage and at the same time be able to reach targets in the surgical workspace for ablation or biopsy.

2.1. Related works

MODEL-BASED POSITION CONTROL

One approach for solving the Inverse Kinematics (IK) or Forward Kinematics (FK) problem for CTRs analytically is to use geometric techniques to derive expressions for the lengths and curvatures of the tubes as a function of the end-effector position and orientation. These expressions can then be used to calculate the required tube lengths and curvatures for a desired end-effector position and orientation.
For example, [15] presents a geometric method to solve IK for single- and multi-section continuum robots. In this work, the single-section part of the algorithm models the continuum manipulator as an arc of a circle and calculates its arc length s, its curvature k and its direction of curvature ϕ as a function of the desired endpoint position P. These trunk parameters can be converted into desired actuator parameters using the robot-specific mapping. The single-section inverse kinematics can be iteratively applied to multiple, serially linked continuum sections to model an n-section continuum manipulator. In this case the values of s, k and ϕ can be computed for each section by determining the values of s, k and ϕ for the base section, subtracting the translation due to the base section from the remaining endpoints, applying the opposite rotation due to the base section to the remaining endpoints, then repeating this process with the remaining sections. This algorithm is tested in a closed-loop controller.

While Dupont et al. in [16] developed a tool-frame position control which involves solving the IK and FK problems for CTRs at real-time rates. To implement this in real-time, a pre-computation of the FK solution over the robot's workspace is performed, and it is then approximated with a truncated Fourier series. Moreover, the FK model used in this work includes torsional compliance along the entire length of the tubes [17–19], with respect to previous models which include torsion only in the straight proximal portion of the tubes. The IK solution is calculated at each time step by using a root-finding method on the functional approximation. Therefore, a closed-loop control as in (Fig. 2.1) was deployed, where the master arm is a haptic device and the slave arm is a CTR.
The complexity of the Boundary Value Problem (BVP) model for CTRs brought researchers to study new computationally efficient IK methodologies. In Leibrandt et al. [20] work for example it was decided to approximate the Jacobian of the robot by finite differences on the solution of the BVP. While Rucker et al. [21] tried to increase the computational efficiency of Jacobian estimation of CTR under external loading. As well as Xu et al.'s work [22] that discretizes the robot into small sublinks for real-time estimation of the robot Jacobian. For all these cases the Jacobian was then exploited for closed-loop control of the robot.

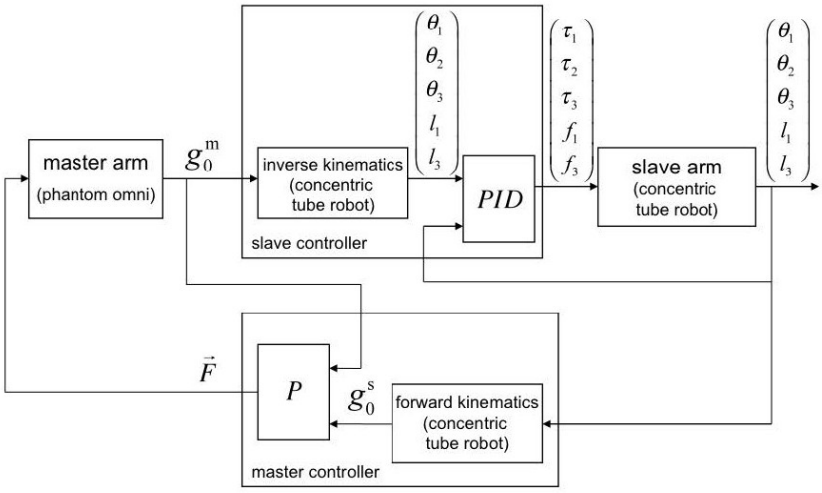
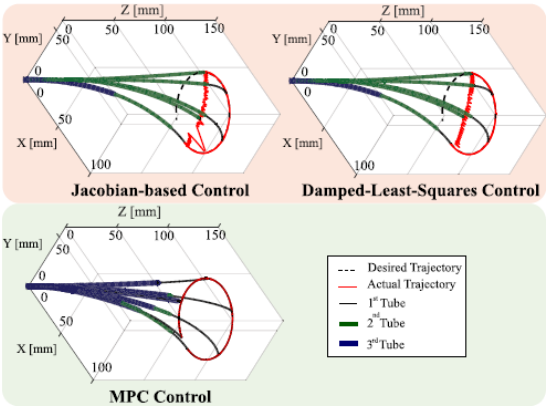
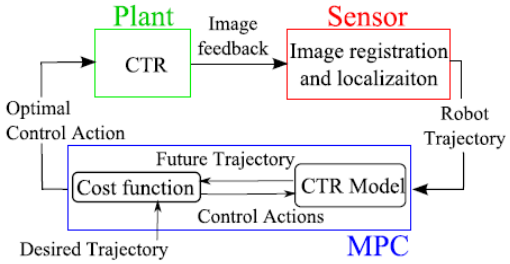


Figure 2.1: Dupont et al. control diagram

It should be noted that these control methods rely only on contemporary information and this could lead the robot to instabilities or configurations from which it cannot recover. This is mainly due to workspace constraints, influenced by tubes' precurvature, that make the tip trajectory and joint values dependent on the path already taken. This is why Khadem et al. [23] decided to develop a novel Model Predictive Controller(MPC) in order



(a) MPC controller and Jacobian-based controller comparison



(b) MPC block diagram

Figure 2.2

to steer CTR's end-effector inside tortuous pathways avoiding unstable configurations that may lead for instance to robot's neighbouring tubes colliding, while Jacobian approach does not take account of these joint limits. This model predicts future robot behaviours and based on those it controls robot's inputs. The advantage of this approach is evident in Fig. 2.2(a), where state of the art Jacobian-based controllers steer the robot blindly and unstable configurations are reached with respect to MPC controller which is able to follow optimal paths.

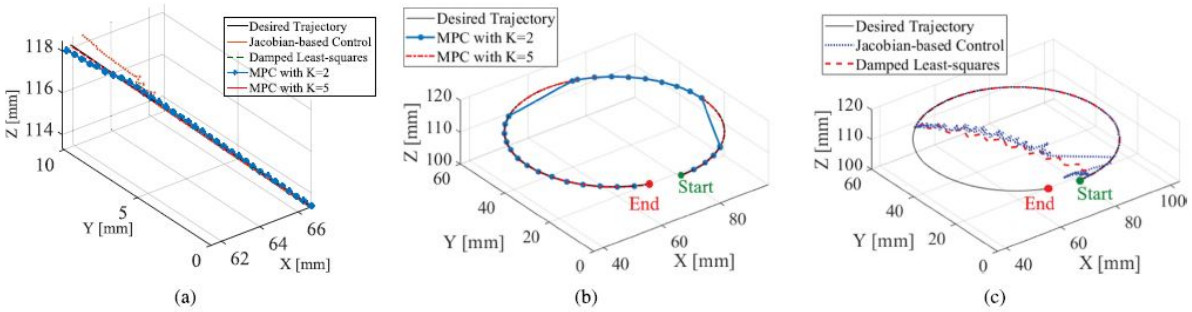
Looking at the block diagram in Fig. 2.2(b) is inferred that the idea of MPC controller is based on the measurement of the state of the system through image feedback at each sampled time instant which will be used as initial boundary condition to solve the Initial Value Problem (IVP) of CTR. The solution to this problem will be extended to a defined time horizon generating future trajectories, then the MPC controller try to optimize the robot predicted behaviour with proper control inputs. The controller generates a certain control action based on the minimization of a quadratic cost function:

$$L(x_p, q) = (x_p(t) - x_d(t))W(x_p(t) - x_d(t))^T \tag{2.1}$$

where $x_d(t)$ is the desired trajectory and $x_p(t)$ is the predicted future trajectory given an initial value of control input q_0 and a control sequence $q(0)$. The predicted trajectory $x_p(t)$ is calculated through the CTR kinematic model:

$$y(t_{k+1}) = g(y(t_k), q(t_k)), \tag{2.2}$$

$$x_p(t_{k+1}) = f(y(t_{k+1})). \tag{2.3}$$



(a) MPC,Jacobian and DLS controller evaluation

COMPARISON OF JACOBIAN-BASED AND MPC CONTROLLERS

	MPC	Jacobian-based	Damped least-squares
RMSE[mm]	0.8	5.3	4.1
σ [mm]	0.5	3.1	2.4

(b) Error comparison

Figure 2.3

where k is the length of the time horizon.

In order to include the boundary condition at the robot's tip was implemented a methodology where CTR states (i.e. initial curvature of the tubes) and kinematics (i.e. final curvature of the tubes) are independent optimization variables and constraints, respectively. The optimization variables are the tubes' curvatures along a certain direction at the entry point, and then they impose a set of constraints on the endpoint to make the optimal solution satisfies the boundary conditions. The controller performances were evaluated in reaching 3-D points and following linear and circular trajectories, comparing it with the state-of-the-art Jacobian-based and Damped-Least-Squares(DLS) controllers. They found that the MPC controller in point reaching task has a mean error as a percentage of the robot length of 0.03%. While in Fig. 2.3 it is evident that Jacobian and DLS methods were not able to follow a circular path, they skipped part of the path when unstable configurations were reached. Furthermore, they found that the optimal K time horizon length for future trajectories is 5. Root mean square error and standard deviation are described in Fig. 2.3.

Learning based

The complexity of CTR's tube interaction results in a path dependence behaviour of them increasing the difficulty to model physical phenomena (like friction, shear, plastic deformation); and this is what makes CTR's model solving a computationally demanding procedure. Therefore model-based control strategies could not fit real-time applications. This is why research in the last years is focusing on data-driven or learning-based methods for CTRs control. The main advantage of these methods is learning to solve the kinematics by exploiting a certain amount of collected data and then using in real-time this knowledge, usually with higher accuracy and less time-consuming resources.
Many Machine Learning methods have been explored for CTRs control starting from Multilayer Perceptron (MLP) to Locally Weighted Projection Regression (LWPR). In one of the first works with MLP [24], a single hidden layer MLP was used to solve FK and two hidden layers one to solve IK. A mechanics model was exploited to create a dataset made of tube rotation/translation and tip pose. Then the loss function that guided training was based on the mean square error at the tip of the robot. No hardware experimental evaluation was performed.
Another interesting contribution was from Grassman et al. [25]. They introduce a complete approximation approach of the FK and IK of CTRs based on neural networks, adding a novel trigonometric joint description for learning purposes. In this case a cylindrical representation of CTR's joints was introduced defining γ_i for each tube as:

$$\{\gamma_i\} = \{\gamma_{1,i}, \gamma_{2,i}, \gamma_{3,i}\} = \{cos(\alpha_i), sin(\alpha_i), \beta_i\} \tag{2.4}$$

where α_i is the i^{th} tube rotation and β_i is the i^{th} tube translation. In this way all entries of γ_i are elements of $\mathbb{R}$ which is important considering that an artificial neural network with a real activation function expects real input values. A feedforward network with a Rectified Linear Unit(ReLU) activation function was used. The training and test set were a set of tip poses measured through an electromagnetic tracking system, where for each sample the measurement was repeated five times in order to reduce the error due to the tracking system. For the FK an averaged result of 10 networks, whose input layers were of 2.4 type, was computed. The approximation error of this set of networks was $e_x = 2.23 \pm 0.25mm$ and $e_\theta = 1.04 \pm 0.08°$ with respect to a model based approach which achieved $e_x = 27.4 \pm 5.1mm$ and $e_\theta = 88.3 \pm 37.3°$ which correspond to 1, where e_x was the position error while e_θ the orientation error. In the case of the IK inputs and outputs are flipped, so the network accepts tips poses while the outputs are joint values of 2.4 type. They extended the network to two hidden layers with respect to the FK one, increasing

the accuracy of the network since the IK problem usually is more complex. They achieved an approximation error for the test set of $e_\beta = 4.0 \pm 0.6mm$ and $e_\alpha = 8.21 \pm 0.28°$.
Kuntz et al. [26] analyzed also performances of different kinds of neural networks employed to compute the full shape of CTRs. In particular, these networks took as input each tube configuration of 2.4 type and then parameterize the robot's configuration as:

$$q = (\gamma_1, \gamma_2, ..., \gamma_k) \tag{2.5}$$

and as output a set of coefficients useful for a set of 5 orthonormal polynomial functions in x,y and z. In this way, three functions x(q,s), y(q,s) and z(q,s) are formed, for example:

$$x(q, s) = len(q) \times (c_{1x}P_1(s) + c_{2x}P_2(s) + ... + c_{5x}(s)), \tag{2.6}$$

where s is an arc length parameter between 0 and 1, len(q) is the total arc length of the robot's backbone at a given q configuration. The shape function is then calculated as:

$$shape(q, s) =< x(q, s), y(q, s), z(q, s) > \tag{2.7}$$

So the network will be evaluated at q configuration giving as output the 15 parameters that will define a space-curve function that can be evaluated at any arc length value. Finally, having the knowledge of robot's radius as a function of arc length, a prediction of the robot's shape in the world is given.
They tested multiple networks from 3 to 7 hidden layers and from 15 to 60 nodes, they also trained each of these networks with simulated data alone, real data alone and pre-trained with simulated data and fine-tuned with real data. Results highlight that both real and simulated+real outperform the simulated-only network across all architectures. Furthermore, simulated+real outperform real data type, underlining the theory that pre-training on large data sets from the related domain and then fine-tuning on a smaller data set from the exact domain of interest allows the model to be more accurate.
Liang et al. [27] work demonstrates how giving a shape type of input to the network for IK resolution results in higher accuracy with respect to tip pose type of input as in [25]. So they tried to solve the Shape-to-Joint(S2J) IK where the shape S is described as a set of m points $p_j = [x_j, y_j, z_j]^T$ equidistant and ordered from the base to the tip of the robot in a fixed reference frame at the robot's base, therefore S is in the form:

$$S = [x_2, y_2, z_2, ..., x_m, y_m, z_m] \in \mathbb{R}^{3(m-1)}. \tag{2.8}$$

While network's output is of the type:

$$q = [\gamma_{1,1}, \gamma_{2,1}, \gamma_{1,2}, \gamma_{2,2}, \gamma_{1,3}, \gamma_{2,3}, \beta_1, \beta_2, \beta_3] \tag{2.9}$$

where $\gamma_{1,i} = cos(\alpha_i)$ and $\gamma_{2,i} = sin(\alpha_i)$, for α and β being joint's rotation and translation. In this way the network architecture was a feedforward neural network with fully connected layers and input dimension of 3(m - 1) since the first point is the base and output dimension of 9. They also proposed a customized loss function in order to improve α_i and β_i learning. This function is composed of a first function based on the cosine distance between the estimated $\hat{\alpha}_i$ and the real rotation angle α_i, whose value is zero when the angles are equal:

$$D_i = 1 - \frac{\gamma_{1,i}\widehat{\gamma_{1,i}} + \gamma_{2,i}\widehat{\gamma_{2,i}}}{\sqrt{\gamma_{1,i}^2 + \gamma_{2,i}^2}\sqrt{\widehat{\gamma_{1,i}^2} + \widehat{\gamma_{2,i}^2}}} \tag{2.10}$$

while the second component is made of the squared error of the translational component multiplied by a constant ω. The entire loss function is of this type:

$$\mathcal{L} = \sum_{n=1}^{3}(D_t + \omega(\beta_t - \hat{\beta}_t)^2) \tag{2.11}$$

They found that this loss function outperforms the common Root Mean Square Loss for this task. The dataset was built calculating the S_k using the physics-based FK model proposed by Rucker et al. [28] for each q_k sample. They found that errors e_{α_i} and e_{β_i} are consistent for $m < 5$ and remain constant with more points. In particular, for $m = 20$ the median e_{α_i} are 1.13°, 1.01°, 0.62°, and median e_{β_i} are 0.39mm, 0.82mm, 0.72mm, for i= 1,2,3. Looking also at a comparison with a numerical approach solution for S2J-IK the learning based S2J-IK has lower e_{α_i} and e_{β_i}, but in both case error decrease and remain relatively constant for $m > 10$, proving that S2J-IK is more efficient then tip to joint IK. What the previous model does not account for is that Young's modulus or Poisson's ratio for a particular tube may contain inaccuracies, also precurvature, stiffness and length estimation for each tube can contain errors. This can lead to incorrect tip pose estimation even using the most advanced kinematics models. Fagogenis et al. [29] tried to overcome this issue by working on a machine learning approach for adaptively modelling CTR's kinematics based on LWPR.

The LWPR concept is based on a kernel-based regression that predicts tip's pose f(x) for any configuration of the type $x = [\alpha_{i1}, d_{i1}, ..., \alpha_{n1}, d_{n1}]^T \in \mathbb{R}^{2n-2}$ where n is the number of CTR's tubes, $alpha_{n1}$ and d_{n1} are the n-tube's rotation and translation with respect to the first tube; using training data. Configurations close to x provide information for the

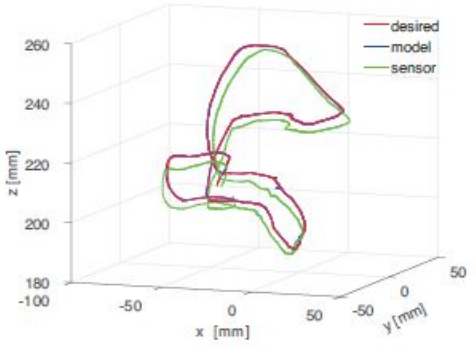
tip pose f(x), to define which x will provide information the kernel will assign a weight w to any pair of robot configurations, all the pairs with $w > \epsilon > 0$ will the define the kernel receptive field(RF). For each receptive field LWPR fits training data with linear model through Partial Least Squares (PLS). Learning the kinematics model means splitting the robot's configuration space in kernels and fitting local models to each kernel RF. So during training a set of parameters that define RF and the local models are updated minimizing a specific cost function C. Parameters for each RF are then update applying Stochastic Gradient Descent on cost C:

$$\theta := \theta - \eta \nabla_\theta C(\theta) \tag{2.12}$$

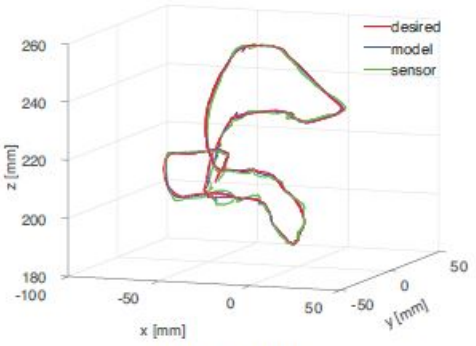
where θ summarizes all the training parameters of LWPR and η is the learning rate. For a given configuration x, LWPR detects all the activated models and the kernel associated to each of the models will assign a weight to each local prediction of the tip's pose. The final estimation of $\hat{y}$ is the weighted sum of all activated model-predictions. The way LWPR incorporates new information is through *Adaptation* concept: when a new set of joint variables and tip pose (x_i, y_i) are provided the algorithm check the activated models(those models whose weight is above a threshold) and let these models update the local parameters through 2.12. This way, given a sample x1, activated models will update local parameters, then for a small displacement Δy in the task space such that the generated x2 is within the same RF, so the prediction in x2 will be improved because models' parameters will be already updated.

For training, they used a dataset of 80 configurations states from which they derived the corresponding end-effector position through a mechanics-based parametric model, the resulting x,y pairs was used to train the LWPR model. Part of this dataset was employed as test dataset, the RMSE of the model prediction with respect to the test dataset was evaluated (Fig. 2.5). Where for orientation they used a relative angle between predicted and measured tangent vectors at the robot's tip.

Then, the model was tested on a three tubes CTR performing a trajectory-following task. The current robot's tip pose was estimated by the trained kinematics model. From Fig. 2.4 is evident that with *Adaptation* turned on, model-predicted path and sensed path rapidly converge to the desired path. Here a teleoperated path was conducted showing an high online adaptation despite the convoluted pathway.



(a) Adaptation off



(b) Adaptation on

Figure 2.4: Teleoperated LWPR test

	x	y	z	orientation
RMSE	0.97mm	0.87mm	0.41mm	1.89 deg
nRMSE [100%]	0.68	0.58	0.52	1.05

Figure 2.5: RMSE of LWPR trained model

Hybrid model-based and model-free control

Most of these learning-based methods are used to control the robot without any information about the environment or the external forces, mainly because it's difficult to gather a large dataset with this kind of information. On the other hand, model-based controllers do not consider realistic environments. Hence, these methods can lead to tissues damages as the robot will traverse constrained environments or couldn't be able to exercise the forces required for tissue manipulation. This is why Thamo et al. [30] decided to develop a hybrid closed-loop data-driven controller initialized by the nominal kinematic model of the robot. The proposed method was designed to control a CTR with unknown dynamics and in contact with an unknown environment.
The model-based control is achieved by solving the Jacobian of the robot, which is numerically estimated using :

$$J_M = \frac{\Delta x}{\Delta q} = \begin{bmatrix} x^T(q + \frac{\Delta q_1}{2} e_1) - x^T(q - \frac{\Delta q_1}{2} e1) \\ \vdots \\ x^T(q + \frac{\Delta q_6}{2} e_6) - x^T(q - \frac{\Delta q_6}{2} e6) \end{bmatrix}, \tag{2.13}$$

where x is a 3x1 vector defining the cartesian coordinates of the robot's tip and $q = [\beta_1, \beta_2, \beta_3, \alpha_1, \alpha_2, \alpha_3]$ is the vector of joint inputs and e_i is the i^{th} unit vector of canonical basis of joint space. This Jacobian is referred to a three tubes CTR whose end-effector is the tip of the innermost tube and β_i and α_i refer to proximal translation and rotation of the i^{th} tube. The Jacobian is calculated by linearising the solution, leading to low accuracy. Data-driven Jacobian could overcome this problem, it chooses a model that compensates for disturbances and external forces updating the elements of the Jacobian in real-time. The algorithm requires the measurements of the robot's tip position x and joint inputs q and updates the Jacobian following this law:

$$\hat{J}_D^{k+1} = \hat{J}_D^k + \mathcal{X} \frac{\Delta x^k - \hat{J}_D^k \Delta q^k}{(\Delta q^k)^T (\Delta q^k)} (\Delta q^k)^T, \tag{2.14}$$

where $\hat{J}^{k+1}$ is the estimated Jacobian matrix at the sample time k+1, Δx^k the displacement of the robot's tip at sample time k, Δq^k a vector of joint input change and $\mathcal{X}$ is the learning rate. The main drawback is the long learning time if the robot is interacting with an unknown environment and if bad initialization happens.
To overcome both models' issues they combine the two ideas. Therefore, the calculated model-based Jacobian (using (2.13)) is used as a weighted initial guess for the data-driven

method (2.14). The novel hybrid Jacobian $\hat{J}_H$ will be:

$$\hat{J}_H^{k+1} = e^{-\lambda_1 k} J_M^k + \frac{1 - e^{-\lambda_2 k}}{1 + e^{-\lambda_2 k}} [\hat{J}_H^k + \mathcal{X} \frac{\Delta x^k - \hat{J}_H^k \Delta q^k}{(\Delta q^k)^T (\Delta q^k)} (\Delta q^k)^T], \tag{2.15}$$

where $\hat{J}_H$ is the data-driven while J_M the model-based Jacobian, λ_1 and λ_2 are the weighting factors. This way at the first time sample $\hat{J}_H^k$ is set to zero and so only the model-based Jacobian will update $\hat{J}_H^{k+1}$. In the next times model-based Jacobian contribution will decrease exponentially and data-driven Jacobian one will increase exponentially. The controller will find an optimal combination of joint values q useful to achieve a desired cartesian position x_d, a pseudo-inverse of the hybrid Jacobian will be computed to find those values. The IK follows this equation:

$$\dot{q} = \hat{J_H^{\dagger}} [\dot{x} + K_P e] \tag{2.16}$$

where $e = x_d - x$ is the error between desired and actual tip's position.
Model-based, data-driven and hybrid controllers were tested on a simulated CTR environment in three different kind of scenarios. The most complex one simulated the CTR robot in contact with a soft tissue without any a priori knowledge of the tissue's mechanical features, trying to represent a tissue ablation procedure where the robot should apply pressure and move on the tissue following a certain path. From Fig. 2.6 it's evident that has some difficulties when it approaches the tissue as the desired trajectory is changing direction, but then it recovers quickly following the path with more accuracy then the model-based method which is not able to impress any force as it remains close to the tissue surface. The data-driven model is not represented as it fails to follow the trajectory and the error is too large.

2.2. Reinforcement Learning background

Among robots' control techniques, Reinforcement Learning (RL) has had an emergence as popular for solving complex control tasks. Moreover, model-free learning methods became popular among CTRs' control strategies being able to overcome tube interaction complexity and modelling for kinematics, as already deepened previously in this chapter. In this work a model-free RL approach is experienced.
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.

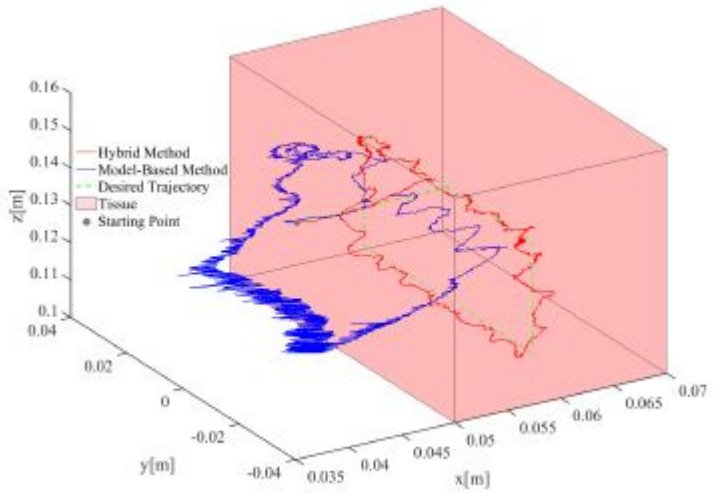


Figure 2.6: Tissue ablation simulation

RL is often used in scenarios where there is no direct supervision, and the agent must learn by trial and error. For example, a game-playing agent may have to learn the rules of the game and the strategies that lead to winning, without having access to an explicit set of labelled examples. In this way, RL differs from supervised learning where a labelled dataset is needed to learn a policy and also from unsupervised learning where the model learns to identify hidden patterns in unlabeled data; in RL the agent goal is to learn a specific task interacting in an unknown environment.
RL can be broadly divided into two categories: model-based and model-free. In model-based RL, the agent learns a model of the environment, such as a probabilistic model of the state transitions and rewards. It then uses this model to predict the consequences of taking a particular action in a particular state. In model-free RL, 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 and this is the case of the current work.
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:

- An **Agent**, the learner and decision maker.
- An **Environment**, physical world where the agent interacts.
- A **State** S_t, the current condition of the agent in the environment.
- An **Action** A_t, which changes the agent state in the environment.

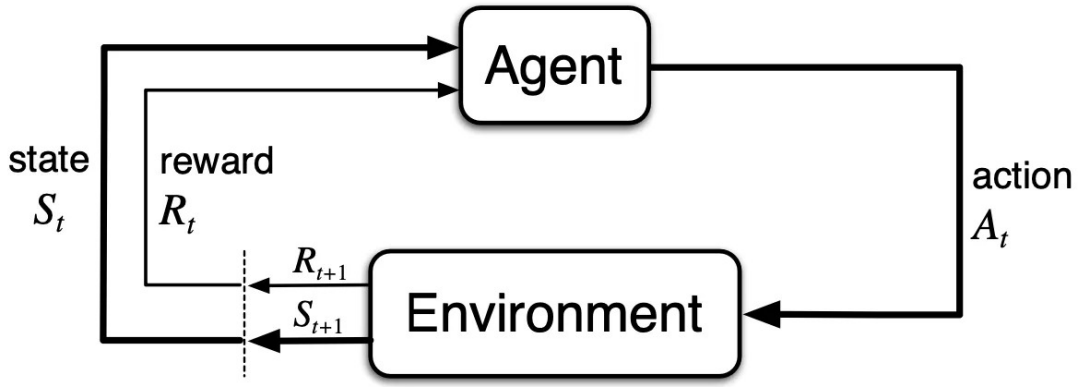


Figure 2.7: Action-Reward feedback loop of a generic RL model

- A **Reward** R_t, the feedback that discriminate actions effects.

As visualized in Figure. 2.7 and described by Barto et al. [31] at each time step t, the agent observes the current state of the environment s_t and selects an action a_t based on its policy π:

$$\pi(a_t|s_t) \tag{2.17}$$

The action is then executed, and the environment transitions to a new state s_{t+1} and emits a reward signal r_{t+1}. The goal of the agent is to learn a policy that maximizes the *expected cumulative reward* over time, where the return G_t is defined as some specific function of the reward sequence. In the simplest case, the return is just the sum of the rewards:

$$G_t = R_{t+1} + Rt + 2 + R_{t+3} + ... + R_T, \tag{2.18}$$

where T is the final time step. This approach is used in applications where agent-environment interaction naturally ends in a terminal state which defines an *episode*, followed by a reset to a starting state. In this case, we are talking about *episodic tasks*. On the other hand, *continuing tasks* are that kind of agent-environment interactions where there is no *episode* definition. Thus, the additional concept of *discounting* is needed where the agent selects actions in order to maximize the sum of the discounted rewards. The *discounted return* is defined as follow:

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{2.19}$$

where $0 \leq \gamma \leq 1$ is a parameter called *discount rate*. The discount rate influences the present value of future rewards. If $\gamma < 1$ the infinite sum in eq. 2.19) has a finite value. If $\gamma = 0$ the agent is "myopic" and gives more weight to immediate rewards. As γ approaches

1 the agent is more farsighted so takes into account more strongly future rewards.
One way to learn a policy in most of RL algorithms involves estimating a *value function*. The value function measures how good it is to perform a given action at a given state. It estimates the expected cumulative reward starting from a particular state and following a particular policy. Where formally if the agent is following a policy π at time t, then $\pi(a|s)$ is the probability that $A_t = a$ if $S_t = s$.
There are two types of value functions: state-value functions and action-value functions. The *state-value function* $v_\pi(s)$ estimates the expected cumulative reward starting from state s and following the current policy π.

$$v_\pi(s) = E_\pi[G_t|S_t = s] = E_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}|S_t = s] \tag{2.20}$$

where $E_\pi[\cdot]$ is the expected value of a variable following the policy π. While, the *action-value* function $q_\pi(s, a)$ estimates the expected cumulative reward starting from state s, taking action a, and following the current policy π:

$$q_\pi(s, a) = E_\pi[G_t|S_t = s, A_t = a] = E_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}|S_t = s, A_t = a]. \tag{2.21}$$

For finite MDPs solving an RL task means achieving an optimal policy, essentially a policy whose expected return is greater than the other policies for all states. Optimal policies share the same optimal state-value function, v_*, defined as:

$$v_*(s) = \max_\pi \ v_\pi(s), \tag{2.22}$$

but also the same optimal action-value function, q_*, defined as:

$$q_*(s, a) = \max_\pi \ q_\pi(s, a). \tag{2.23}$$

The optimal policy can be learned using a variety of algorithms, each with its own strengths and weaknesses. RL algorithms can be generically devided into value-based and policy-based methods. The former learn a value function, mapping each state-action pair to a value, then selecting actions with the highest value. The latter directly optimize a policy without using value functions, a score function is used to compute how good a policy is.

VALUE-BASED METHODS

Depending on state and action space dimension these algorithms can be differentiated in tabular or approximate solution methods.
For *tabular solution methods* state and action space are small enough for the value function to be represented as arrays or tables. One of the tubular solution methods is the *Dynamic Programming*(DP) [32] set of algorithms that compute the optimal policy using a perfect model of the environment.
Instead, in *Monte Carlo*(MC) [31] algorithms there is no assumption of complete knowledge of the environment, they require a complete episode of experience data to update the policy. Without a model is particularly useful to estimate the values of state-action pairs, q_*. Then, a policy evaluation followed by a policy improvement is repeated in loop until an optimal value function and policy are obtained. The last family of tabular algorithms are *Temporal-Difference*(TD) methods. TD is a combination of MC idea of learning from raw experiences without the knowledge of the environment model and DP method to update estimates based also on other learned estimates without waiting for a final outcome(bootstrap). TD algorithms are divided into *on-policy* and *off-policy* methods. In on-policy methods the evaluated or improved policy is the same used to make decisions, whereas off-policy ones use a policy to be evaluated or improved and one to generate data. The off-policy version of TD is the *Q-learning* [33] algorithm, which is one of the most important among value-based algorithms. In this case, the learned state-value function Q, approximates directly the optimal state-value function q_*, independent of the policy being followed. The policy has the role to choose the state-action pair to be visited and updated. The Q-learning algorithm will be of this type:

Algorithm 2.1 Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\epsilon > 0$
Initialize $Q(s, a)$, for all $s \in S^+$, $a \in A(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

1: Loop for each episode:
2: Initialize S
3: Loop for each step of episode:
4: Choose A from S using policy derived from Q (e.g., ϵ-greedy)
5: Take action A, observe R, S$^{'}$
6: $Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma max_a Q(S^{'}, a) - Q(S, A)]$
7: $S \leftarrow S^{'}$
8: until S is terminal

Under the assumption that all pairs continue to be updated, Q converge with probability 1 to q_*.
The motivation that moves to the on/off policy differentiation lies in the critical trade-off between *exploration* of the environment and *exploitation* of the gained knowledge. The introduction of off-policy methods like Q-learning should give more flexibility to this trade-off, despite sometimes slowing learning.

POLICY-BASED METHODS

So far value-based methods have been analyzed that learn values of action and then select actions based on their estimated action values, *policy-based methods* instead learn a policy without consulting a value function. Policy-based methods directly search for a policy that maximizes the expected cumulative reward. Essentially, the main difference with value-based methods is that in this case the optimal policy is found by training directly the policy, while in value-based cases finding an optimal value function leads to having an optimal policy.
The most important subclass of policy-based methods are policy-gradient methods. Dealing with these methods the idea is to parameterize the policy, for instance using a neural network π_θ where θ are the weights of this network. This policy will output $\pi(a|s,\theta) = Pr(A_t = a|S_t = s|\theta_t = \theta)$ which is the probability for the action a to be selected in state s, at time t and with parameters θ. The aim of this kind of methods is to use gradient ascent to update the parameters of the policy function in the direction that maximizes the expected cumulative reward. So the general policy gradient methods update is:

$$\theta_{t+1} = \theta_t + \alpha\widehat{\nabla J(\theta_t)}. \tag{2.24}$$

where $\widehat{\nabla J(\theta_t)}$ is the gradient of an objective function $J(\theta)$, that is the expected cumulative reward, and is computed according to the *policy gradient theorem*, which states the gradient proportionality to the product of the *advantage function*:

$$A(s_t, a_t) = Q(s_t, a_t) - V(s_t) \tag{2.25}$$

and the gradient of the log-probability of the selected action. In this frame REINFORCE [34] is an important method to be analysed, where equation 2.24 become:

$$\theta_{t+1} = \theta_t + \alpha G_t \frac{\nabla\pi(A_t|S_t,\theta_t)}{\pi(A_t|S_t,\theta_t)} \tag{2.26}$$

in this way each update is proportional to the gradient of probability of taking the action actually taken, updating the parameter toward the direction that most increases the probability of taking that action weighted by the return value G_t. On the other hand, the inverse proportionality to the action probability makes more probable action not in advantage with respect to others. In eq.2.26 the fractional vector is an extended form of the most used $\nabla ln\pi(A_t|S_t, \theta_t)$.

ACTOR-CRITIC METHODS

A combination of value-based methods and policy gradient based methods is the family of *Actor-Critic*(AC) methods.
They are based on two networks: the actor and the critic. The actor decides which action should be taken and the critic judges the goodness of that action and how to adjust it. The actor learns in a policy gradient way, while the critic computes the value function. From Fig. 2.8 can be deduced how the agent selects an action in a certain state using the policy π_θ from the actor-network, generating a return. The critic network will use this return to compute the Q-value of that action taken in that state, which in turn is useful for actor gradient evaluation and policy parameter, θ, update.

DEEP REINFORCEMENT LEARNING

A notable and very popular method to approximate the value function instead of using arrays or tables is the introduction of neural networks. DRL allows for high-dimensional states and actions that are usually a limiting factor for classic RL methods.
One popular example of DRL is *Deep Q-Learning*(DQL) [35]. In this case, the neural network will approximate, given a state, the Q-values for each possible action at that state. Therefore, the Deep Q-network will take as input a state and will output a vector of Q-values for each possible action at that state. Then after the evaluation, a policy that selects which action to take according to Q-values is needed. Obviously, at the beginning, the network estimation will be very bad, but during training the network's weights will be updated to achieve a better estimation.
In DQL the weights update is done using a stochastic gradient descent of a loss function. The loss function is the difference between the predicted Q-value and the target Q-value:

$$L(\theta_t) = E(r_t + \gamma \underset{a'}{max} Q(s_t, a', \theta_t) - Q(s_t, a_t, \theta_t))^2, \tag{2.27}$$

where θ are the weights of the neural network, and the first two terms represent the

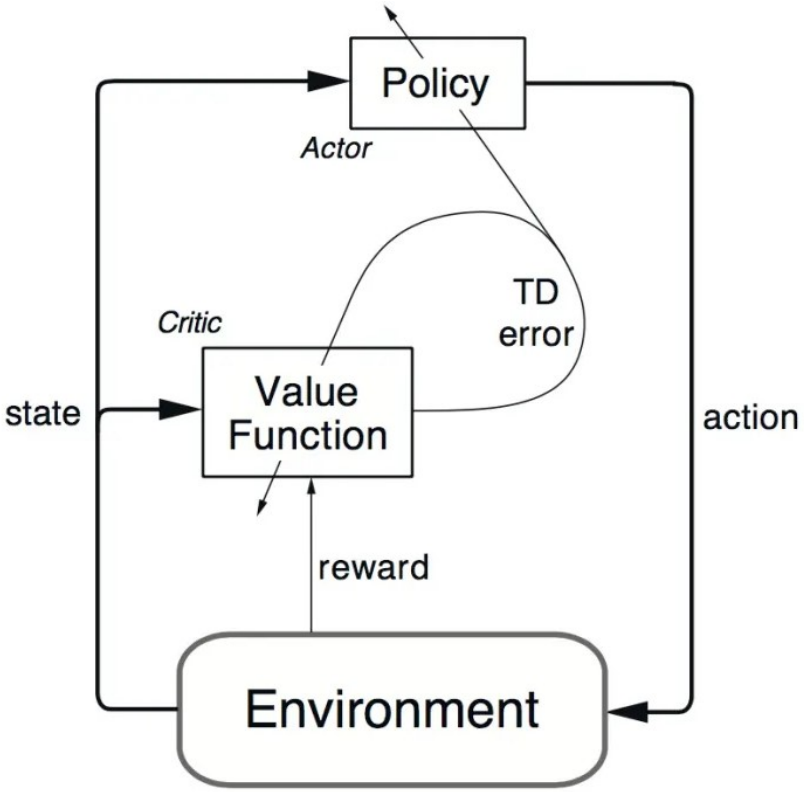


Figure 2.8: Actor-Critic architecture. At each timestep t, the policy selects an action A_t in the state S_t. The critic takes A_t and S_t as input and computes the Q-value for that pair. A new state S_{t+1} and return R_{t+1} will be generated. The actor updates the policy with the Q-value and produces a new action A_{t+1}, the critic then updates its network.

$$A(s,a) = Q(s,a) - V(s)$$

$$Q(s,a) = r + \gamma V(s')$$

$$A(s,a) = r + \gamma V(s') - V(s)$$

TD Error

Figure 2.9: Advantage function

optimal Q-value. Then, the DQL algorithm follows two phases:

- sampling, actions were performed and observed experiences stored in a replay memory
- training, a small batch of experiences was randomly selected and a gradient descend update step was executed.

2.2.1. Advantage Actor-Critic

One of the selected methods in this work is *Advantage Actor-Critic*(A2C) [36] algorithm. The idea is to stabilize further the Actor-Critic learning using the advantage function (A, in eq.2.25) as critic instead of the action value function (Q-value). The advantage function calculates how better taking that action at a state is compared to the average value of the state, which is essentially the extra reward obtained if this action is taken at that state compared to the mean reward obtained at that state. In this way:

- If $A(s,a) > 0$: the gradient is pushed in that direction
- If $A(s,a) < 0$: the gradient is pushed in the opposite direction.

Thus, two value functions are needed for advantage function calculation, Q(s,a) and V(s,a). Fortunately, the TD error could be used as a good estimator of the advantage function as see in Fig. 2.9.

2.2.2. Proximal Policy Optimization

Another exploited method in this thesis is *Proximal Policy Optimization* (PPO) [37], an on-policy policy-gradient method. PPO is considered as the state of the art RL algorithm.

The main idea behind PPO is to improve training stability by limiting the changes made to the policy after each update. This theory is supported by two main reasons: smaller updates during training are more likely to converge to an optimal solution and a too big change can result in getting a bad policy and no possibility to recover from that.
Recalling the main objective to optimize in policy gradient methods:

$$L(\theta) = E_t[log\pi_\theta(a_t|s_t) * A_t] \tag{2.28}$$

where A_t is the advantage function. Taking a gradient ascent on this function should push the agent to choose actions that increase rewards and avoid very bad actions. In order to accomplish the main PPO idea, a new objective function that constrains the policy change in a small range is needed and is the *Clipped surrogate objective function*:

$$L^{CLIP} = \hat{E}_t[min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)]. \tag{2.29}$$

In this function $r_t(\theta)$ represents the probability ratio between the old and the new policy, thus is an estimation of the divergence dimension of the policy changes. The clipped part of the function will constrain this ratio inside a range, avoiding too large policy updates. Therefore, the policy will be updated only when the minimum is the unclipped part, while if the minimum is the clipped part then the gradient will be zero and the weights will not be updated. The advantage function has the role to define the direction of the gradient, and also making the policy updated even if the ratio is outside the range.
The final objective function for PPO Actor-Critic style is:

$$L_t^{CLIP+VF+S}(\theta) = \hat{E}_t[L^{CLIP}(\theta) - c1L_t^{VF}(\theta) - c2S[\pi_\theta](s_t)] \tag{2.30}$$

where L^{VF} is the baseline update, it estimates how good is that state, while S is the entropy argument and it guarantees enough exploration.
In PPO usually there are two threads alternating:

- for N parallel actors the current policy interacts with the environment for T timesteps and calculate the advantage $\hat{A}_t$
- contruct the surrogate loss L on this collected data and optimize it for K epochs

as described in pseudo code 2.2.

Algorithm 2.2 PPO, actor-critic style

for *iteration* $= 1, 2, ...$ **do**
  for *actor* $= 1, 2, ...N$ **do**
    Run policy $\pi_{\theta_{old}}$ in environment for T timesteps
    Compute advantage estimates $\hat{A}_1, ..., \hat{A}_t$
  end for
  Optimize surrogate L wrt θ with K epochs
  $\theta_{old} \leftarrow \theta$
end for

2.2.3. Previous work method

Previous work training were based on *Deep Deterministic Policy Gradient*(DDPG) [38] method. DDPG is an off-policy algorithm that can be used to solve continuous control problems and concurrently learns a Q-function and a policy. As in Q-learning algorithm an optimal action-value function $Q^*(s, a)$ is learned, then in any given state the optimal action a^* can be found:

$$a^*(s) = \underset{a}{argmax} Q^*(s, a) \tag{2.31}$$

Since DDPG is intended for continuous action space problems, calculate $max_a Q^*(s, a)$ is a highly non-trivial problem. On the other hand, $Q^*(s, a)$ is differentiable for a continuous action space, thus a gradient-based learning rule could be exploited.Then DDPG learns an approximator for $Q^*(s, a)$ and one for $a^*(s)$.
The Q-learning part of the algorithm aims to learn an approximator of $Q^*(s, a)$ based on Mean Squared Bellman error (MSBE) minimization, which explains how close a neural network $Q_\phi(s, a)$ is satisfying the Bellman equation. For this scope, a replay buffer is needed for the neural network to approximate $Q^*(s, a)$, and this buffer is a set of previous experiences with the correct dimension. Moreover, another method to stabilize MSBE minimization is to use a target network which is a network, $Q_{\phi_{targ}}$ that lags the first $Q_\phi(s, a)$. Thus, a solution to find a maximum over actions in the target for continuous action space is to exploit a target policy network, $\mu_{\phi_{targ}}$, that computes an action which approximately maximizes $Q_{\phi_{targ}}$.
Dealing with the policy learning part of the algorithm the aim is to learn a policy $\mu_\theta(s)$ that maximizes $Q_\phi(s, a)$ performing a gradient ascent with respect to policy parameters only to solve:

$$\underset{\theta}{max}\ E[Q_\phi(s, \mu_\theta(s))] \tag{2.32}$$

Moreover to improve convergence Iyengar et al. applied Hindsight Experience Replay (HER) [39]. HER is a replay strategy that relabels unsuccessful trajectories to successful ones and stores them in a replay buffer, thus trajectories that does not reach the goal (within some tolerance) are saved in the replay buffer as trajectories to achieve another point in the workspace.
Compared to PPO and A2C, the substantial difference that mostly impacts the policy is that DDPG is an off-policy method while PPO and A2C are both on-policy. This feature has great relevance if the attempt of the RL training is to teach an agent to solve an IK problem. Being off-policy DDPG collects experiences in a replay buffer and then tries to learn a policy from those experiences. On the other hand, on-policy methods learn online, which means that the policy is updated based on current experiences. This affects the agent's capability to solve the IK problem while some training parameters are changing. For instance, if the requested accuracy for targeting tasks increases during training the computation ability of on-policy methods is improved with respect to off-policy ones, thus the number of necessary steps to reach error convergence decreases significantly.
The aim of this work is to prove that these theoretical improvements also transfer to the analysed CTR control problem.

2.3. Thesis objective

As stated above the solutions found in literature for CTRs control have some limitations. Model-based methods could reach a very high accuracy in modelling all the physical phenomena that are involved in CTRs motion, but this will lead to a high computational cost not allowing these methods to be used in real-time applications. On the other hand, learning-based approaches has the potential to increase accuracy and computational speed. Differently from currently machine learning proposed solutions, the proposed DRL solution has the advantage that no dataset is needed to train the network, since experience is gained during the training. This is what moved this thesis analysis towards DRL method. The main concept behind DRL is that an agent will interact with an environment taking different actions which will generate a numerical reward. The agent aims to learn a policy by maximizing the reward. In this case, the agent can collect experiences interacting with a CTR simulation environment. At each episode of the training a new target point inside the workspace will be fixed, the agent will change CTR joint values trying to reach the desired goal and a reward is given based on goal distance. Once trained the agent policy will be evaluated sampling a set of target points and computing the error between reached and desired goal, and giving a certain path estimating agent capacity to

follow it.
The objective of this thesis is to analyse other algorithms and reward functions in order to improve the following performances: policy accuracy, training time, generalization.

3 | Materials and Methods

3.1. Workflow overview

As already deepened, technological improvements in MIS have made these procedures more and more exploited. The tiny incision performed enable to minimize the damages of not interested tissues leading to a reduced risk of postoperative adversities, shortening hospitalization time and convalescence. This is why, nowadays, MIS most of the time is preferred to open surgery. However, this approach significantly confines the workspace, compromising hand-eye surgeon coordination and so extending the learning curve and the duration of the procedure. In this context, Robotic-Assisted instrumentation helps to overcome these manoeuvrability issues; robotic control ensures human tremor cancellation as well as more accurate motions. Coupled with robotic control high dexterity, flexibility and small dimensions for the instrumentation are needed, for this purpose the investigation of CTRs is important.

Interventions like FLC are eligible for CTRs employment thanks to the high DOFs of this device, even in the confined environment, where this procedure is conducted, good manoeuvrability could be reached. But the necessity to keep a correct distance from the tissue and to ablate the right portion of tissue, make the tip position control a crucial factor. The aim of this project is to develop a fast and accurate enough controller that can be used in real-time conditions.

In order to pass all the model-based difficulties, a DRL-based control method has been exploited. DRL-based model can potentially learn a more realistic behaviour since the training phase can be translated on the hardware, representing in this way all the physical phenomena involved. On the other hand, control strategies where a kinematic model of CTR is involved are limited by the computational cost.

The model-free control of CTR is formulated as a Reinforcement Learning problem where the agent interacts in CTR simulation environment. The training episode evolves as see in Fig. 3.2 where a desired cartesian point is selected at the beginning, then the agent selects different joint values at each step and will be rewarded according to a reward function. Different algorithms have been tested to train the agent, preceded by an hyperparameters

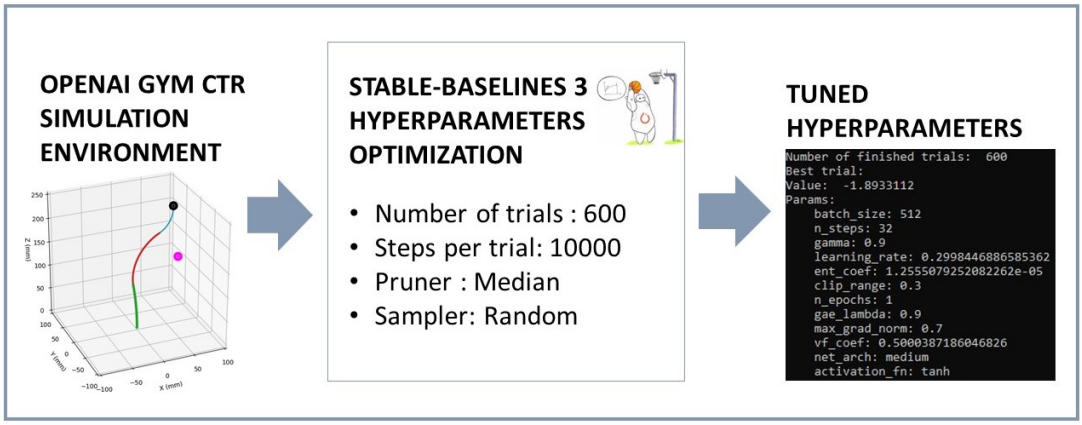


Figure 3.1: Hyperparameters research workflow: for each algorithm and each reward function the simulation environment was run for 600 trials. In every trial a targeting task was simulated and the most rewarded trial was picked as hyperparameters reference. Stable-Baselines 3 framework was involved for this process.

optimization (Fig. 3.1) process for each of them in order to achieve better results. Then the obtained parameters have been used to train the agent as explained in Fig 3.2. Once trained the model has been evaluated based on two different test: targeting and path following.
The trained model could then be tested in a real loop control. For instance the surgeon is performing an FLC procedure via a teleoperated haptic device for Cartesian position control of the CTR tip position. The desired tip position can then be reached through the trained policy which determines the joint values needed to achieve that position.

3.2. CTR model

As this thesis is intended to be a parallel study of Iyengar et al. [40–44] work, the focus will be on the same CTR system. Made of three concentrically arranged tubes with index i going from the innermost to the outermost tube as see in Fig. 3.4, where each tube length is the summation of the curved and straight 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 $\mathbf{q} = [\beta_1, \beta_2, \beta_3, \alpha_1, \alpha_2, \alpha_3]$. The extension constraints due to

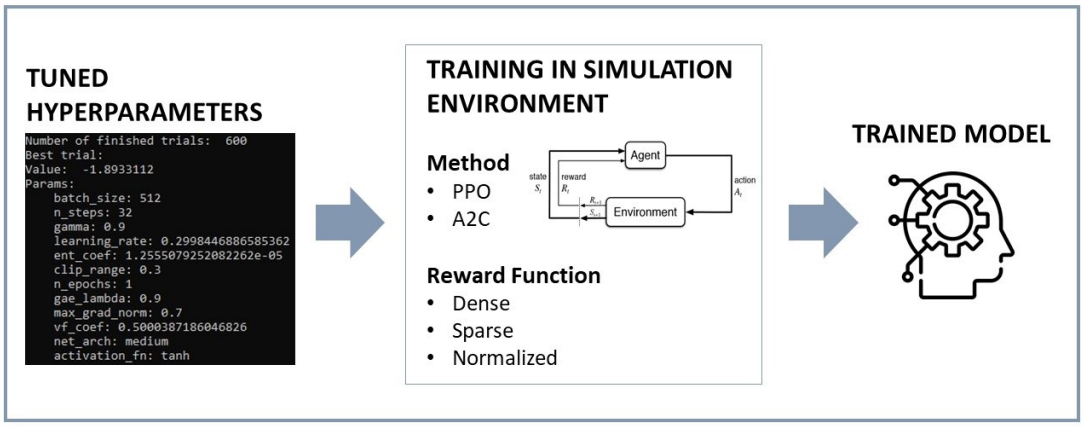


Figure 3.2: Training workflow: the tuned parameters are used to train the corresponding method and reward function. In every episode of the training a DRL problem is solved as a targeting task. The trained agent is saved after the end of the training.

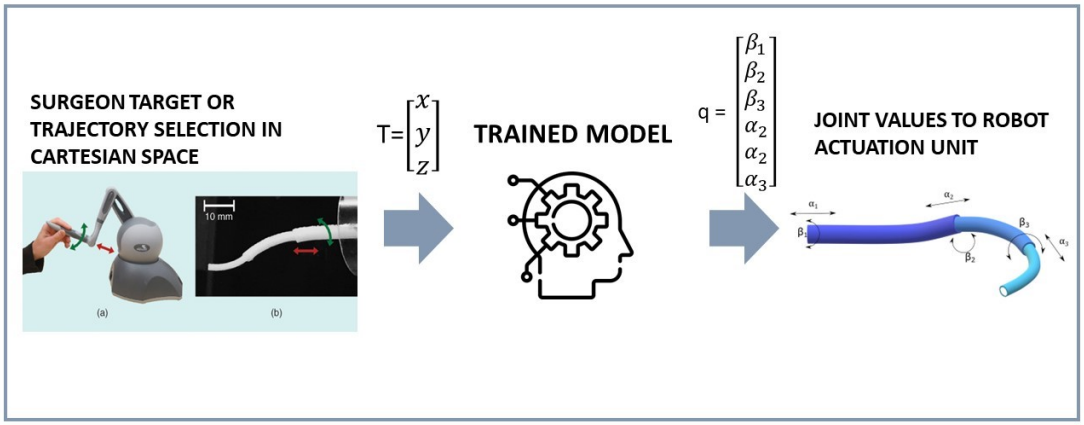


Figure 3.3: Controller workflow: the surgeon selects a cartesian point or trajectory, through a haptic device for instance, the cartesian coordinates become the input for the trained model which computes the joints values to achieve that point or trajectory. The selected joint values are sent to the robot actuation unit.

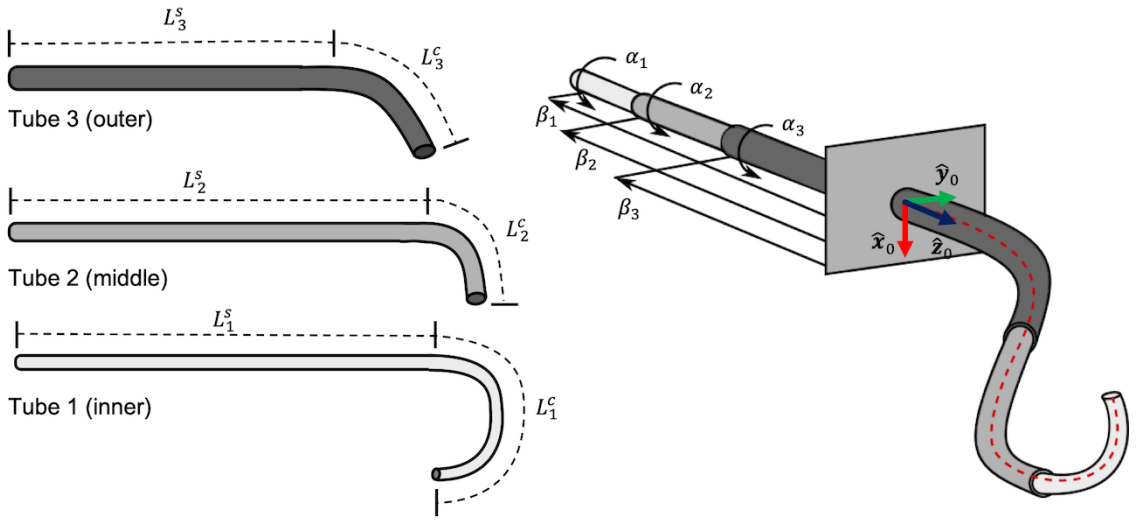


Figure 3.4: CTR system notation

actuation limitations are the following:

$$\begin{aligned} &\beta_i \in [-L_i, 0) \\ &\beta_1 \leq \beta_1 \leq \beta_1 \leq 0 \\ &0 \leq L_3 + \beta_3 \leq L_2 + \beta_2 \leq L_1 + \beta_1 \end{aligned} \tag{3.1}$$

Where the zero position is defined based on the reference frame represented in Fig 3.4. Usually, in real systems, this point is a port where tubes can fully retract and is used as global origin to define robot shape and end-effector position. While rotation is not constrained, tubes can freely rotate relative to one another.

3.3. Controller development

3.3.1. Concentric Tube Robot control problem in Reinforcement Learning

The CTR control problem in this case is an IK problem and is formulated in RL context as an MDP. 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. The state observations are described as current joint values, current end-effector positions and desired end-effector positions. The reward function is a numerical feedback that penalizes the agent if it moves away from the target point and rewards it if it gets close to the target, within some tolerance. Different type of reward functions has been

tested and will be described further on. As already underlined the proposed method is a model-free strategy since there is no kinematic model involved in the learning process, the agents learns the IK only based on experience. However, a CTR's workspace definition is needed otherwise unreachable points could be selected as target points. In order to do so the following kinematic model is implemented to select points inside of the workspace through a forward kinematic process. Thus the kinematic model is only used to simulate the robotic system and to collect experiences useful for the training process.

CONSTANT-CURVATURE MODEL

The kinematic model exploited to solve the forward kinematic for target selection inside of robot workspace is the Constant-Curvature model [45]. This approach makes multiple simplified assumption based on the principles of beam mechanics to solve the forward kinematic problem: pure bending, no torsional deformation, no friction, no effect of gravitation and external forces. Based on these assumptions the model can compute the resulting centerline curvature in closed form.
The model assumes that CTR's shape is composed of constant curvature segments, it can be expressed as a concatenation of multiple circular arcs. Each circular arc corresponds to one segment i and each segment is described by its length l_i, curvature k_i, and bending plane angle ϕ_i. Each segment is bounded by transition points T, where the component tubes either end or transition from their straight to curved section. The set T of transition points can be defined as algebraic functions of the tube geometry and the translational joint positions β.
Thus, for each segment k_x and k_z can be computed using:

$$\begin{bmatrix} k_x \\ k_y \end{bmatrix} = \frac{1}{\sum_{j=1}^{N_t} EI_j} \sum_{j=1}^{N_t} EI_j k_j \begin{bmatrix} cos\alpha_j \\ sin\alpha_j \end{bmatrix}. \tag{3.2}$$

Which is the formula for the resulting curvature of a single segment composed of several overlapping concentric curved tubes. The formula is generalized over j number of overlapping tubes and computes x and y components of segment curvature considering tubes' axial rotation of an angle *alpha*. The robot shape is defined by the equilibrium plane reached after tubes' rotation, and its component are computed as:

$$\phi = atan2(k_x, k_y) \tag{3.3}$$

$$k = \sqrt{k_x^2 + k_y^2} \tag{3.4}$$

where ϕ is the equilibrium plane angle. Thus this kinematic model computes the overall shape of the CTR system from joint values.

3.3.2. MDP formulation

State. The observation state s_t at timestep t is defined by a trigonometric joints representation γ_i, the Euclidean norm error e_t between current tip position and desired tip position and the current error tolerance $\delta(t)$:

$$\begin{aligned} \gamma_i &= \{cos(\alpha_i), sin(\alpha_i), \beta_i\}, \\ s_t &= \{\gamma_1, \gamma_2, \gamma_3, e_t, \delta(t)\}, \end{aligned} \tag{3.5}$$

Instead, the current error tolerance is a value useful for reward function definition and is updated during training. Three different goal tolerance functions are implemented:

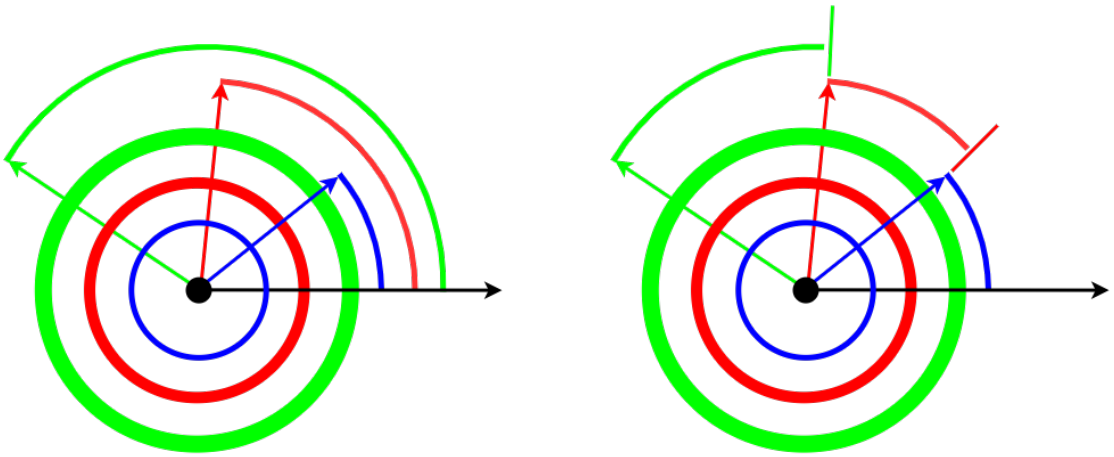
- constant, the tolerance $\delta(t)$ remains constant during training
- linear decay, the tolerance decreases linearly with timesteps. In the equation b is the initial tolerance ,a the slope and N_{ts} the current timestep:

$$\begin{aligned} \delta(t) &= at + b \\ a &= \frac{\delta(N_{ts}) - \delta(0)}{N_{ts}} \\ b &= \delta(0) \end{aligned} \tag{3.6}$$

- exponential decay, the tolerance decreases exponentially with training steps. In the equation a is the initial tolerance while r is the rate decay:

$$\begin{aligned} \delta(t) &= a(1-r)^t \\ a &= \delta(0) \\ r &= 1 - \left(\frac{\delta(N_{ts})}{\delta(0)}\right)^{\frac{1}{N_{ts}}} \end{aligned} \tag{3.7}$$

About joint values two different representations are implemented. The propriocentric one is an absolute representation where joint values are referenced from a base reference as see in Fig. 3.5 from the previous Iyengar et al. work, while in the egocentric mode, the current joint variable is relative to the previous tube. In Fig. 3.5 is visualized only the rotation joint variable, and the same concept is applied to the translational one.



(a) Propriocentric (b) Egocentric

Figure 3.5: Rotation joint representation

Action. At each step the agent selects a set of joint 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:

$$a = (\Delta\beta_1, \Delta\beta_2, \Delta\beta_3, \Delta\alpha_1, \Delta\alpha_2, \Delta\alpha_3) \tag{3.8}$$

The joint constraints expressed in eq. 3.1 are implemented through a check that happens after each action is selected, joint values are clipped in a way that constraints are respected. Moreover, the translational component of joint values is normalized over tube length and extension values that cause tube collisions are avoided.

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. Different reward strategies have been tested starting from the previous work ones that were divided in dense and sparse reward functions.
The sparse rewards strategy in RL aims to give feedbacks for a small handful of states, in this case the reward function is defined as:

$$r = \begin{cases} 0, & e_t \leq \delta \\ -1, & \text{otherwise,} \end{cases} \tag{3.9}$$

where the error is the distance between the achieved end-effector position $\hat{G}$ and the desired one G calculated as Euclidean distance at each timestep t:

$$e_t = ||G - \hat{G}||_2. \tag{3.10}$$

In this case a constant -1 signal is provided for all the states with the exception for those states with an error below the tolerance, thus the important signal is given only for few states and this is why it is called sparse. On the other hand, with a dense reward function the feedback signal is given at each step and the function is defined as:

$$r = \begin{cases} 0, & e_t \leq \delta \\ -e_t, & \text{otherwise,} \end{cases} \tag{3.11}$$

In this way, smaller error distances are more rewarded than larger ones also outside of the current tolerance, providing a more frequent reward signal. The reward feedback is calculated at each timestep and is cumulative through an episode.

3.3.3. Training strategy

The training process is divided into 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 the CTR workspace, then for each step of the episode, the agent selects an action and the reward feedback is assigned. The episode terminates if the number of steps overcomes the maximum episode length or if the currently 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. The simulation environment (Fig. 3.6) is modelled following OpenAI Gym [46] framework. Gym is a standard API for reinforcement learning problems, focusing on episodic tasks, where the agent experience is broken down into a series of episodes. Gym provides an abstraction for the environment, not for the agent. A *reset* function is called at the beginning of each episode, which samples the robot's joint values (compliant with constraints) and computes the forward kinematics based on those values to generate a new target point. Then, for each step of the episode a *step* function is executed, that applies a set of actions (compliant with joint constraints) to joints and computes the reward feedback based on the current end-effector position achieved, also in this case computed through forward kinematics.
In both forward kinematic problem resolutions, the kinematic model is based on tubes' parameters manually measured on a Cad program using scanned images coming from

a three tubes hardware system (provided in collaboration with King's College London, components described in appendix A) and are listed in Tab. 3.1. In addition, stiffness and torsional stiffness are common parameters to all the tubes and correspond to $7.5 \times 10^{10} Nm^{-2}$ and $2.5 \times 10^{10} Nm^{-2}$.

	Inner Tube	**Middle Tube**	**Outer Tube**
Length(mm)	340.36	169.69	72.75
Length Straight(mm)*	250.36	82.19	11.71
Length Curved(mm)	90.00	87.50	61.03
Inn. Diameter(mm)	0.51	0.70	1.15
Out. 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($mm^{-}1$)	24.61	19.12	14.04
Home offset(mm)**	24.61	19.12	14.04

Table 3.1: CTR tube parameters. *There is a straight section of 96.41 mm steel and 153.95 mm nitinol. **Home offset calculated from base plane to limit switch of each extension block

3.3.4. Methods

To train the model Stable Baselines 3(SB3) [47] has been exploited, which is a set of reliable implementations of DRL algorithms. SB3 provides policy networks for images (CnnPolicies), other types of input features (MlpPolicies) and multiple different inputs (MultiInputPolicies). For this work, an Mlp network was used.
SB3 implement these networks as separated into two main part:

- a **feature extractor** network that extract features from high dimensional observations,
- a **fully connected** network that maps features to actions.

Two different network for actor and critic are 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:

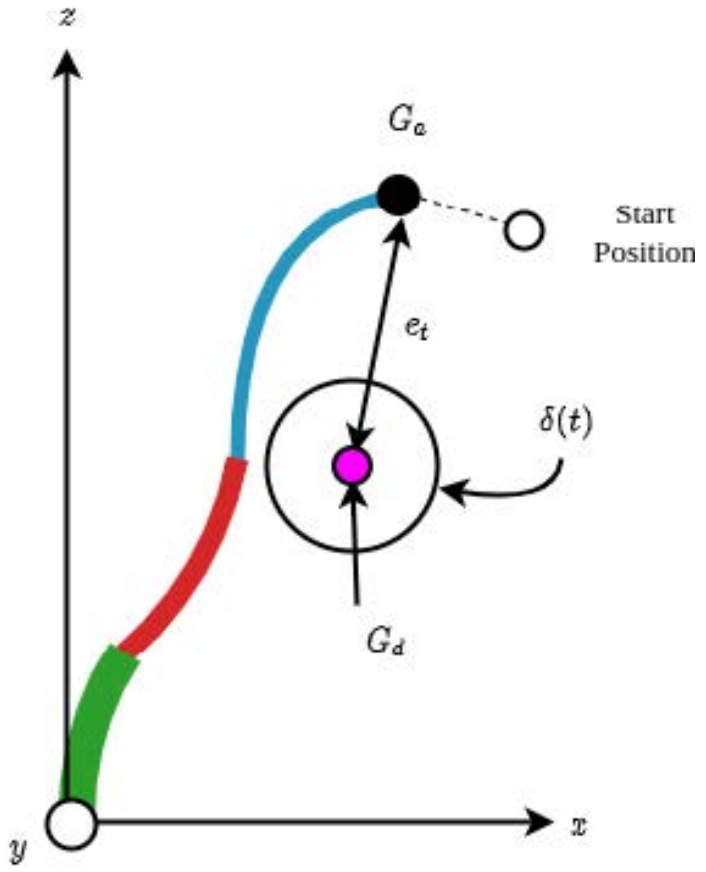


Figure 3.6: CTR simulation environment with starting end-effector position(black dot) and target end-effector position(pink dot)

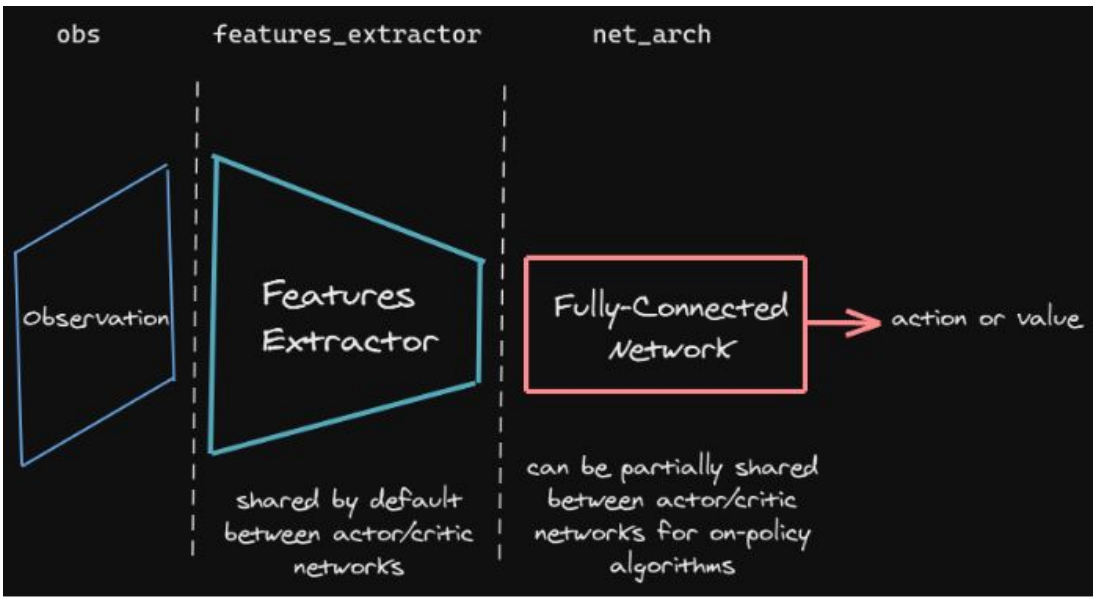


Figure 3.7: Network architecture

- 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 [44] 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.
- Geometrical Jacobian-based controller that is one of the most common method for CTRs control [11, 21, 22], where the exploited closed-form law to steer the CTR is the following:

$$\dot{q_d} = J^{\dagger}[\dot{x_d} + K_p(x_d - x)] \tag{3.12}$$

  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.

3.3.5. Hyperparameter Tuning setup

An essential step for an RL training process is *Hyperparameters Tuning*, because most of these parameters will heavily impact on training performance. The discount factor γ for the discounted reward calculation, network size and activation function, batch size, learning rate are some of the most important. This is why several automatic hyperparameters optimization method have been developed, Zhang et al. [48] demonstrate how hyperparameters optimizer outperform human experts in model-based RL parameters tuning.
The exploited method for hyperparameters research is a framework developed by Akiba et al. [49] 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 is sampled from a group of standard values. The result of this research is the best combination of parameters that comes from the best trial which is the trial 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 the median of intermediate results of previous trials at the same step. The optimization has been performed over 600 trials of 10000 episodes each.
Table 3.2 is the result of hyperparameters tuning for the two selected methods, some of them have been manually adjusted in order to further improve training performance. One

of these parameters with the higher impact over training for both algorithms is *learning rate*. It represents the step size at each iteration while moving toward a minimum of the loss function, in other words is the measure of how much newly information overrides old information. Moreover, PPO performs a stochastic gradient descend update of *batch size* on all the gathered trajectories for a specified number of *epochs*, thus these two factors have been tested as very influential over training performance.

	PPO	**A2C**
Horizon	512	8
Batch size	32	-
Gamma	0.9	0.999
Learning rate	0.0003	0.0003
# epochs	4	-
Gae-lambda	-	0.8
Max gradient norm	-	0.7
Value-function coeff.	0.45	0.87
Actor net size	(256,256)	(64,64)
Critic net size	(256,256)	(64,64)

Table 3.2: Hyperparameters research results

3.3.6. Hardware and Software specifications

Both training and hyperparameters tuning have been performed on Univirsity College London's Myriad cluster, equipped with Intel(R) Xeon(R) Gold 6240 CPU and two NVIDIA Tesla V100s for the node involved. Training results have been visualized by TensorBoard, which is a web application for TensorFlow runs and graphs visualization.

4 | Experimental Setup and Results

4.1. Experimental setup

The tuned hyperparameters have been used as input for the training process. Both the considered 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 [47] implementations. Each algorithm has been tested with longer and shorter trainings, three and one million steps respectively. SB3 implementation supports environment parallelization, thus 1 and 8 parallel environments training have been tested. All the three goal tolerance decay functions (eq.3.6,3.7) 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 performances evaluation have been conducted looking into this 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.

In order to evaluate the trained models two kinds of experiments have been conducted:

- **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's starting pose is resampled at each episode.

- **path following**, the trained policy is used to follow a predefined path. Two kinds of path have been tested: linear and circular. 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.

For both experiments, the exploited simulation environment is the same of the training phase, thus same tube parameters are described in Table 3.1.
These kind of evaluation tests are useful as similar to a real surgical scenario where a surgeon control the end-effector tip position through a haptic device giving Cartesian coordinates as input for the controller.
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:

$$E = \sqrt{(G_{dx} - G_{ax})^2 + (G_{dy} - G_{ay})^2 + (G_{dz} - G_{az})^2} \tag{4.1}$$

and episode length, which defines how fast is the policy to find joint values that perform the required task.
The first comparison is between the two analysed method policies PPO and A2C. A demonstration of better performance for dense reward with respect to sparse reward with PPO policy is described in section 4.2.2 Then PPO's trained policy has been compared with the previous work method and with a Jacobian control method.

4.2. Results

4.2.1. PPO vs A2C

The two proposed methods have been trained for three and one million each and with both sparse and dense reward functions. Training results show that PPO outperforms A2C in both three and one million steps training and with both sparse and dense reward functions. Results for three million steps with dense reward training are shown in Fig. 4.3. PPO converges to a 100% success rate in less than 500000 steps, also reward and episode length show better performance.
A targeting evaluation test has been conducted over 500 points in order to assess the two methods performance in simulation environment. Fig. 4.2 demonstrates how PPO outperform A2C also in evaluation test. The average error distance with its standard deviation and the episode length mean of the targeting test are described in Tab. 4.1, while a graphic visualisation of the same test is represented in Fig. 4.3.

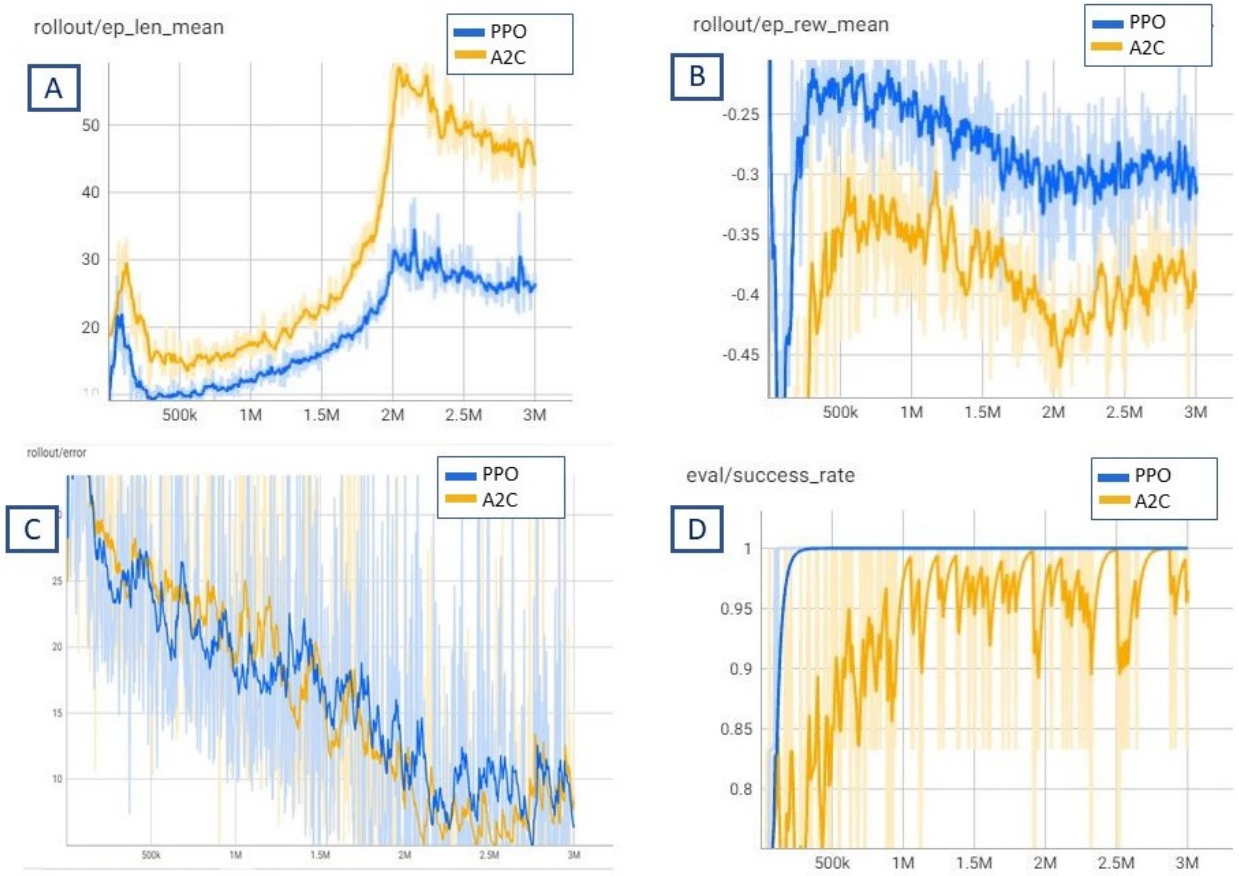


Figure 4.1: Training results, PPO vs A2C. **(A)** episode lenght mean, **(B)** episode reward mean, **(C)** error during training, **(D)** success rate.

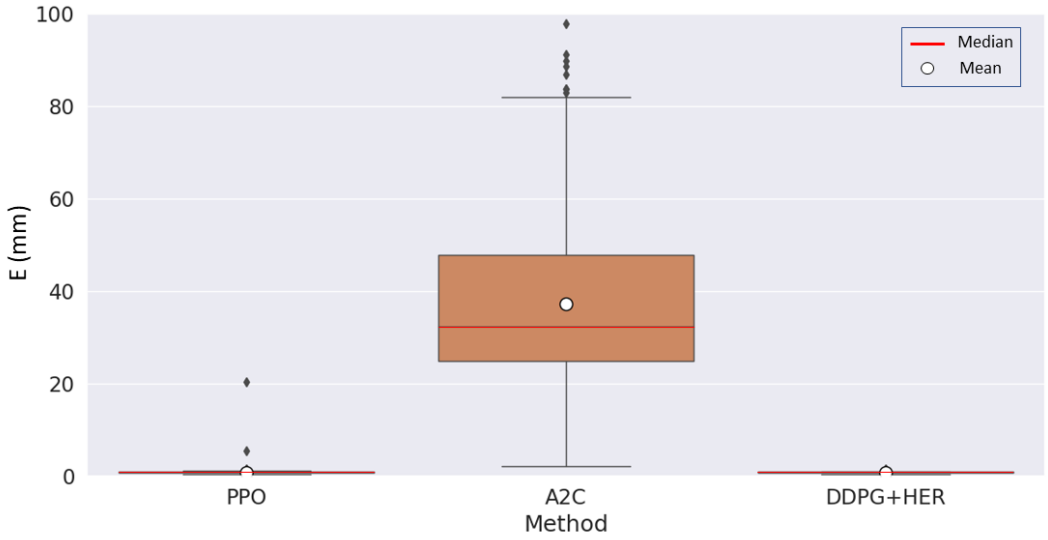


Figure 4.2: Evaluation error boxplot

	Average	Standard deviation	Episode Length mean
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 4.1: Targeting results of the three DRL methods: average and standard deviation of error E, episode length.

4.2.2. Dense vs Sparse reward

As described in Section 3.3.2 the exploited reward signal is of two types: sparse and dense. Training performance for PPO one million steps training in both sparse and dense reward cases are resumed in the success rate parameter shown in Fig. 4.4. Both the trained policy has been evaluated with a targeting experiment over 500 points, the error distance is visualized in Fig. 4.5. 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.

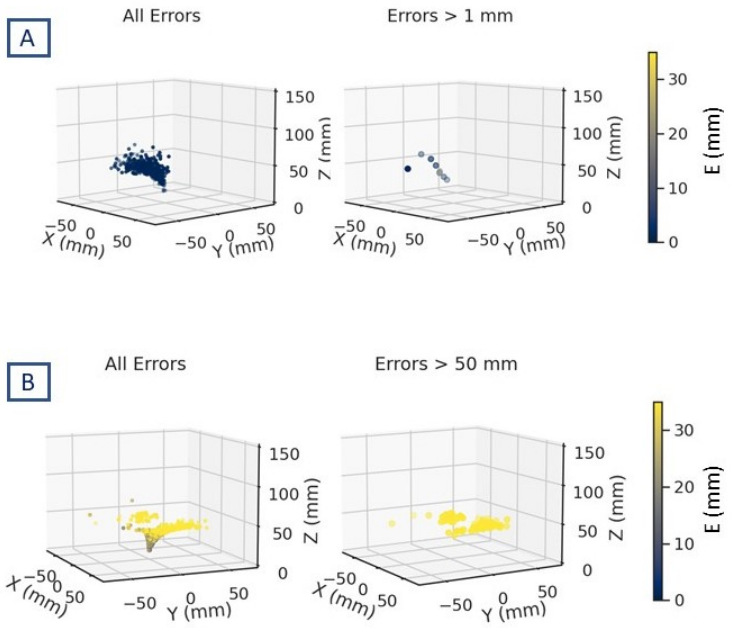


Figure 4.3: Evaluation error distribution, measured in mm. **(A)** PPO, **(B)** A2C

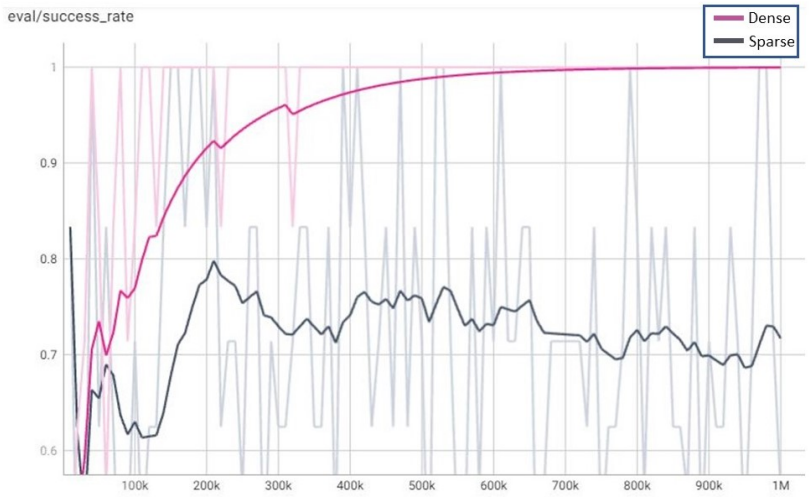
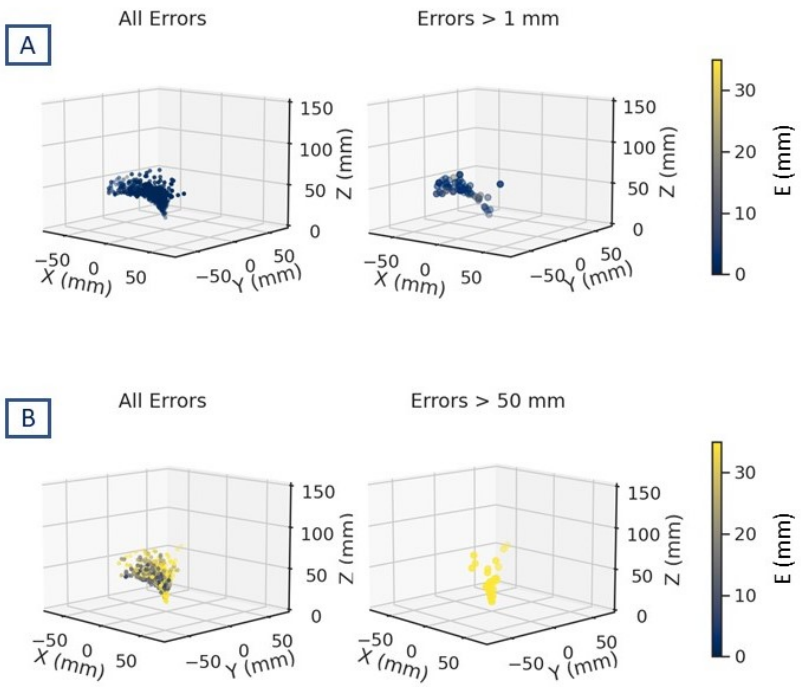


Figure 4.4: Success rate comparison between dense and sparse reward trainings.

(a) Evaluation error boxplot



(b) Evaluation error distribution, measured in mm. **(A)** dense reward function, **(B)** sparse reward function

Figure 4.5

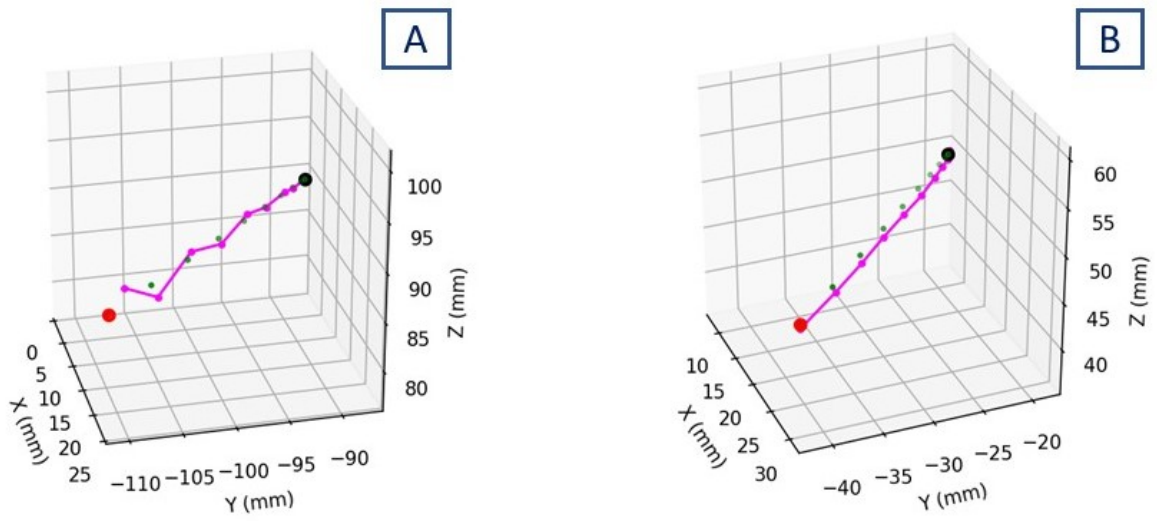


Figure 4.6: PPO vs Jacobian controller evaluation test, the black dot is the starting point of the trajectory and the black one is the ending point while the green dots describe the trajectory. The pink line is the end-effector trajectory. **(A)** is the Jacobian controller trajectory, **(B)** is the PPO trajectory.

4.2.3. PPO vs Jacobian

Once established that PPO method is the most successful of the proposed methods it has been compared with geometric Jacobian controller. The comparison between the two methods has been done on the same system with parameters in Tab. 3.1, K=2I for Jacobian control law and a PPO policy trained with dense reward function for one million steps for DRL method has been employed.
The path-following test has been exploited with 10 different starting points for both linear and circular trajectories, in order to strengthen the robustness of the experiment. In Fig. 4.6 is shown one the most successful of the experienced tests, while in most of the cases the Jacobian controller steered the robot far away from the target trajectory, thus the error metrics of this successful test is not a clear evidence of the bad Jacobian performance and for this reason is not useful to compare them with the other methods.

Figure 4.7: PPO vs DDPG+HER success rate

4.2.4. PPO vs DDPG+HER

In the previous work has been demonstrated that DDPG needs 2 million steps and 19 parallel workers to converge [41], 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. Training results over the same simulation environment are shown in Fig. 4.7, it is evident that PPO policy converges after 600000 steps while DDPG+HER after 1.6 million steps.
DDPG+HER and PPO policies has been evaluated over the same CTR simulation environment with both targeting and path following test and the evaluation error (E) statistics are shown in Fig. 4.2 and Tab. 4.1. The result of path following test with linear and circular trajectories is shown in Fig. 4.8.

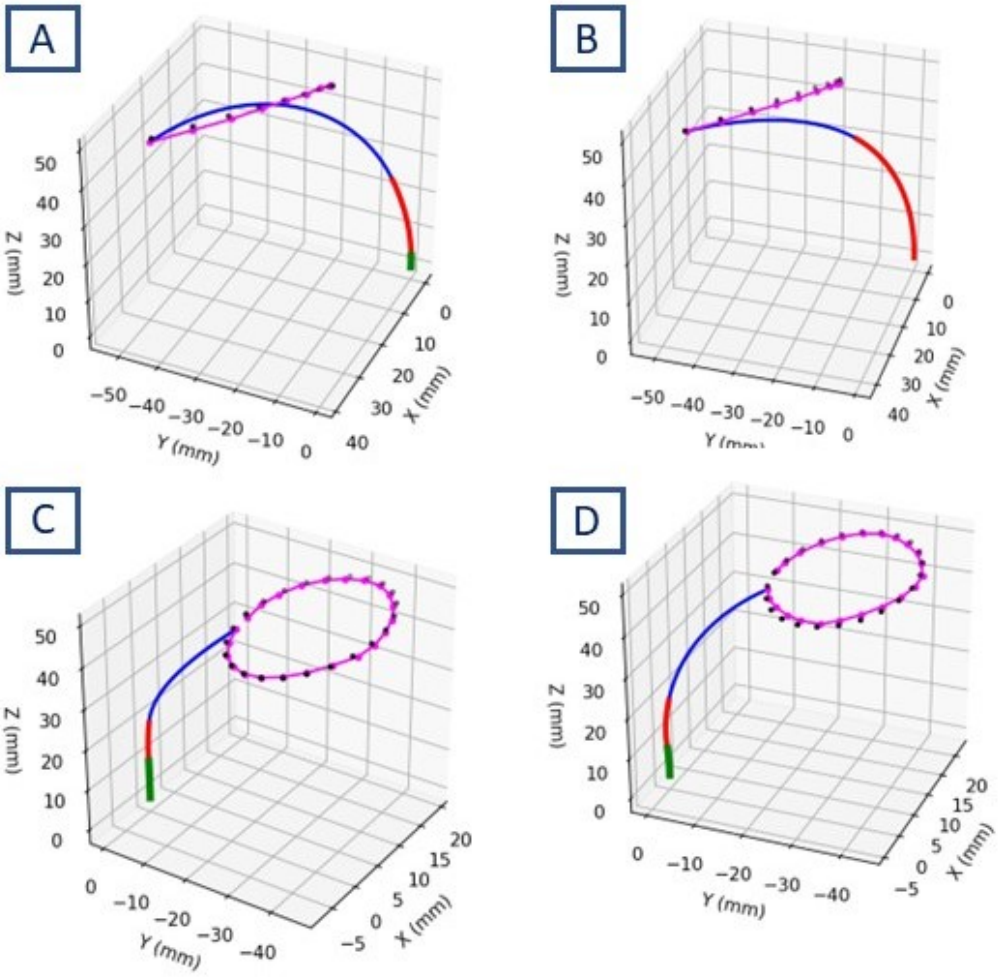


Figure 4.8: Path following test. **(A),(C)** refers to PPO policy evaluation, **(B)(D)** refers to DDPG policy evaluation

4.2.5. Domain Randomization

Another experiment has been conducted to prove PPO method's ability to learn a more general policy. To this purpose a *domain randomization* strategy has been applied to the environment during training. The concept of domain randomization is to sample at each episode a set of parameters with a variation in tubes' parameters inside of a defined range. The chosen range in this experiment was 10% of the selected CTR system (3.1). At each episode a random value of the parameter P inside the range between $P + 0.1 * P$ 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, the average and standard deviation of the distance error with the episode length are listed in Tab. 4.2. While, a graphical visualization of the distance error is represented in Fig. 4.9.

	Average	**Standard deviation**	**Episode Length mean**
10% Randomization	1.62mm	6.8mm	28.6 steps
20% Randomization	1.28mm	2.99mm	26.82 steps
30% Randomization	2.74mm	10.9mm	30.45 steps
40% Randomization	2.39mm	6.4mm	31.31 steps

Table 4.2: Targeting results from Randomization

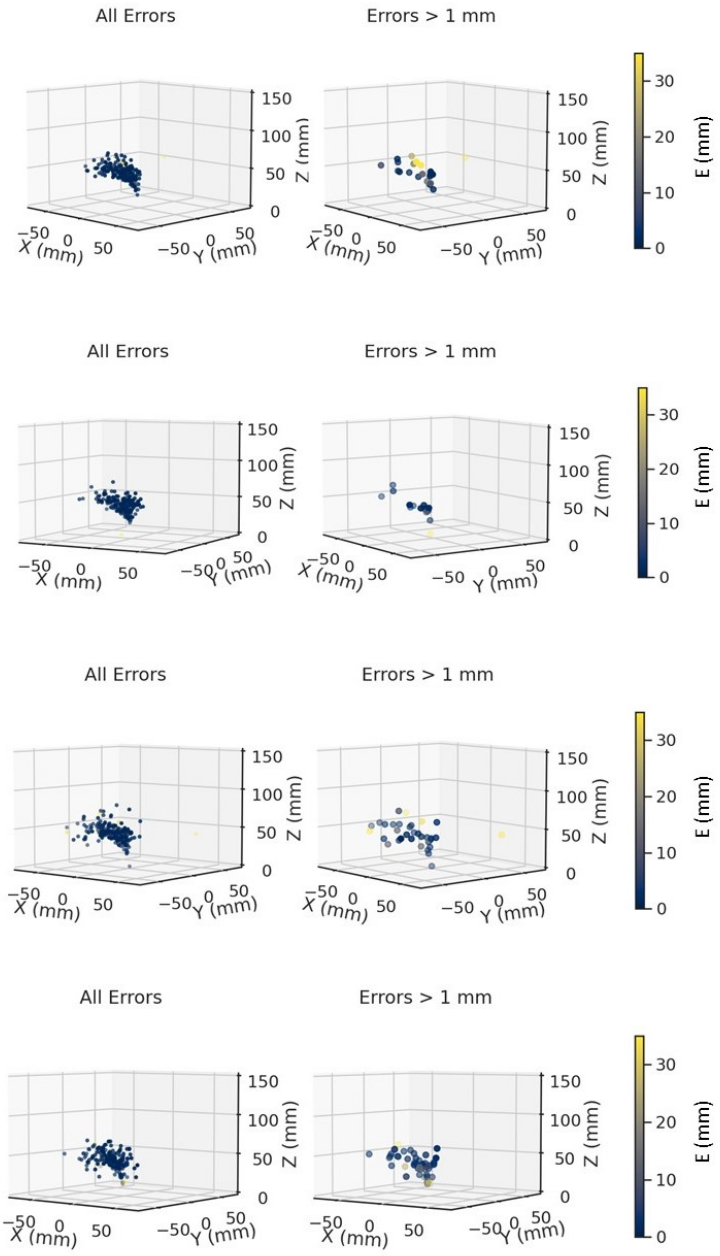


Figure 4.9: Domain randomization error distance visualization, from the top to the bottom the randomization range goes from 10% to 40%

5 | Discussion

In this chapter a discussion about the obtained results will be conducted. The objective of this study was to improve the performance of the DRL-based CTR control method. For this purpose, two new DRL methods with respect to previous work have been tested. From the analysis of the comparison between the two proposed methods, it has been obtained that PPO outperforms A2C for the involved CTR simulation environment. This is firstly observable from training performance where PPO has a higher reward, lower episode length and faster convergence to 100% success rate. And it is confirmed in the targeting evaluation test where PPO reaches a 0.88mm average error distance with respect to the 37.2mm of A2C policy. This behaviour is justifiable by the clipped feature of PPO's loss function (eq.2.30) that keeps the policy changes in a limited range increasing the convergence to an optimal solution probability. On the other side A2C loss function is not clipped resulting in a higher probability of getting a bad policy without the possibility to recover from that.

Once established that PPO is the working method, slightly better performance has been assessed with a linear goal tolerance decay function with respect to the exponential and constant functions. Moreover the reward function has been tested as a very influential factor in training performance, indeed looking at Fig. 4.4 and Fig. 4.5 an important gap between the two policies can be estimated in both training and evaluation results. 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 in order to push the agent in the right direction.

When compared to the geometric Jacobian control strategy, the selected DRL method shows more stable and accurate performance. The trained PPO policy is able to 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. Without an implementation of these constraints (described in 3.1) tubes could collide or the robot reach unfeasible configurations.

In section 4.2.4, is evident a faster and more stable convergence of PPO policy compared

to the previous work one. The advantage of PPO 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. This is again addressable to the PPO's on-policy nature that makes training faster with respect to off-policy methods, the letter in fact uses one policy to be evaluated and updated and one for data generation.
On-policy nature of PPO is also useful to learn a policy in a dynamic environment where parameters change during training. Considering the domain randomization experiment results it can be inferred that the trained agent has learned an inverse kinematic problem solution for CTRs systems with 10% and 20% randomization with an average error calculated from evaluation test that is below 2 mm, while for 30% and 40% the average is larger than 2mm. This result is an initial proof of concept that this method is able to produce some form of policy generalization, despite improvements have to be done. Moreover, the domain randomization technique is proved to be useful for policy translation from simulation to hardware domain [50]. In hardware scenario most of the tube's parameters can be affected by measuring errors, thus randomization helps train a more flexible policy.

6 | Conclusion and Future Developments

In this thesis, a novel DRL-based method for CTRs control has been investigated. The proposed method is intended for specific surgical scenario where no forces are involved at the robot's end-effector, such as ablation procedures like FLC for TTTS. The developed controller could be involved in a closed-loop control where the surgeon selects the point or the path that the robot should follow in the Cartesian space, through a haptic device, and the trained policy solves the inverse kinematics to compute the joint values to send to the actuation unit.

The controller is based on a DRL approach and PPO algorithm has been selected as the most performant method to solve CTRs inverse kinematics. After hyperparameters tuning the model has been trained in the CTR simulation environment for 1 million steps and 8 parallel workers in order to explore properly the workspace. Then the trained agent has been tested in the same environment with targeting and path following tasks. The results showed that PPO is able to solve the kinematic problem with a good accuracy and consistency supported by a low computational reduced computational time when compared with the very common Jacobian approach. Moreover, PPO model has been found to converge faster to a solution during training with respect to previous work DDPG+HER method, and to be promising for policy generalization solving the domain randomization problem.

To further improve the method's accuracy more workers could be involved during training in order to extend the exploration feature of training in particular for the randomization training which is essential for hardware translation. In addition, could be useful to train the PPO agent in a noisy simulation environment, where some noise is introduced to joint values and end-effector position tracking in order to simulate motors encoder errors and end-effector tip position tracking errors. This is also very important for hardware work.

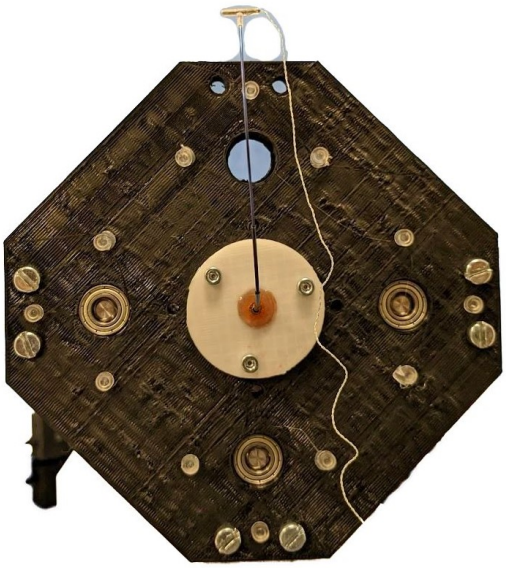
An additional future work can be a test exploited in the attempt to generalize the policy over CTR systems. During the training phase a different CTR system with different tube parameters can be sampled at each episode with the aim to train an agent able to solve

multiple kinematics problems.
Another common issue for CTR devices are the elastic instabilities that can rise from high curvature structure. For instance, *snapping* is a phenomena that happens when tube bending and twisting leads to the elastic potential energy storage followed by the sudden release of it when unstable configurations are reached and the resulting loss of robot control. For this purpose different design [51–53], kinematic models [54] and path planning [55] strategies tried to include this issue in order to improve CTR stability. Dealing with the proposed DRL control method a future work idea could include the unstable robot configurations, that lead to snapping, as penalizing in the reward function, training in this way a model that avoids those configurations.
In conclusion, a future necessary step is to transfer the trained policies to a hardware system to evaluate how policy performance adapts to a real system. The policy coming from the CTR simulation environment is not directly applicable to the real CTR system. Thus one important step is to test the hardware system to understand its functionalities. For this purpose, a GUI to control CTR's joint has been developed exploiting ROS rqt framework. Rqt is a ROS package that enables plugin development. The work done is based on a preexistent plugin for robotic control.
As see in Fig. 6.1 the plugin is made of three pairs of sliders, each pair control extension(vertical slider) and rotation(orizontal slider) of a single tube. Moreover, the top couple controls the innermost tube, the middle one the middle tube and the lower one the outermost tube. As a ROS application there are two nodes involved which communicate using a publish/subscribe messaging model. Every time the user modifies any slider position a Joint State message is sent to the Joint Command topic, the message is in the form:

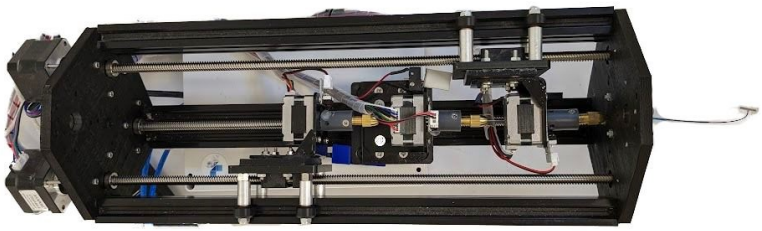
$$joint\_state = (\beta_1, \beta_2, \beta_3, \alpha_1, \alpha_2, \alpha_3) \tag{6.1}$$

where the β values correspond to the vertical slider current values and will act on joint extension, while α values correspond to the horizontal slider current value and will act on joint rotation. Sliders have constrained values which reflect the hardware joint limits.
The plugin has been tested on the hardware system shown in Fig.6.2 whose components are described in appendix A.

Figure 6.1: Joint Control Plugin



(a) Frontal view



(b) Top view

Figure 6.2: CTR harware system

Bibliography

[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] B Jaffray. Minimally invasive surgery. *Archives of Disease in Childhood*, 90(5):537–542, 2005.

[3] Stavros A Antoniou, George A Antoniou, Athanasios I Antoniou, and Frank-Alexander Granderath. Past, present, and future of minimally invasive abdominal surgery. *JSLS: Journal of the Society of Laparoendoscopic Surgeons*, 19(3), 2015.

[4] Michele Tonutti, Daniel S Elson, Guang-Zhong Yang, Ara W Darzi, and Mikael H Sodergren. The role of technology in minimally invasive surgery: state of the art, recent developments and future directions. *Postgraduate Medical Journal*, 93(1097):159–167, 2017.

[5] 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.

[6] Mohammed Abdel-Nasser and Omar Salah. New continuum surgical robot based on hybrid concentric tube-tendon driven mechanism. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 235(24):7550–7568, 2021.

[7] Junji Furusho, Tomoya Ono, Raifu Murai, Tatsuya Fujimoto, Yoshihide Chiba, and Hiroyuki Horio. Development of a curved multi-tube (cmt) catheter for percutaneous umbilical blood sampling and control methods of cmt catheters for solid organs. *IEEE International Conference Mechatronics and Automation, 2005*, 1:410–415 Vol. 1, 2005.

[8] Cédric Girerd and Tania K. Morimoto. Design and control of a hand-held concentric tube robot for minimally invasive surgery. *IEEE Transactions on Robotics*, 37(4):1022–1038, 2021.

[9] Patrick Sears and Pierre Dupont. A steerable needle technology using curved concentric tubes. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2850–2856, 2006.

[10] Hunter B. Gilbert, Joseph Neimat, and Robert J. Webster. Concentric tube robots as steerable needles: Achieving follow-the-leader deployment. *IEEE Transactions on Robotics*, 31(2):246–258, 2015.

[11] Jessica Burgner, D. Caleb Rucker, Hunter B. Gilbert, Philip J. Swaney, Paul T. Russell, Kyle D. Weaver, and Robert J. Webster. A telerobotic system for transnasal surgery. *IEEE/ASME Transactions on Mechatronics*, 19(3):996–1006, 2014.

[12] I Blickstein. Monochorionicity in perspective, 2006.

[13] Lynn L Simpson, Society for Maternal-Fetal Medicine (SMFM, et al. Twin-twin transfusion syndrome. *American journal of obstetrics and gynecology*, 208(1):3–18, 2013.

[14] Femke Slaghekke, Enrico Lopriore, Liesbeth Lewi, Johanna M Middeldorp, Erik W van Zwet, Anne-Sophie Weingertner, Frans J Klumper, Philip DeKoninck, Roland Devlieger, Mark D Kilby, Maria Angela Rustico, Jan Deprest, Romain Favre, and Dick Oepkes. Fetoscopic laser coagulation of the vascular equator versus selective coagulation for twin-to-twin transfusion syndrome: an open-label randomised controlled trial. *The Lancet*, 383(9935):2144–2151, 2014.

[15] Srinivas Neppalli, Matthew A. Csencsits, Bryan A. Jones, and Ian D. Walker. Closed-form inverse kinematics for continuum manipulators. *Advanced Robotics*, 23(15):2077–2091, 2009.

[16] Pierre E. Dupont, Jesse Lock, and Brandon Itkowitz. Real-time position control of concentric tube robots. In *2010 IEEE International Conference on Robotics and Automation*, pages 562–568, 2010.

[17] Pierre E. Dupont, Jesse Lock, and Evan Butler. Torsional kinematic model for concentric tube robots. In *2009 IEEE International Conference on Robotics and Automation*, pages 3851–3858, 2009.

[18] D. Caleb Rucker and Robert J. Webster. Mechanics of bending, torsion, and variable precurvature in multi-tube active cannulas. In *2009 IEEE International Conference on Robotics and Automation*, pages 2533–2537, 2009.

[19] Pierre E. Dupont, Jesse Lock, Brandon Itkowitz, and Evan Butler. Design and control of concentric-tube robots. *IEEE Transactions on Robotics*, 26(2):209–225, 2010.

[20] Konrad Leibrandt, Christos Bergeles, and Guang-Zhong Yang. Concentric tube robots: Rapid, stable path-planning and guidance for surgical use. *IEEE Robotics and Automation Magazine*, 24(2):42–53, 2017.

[21] D. Caleb Rucker and Robert J. Webster. Computing jacobians and compliance matrices for externally loaded continuum robots. In *2011 IEEE International Conference on Robotics and Automation*, pages 945–950, 2011.

[22] Ran Xu, Ali Asadian, Seyed Farokh Atashzar, and Rajni V. Patel. Real-time trajectory tracking for externally loaded concentric-tube robots. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4374–4379, 2014.

[23] Mohsen Khadem, John O'Neill, Zisos Mitros, Lyndon da Cruz, and Christos Bergeles. Autonomous steering of concentric tube robots via nonlinear model predictive control. *IEEE Transactions on Robotics*, 36(5):1595–1602, 2020.

[24] C Bergeles, FY Lin, and GZ Yang. Concentric tube robot kinematics using neural networks. In *Hamlyn symposium on medical robotics*, volume 6, pages 1–2, 2015.

[25] Reinhard Grassmann, Vincent Modes, and Jessica 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)*, pages 5125–5132, 2018.

[26] Alan Kuntz, Armaan Sethi, Robert J. Webster, and Ron Alterovitz. Learning the complete shape of concentric tube robots. *IEEE Transactions on Medical Robotics and Bionics*, 2(2):140–147, 2020.

[27] Nan Liang, Reinhard M. Grassmann, Sven Lilge, and Jessica 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)*, pages 1387–1393, 2021.

[28] D. Caleb Rucker, Bryan A. Jones, and Robert J. Webster III. A geometrically exact model for externally loaded concentric-tube continuum robots. *IEEE Transactions on Robotics*, 26(5):769–780, 2010.

[29] Georgios Fagogenis, Christos Bergeles, and Pierre E. Dupont. Adaptive nonparametric kinematic modeling of concentric tube robots. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4324–4329, 2016.

[30] Balint Thamo, Farshid Alambeigi, Kev Dhaliwal, and Mohsen Khadem. A hybrid dual jacobian approach for autonomous control of concentric tube robots in unknown

constrained environments. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2809–2815, 2021.

[31] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[32] Richard Bellman. Dynamic programming. *science*, 153(3731):34–37, 1966.

[33] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8:279–292, 1992.

[34] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Reinforcement learning*, pages 5–32, 1992.

[35] Jianqing Fan, Zhaoran Wang, Yuchen Xie, and Zhuoran Yang. A theoretical analysis of deep q-learning. In *Learning for Dynamics and Control*, pages 486–489. PMLR, 2020.

[36] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, 2016.

[37] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[38] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[39] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

[40] K Iyengar and D Stoyanov. Deep reinforcement learning for concentric tube robot control with goal based curriculum reward. In *https://cras-eu. org/wp-content/uploads/2020/09/CRAS_ 2020_ proceedings. pdf*, pages 60–61. Computer-Assisted Radiology and Surgery, 2020.

[41] Keshav Iyengar, George Dwyer, and Danail Stoyanov. Investigating exploration for deep reinforcement learning of concentric tube robot control. *International Journal of Computer Assisted Radiology and Surgery*, 15(7):1157–1165, 2020.

[42] Keshav Iyengar and Danail Stoyanov. Deep reinforcement learning for concentric tube robot control with a goal-based curriculum. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1459–1465. IEEE, 2021.

[43] Keshav Iyengar, Sarah Spurgeon, and Danail Stoyanov. Unconstrained rotation for control of concentric tube robots with deep reinforcement learning. CRAS, 2022.

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

[45] 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.

[46] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[47] Antonin Raffin, Ashley Hill, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, and Noah Dormann. Stable baselines3, 2019.

[48] Baohe Zhang, Raghu Rajan, Luis Pineda, Nathan Lambert, André Biedenkapp, Kurtland Chua, Frank Hutter, and Roberto Calandra. On the importance of hyperparameter optimization for model-based reinforcement learning. In *International Conference on Artificial Intelligence and Statistics*, pages 4015–4023. PMLR, 2021.

[49] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2623–2631, 2019.

[50] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter 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)*, pages 23–30. IEEE, 2017.

[51] Hunter B Gilbert, Richard J Hendrick, and Robert J Webster III. Elastic stability of concentric tube robots: A stability measure and design test. *IEEE Transactions on Robotics*, 32(1):20–35, 2015.

[52] Richard J Hendrick, Hunter B Gilbert, and Robert J Webster. Designing snap-free concentric tube robots: A local bifurcation approach. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2256–2263. IEEE, 2015.

[53] Ji-Suk Kim, Dae-Young Lee, Keri Kim, Sungchul Kang, and Kyu-Jin Cho. Toward a solution to the snapping problem in a concentric-tube continuum robot: Grooved tubes with anisotropy. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5871–5876. IEEE, 2014.

[54] SM Hadi Sadati, Zisos Mitros, Ross Henry, Lingyun Zeng, Lyndon Da Cruz, and Christos Bergeles. Real-time dynamics of concentric tube robots with reduced-order kinematics based on shape interpolation. *IEEE Robotics and Automation Letters*, 7(2):5671–5678, 2022.

[55] Christos Bergeles and Pierre E Dupont. Planning stable paths for concentric tube robots. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3077–3082. IEEE, 2013.

A | Appendix A

CTR HARDWARE SYSTEM

The hardware system used for plugin experiments as well as for tube parameters measurement has the following components:

- 3 nitinol tubes, 2 full nitinol, innermost has steel section in straight length
- Chassis: 3 aluminum extrusions of 200mm length, 3D printed plates for motor mounts and robot front with tube entry
- 3x HANPOSE 17HS3401S for translation set to 4 microsteps (motor has 1.8 deg / step accuracy)
- 3x HANPOSE 17HS3401S for rotation set to 16 microsteps (motor has 1.8 deg / step accuracy)
- 6x limit switches (Creality Official Limit Switch End Stop 3PCS 3D Printer Part Compatible with Ender 3 / Ender 3 Pro/Ender 3 V2 / Ender 5 Series/CR-10 Series 3D Printer)
- 3x brass collets and chucks (to hold tubes)
- 3x linear rails (RUOKL 2Kit Small V-Wheel Plate for 2020 Series V-Slot Aluminum Profiles Linear Rail,Assembled V-Slot Gantry Plate with POM Wheels suitable)
- 3x 1 mm per rev linear screw
- 12x calibration pins (designed and 3D printed)
- 3x rotational bumps for rotation homing (designed and 3D printed)
- 3x transltation limit blocks (designed and 3D printed)
- 2x Arduinos uno
- 2x motor controller shields (Arduino GRBL CNC shield v3)
- Aurora tracking system

List of Figures

List of Tables

Acknowledgements

I would like to thank Prof.ssa Elena De Momi for giving me the opportunity to work on a fascinating project in an inspiring environment like WEISS Lab in London.
I would like to express my gratitude to all the guys from WEISS for welcoming me and teaching me so much. A particular mention to Keshav Iyengar, thank you for your patience, care and humility, for expanding my technical and personal background.
A special thought to my family and all my friends who always supported me in all my choices.