The Control of Soft Continuum Robot by Reinforcement Learning Algorithm

Master's Thesis Defense Presentation

Turhan Can KARGIN

Supervisor: Dr Hab. Inż. Jakub Bernat

POZNAN UNIVERSITY OF TECHNOLOGY

Poznań 2022

Outline

Agenda

Timeline

- Development of kinematic model of the continuum robot : **March** - **April**
- Design the RL environment to solve control goal : **April** - **May**
- Learning the agent for defined set of cases : **May** - **June**
- Preparing simulations and analysis of obtained controller : **June** - **August**

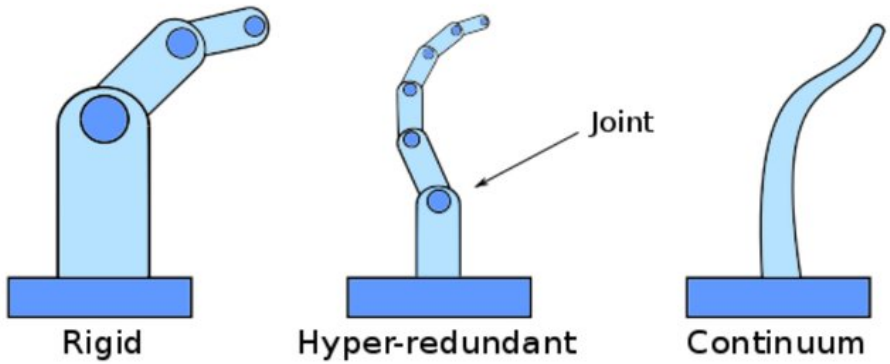


Figure: From rigid to continuum robots

Source: M. Thieffry, Model-Based Dynamic Control of Soft Robots. PhD thesis, Universite Polytechnique des Hauts-de-France, 2019.

Main Goal

The main goal of the project is to model the continuum robot using existing frameworks and to investigate and simulate the advantages of applying deep reinforcement learning to continuum robots using the model. While designing the simulation, the deep deterministic policy gradient algorithm of deep reinforcement learning, which is actor-critical based, was used because it can work in a very large state space and gives successful results in continuous action situations. The idea of the simulation is to control the three segment continuum robot by taking from a random starting point to a random target point.

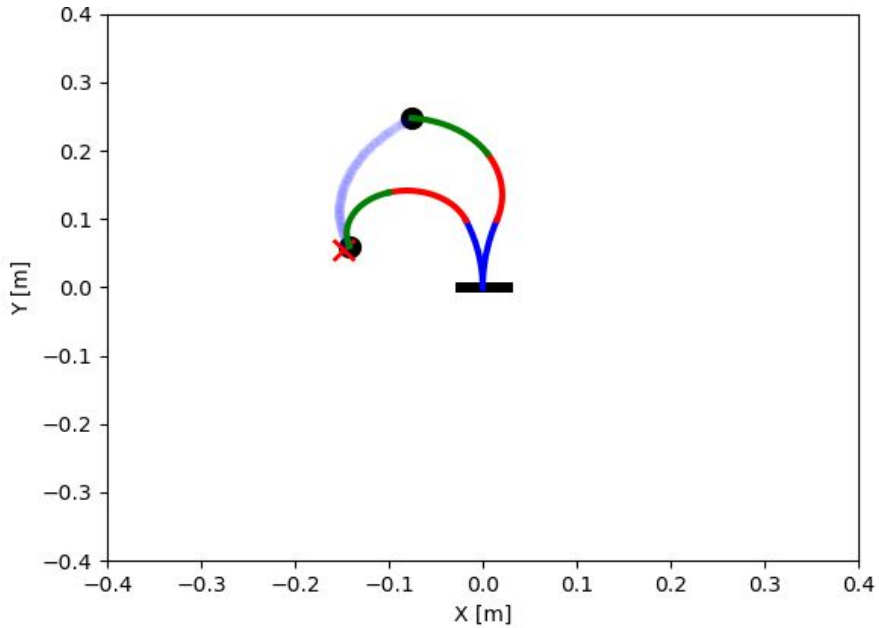


Figure: From a random starting point to a random target point

Introduction

Motivations

For decades, researchers have made massive efforts to make machines intelligent, in expectation of relieving human labors from repetitive, dangerous, and heavy work. In traditional robotics, control of robots is realized by establishing kinematic and dynamic models in the form of a transformation matrix. This method has achieved excellent results in conventional robots with discrete rigid links, but becomes difficult to implement when dealing with soft robots such as continuum robots.

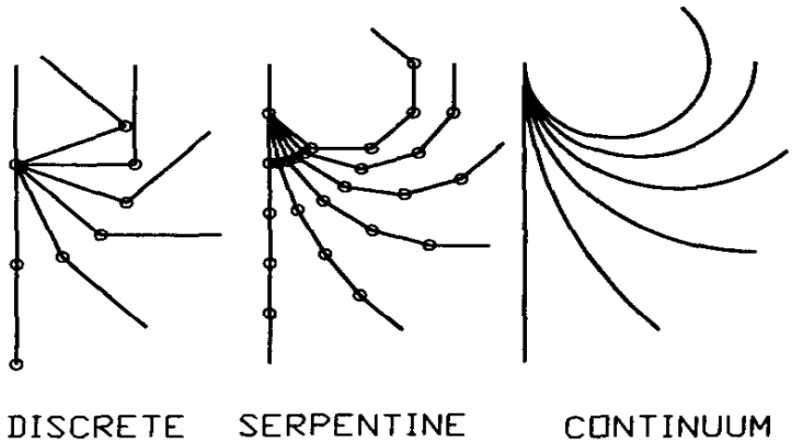


Figure: Robot Motions

Source: Robinson, G., and Davies, J. B. C. (1999, May). Continuum robots - a state of the art

Introduction

Motivations

The kinematic and dynamic models for continuum robots are often described in the form of nonlinear partial differential equations, which makes the control more complex. Ever since reinforcement learning (RL) theory was proposed, developers have been trying to apply it to robotics. With introducing RL methods, it enhances the traditional methods.

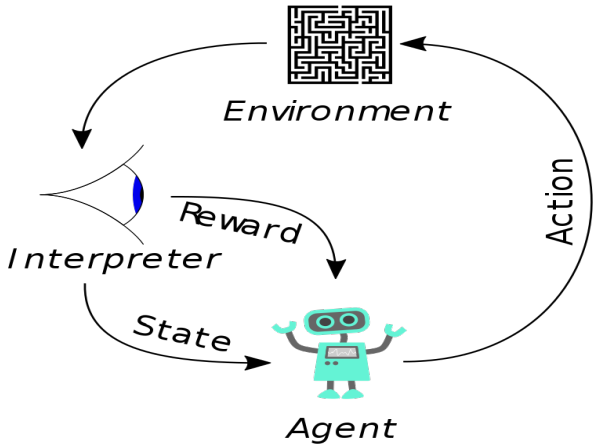


Figure: The typical Reinforcement Learning (RL) scenario

Source: https://en.wikipedia.org/wiki/Reinforcement_learning

Definition of Continuum Robot

Nature Inspired Continuum Robot

Continuum robots are an increasingly popular class of manipulators characterized by their ability to assume continuous curved shapes. The concept of continuum robot is indeed inspired by biological entities like octopus tentacles, snakes, trunks.

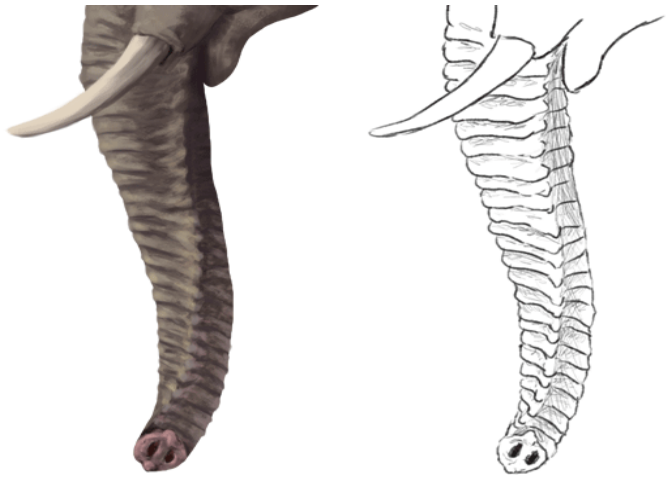


Figure: Click Here - Continuum robot can be designed like elephant trunk

Definition of Continuum Robot

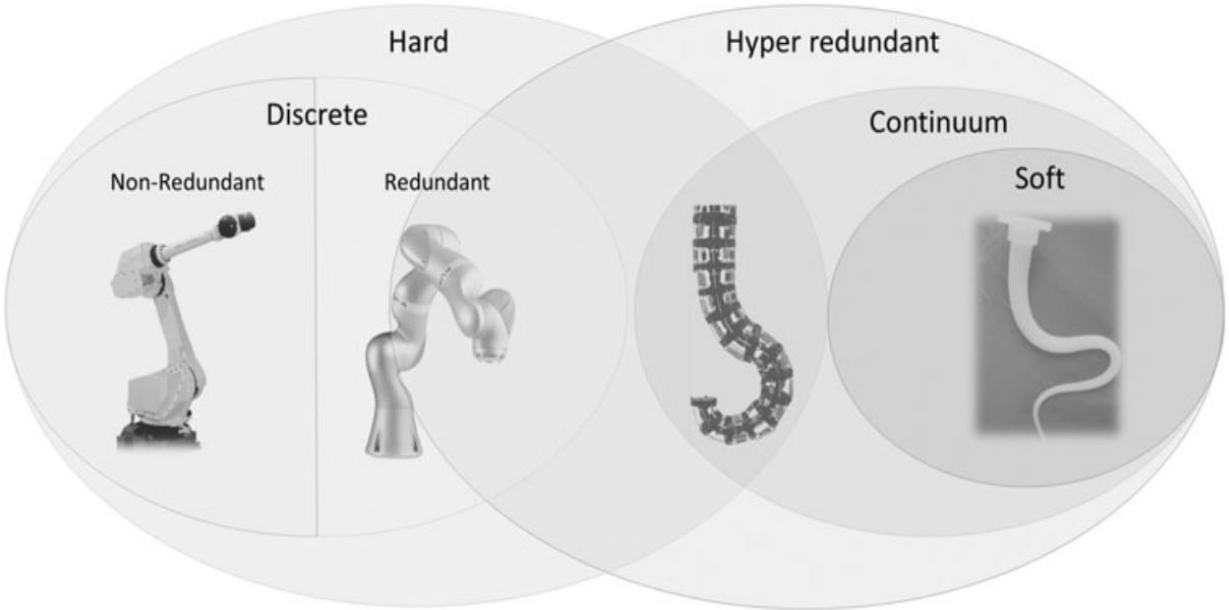


Figure: Classification of manipulators based on materials and DoF

Source: T. George Thuruthel, Y. Ansari, E. Falotico, C. Laschi, Control strategies for soft robotic manipulators: A survey. Soft Robot. 5, 149-163 (2018). https://doi.org/10.1089/soro.2017.0007

Definition of Continuum Robot

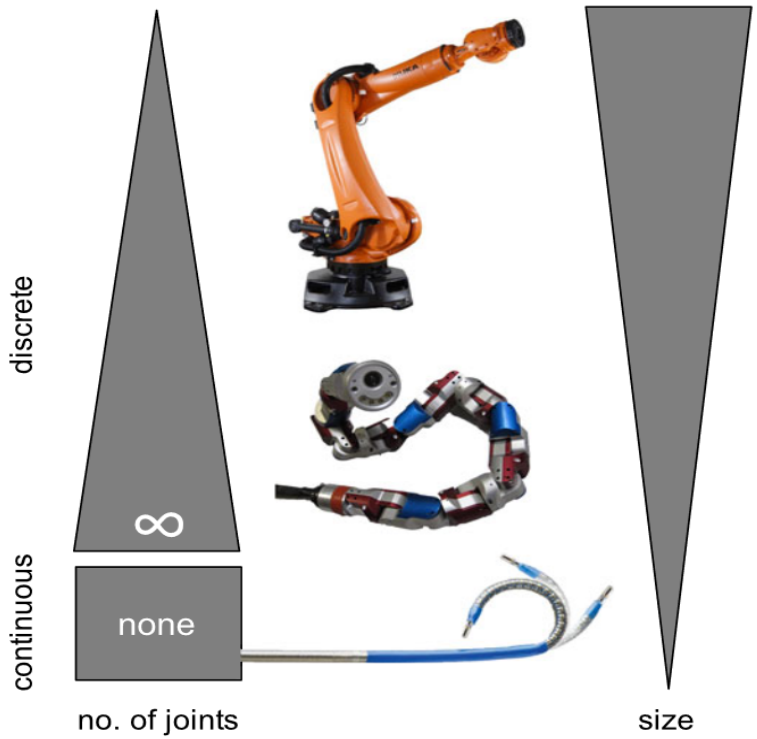
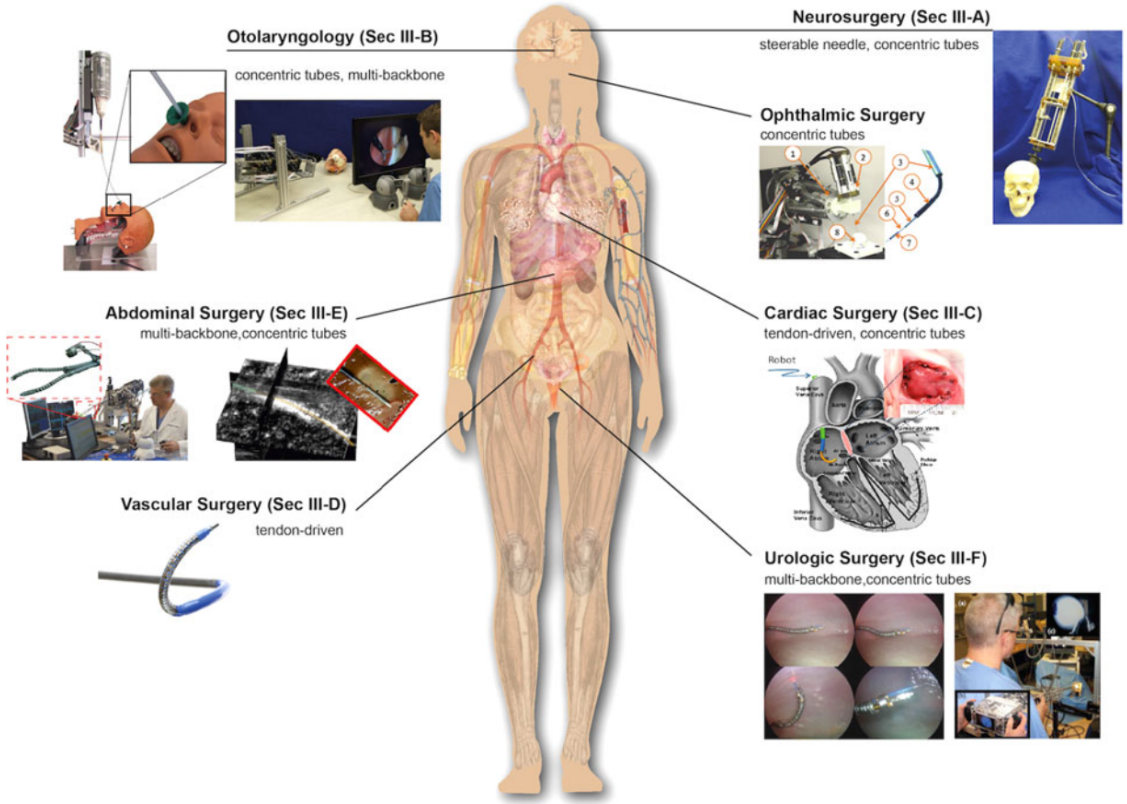


Figure: Continuum robots have an infinite-DOF curvilinear elastic structure

Source: Burgner-Kahrs, J., Rucker, D. C., and Choset, H. (2015). Continuum Robots for Medical Applications: A Survey. IEEE Transactions on Robotics

Application of Continuum Robot

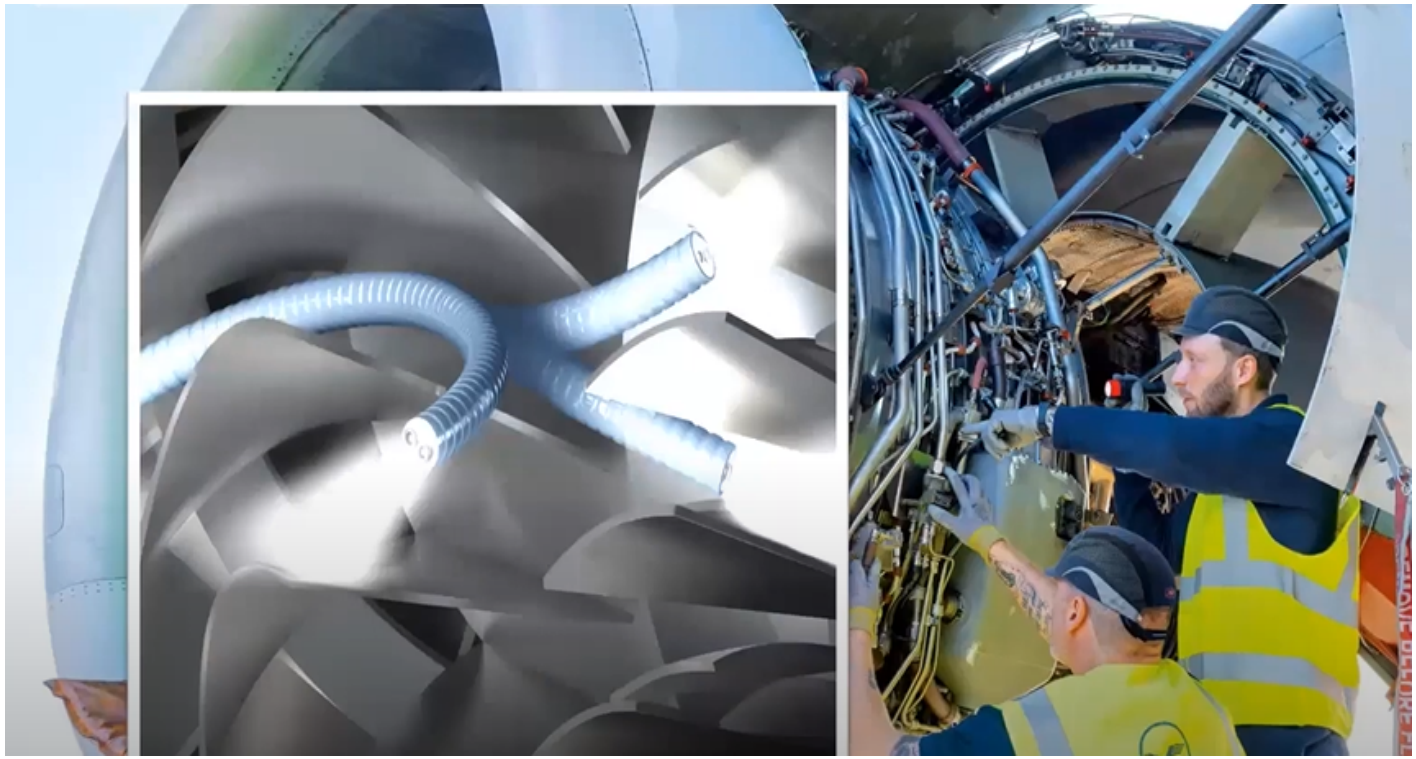
Medical Applications



Source: Burgner-Kahrs, J., Rucker, D. C., and Choset, H. (2015). Continuum Robots for Medical Applications: A Survey

Application of Continuum Robot

Maintenance, Repair, and Operation Applications



Source: ICRA 2021 Keynote Talk – Jessica Burgner-Kahrs: I, Continuum Robot

Definition of Reinforcement Learning

- It is an area of ML concerned with how intelligent agents ought to take actions in an environment in order to maximize the notion of cumulative reward. Reinforcement learning is one of three basic machine learning paradigms.
- The focus is on finding a balance between exploration (of uncharted territory) and exploitation (of current knowledge).
- **The main purpose is to maximize the reward. So it is a optimization problem.**

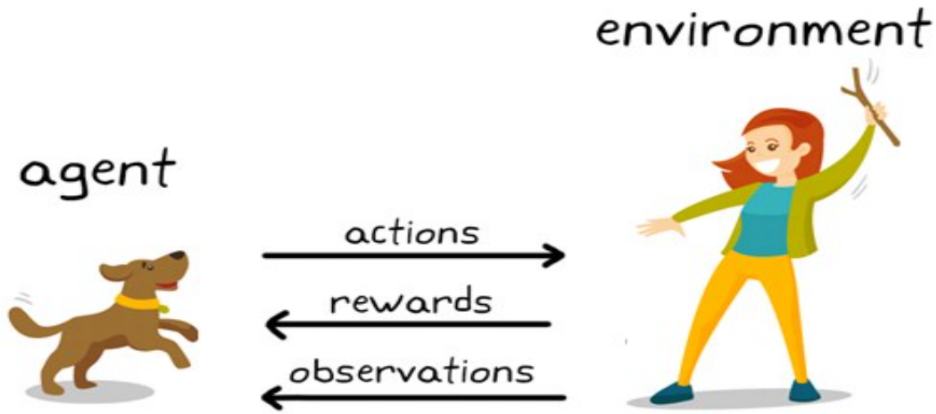


Figure: Reinforcement learning in dog training

Source: https://www.kdnuggets.com/2019/10/mathworks-reinforcement-learning.html

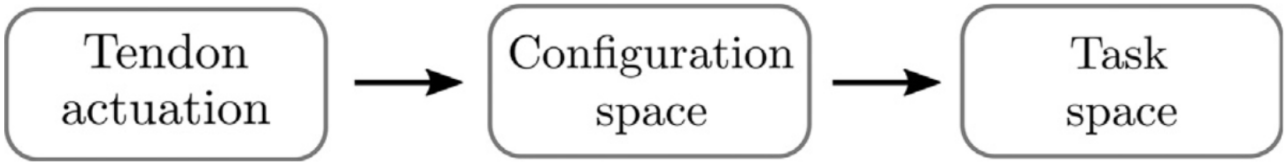
Definition of Reinforcement Learning

- Reinforcement learning is emerging in the robotics community, which is a natural application for learning-based control since the interaction between robots and the environment is necessary. Reinforcement learning offers to complete sophisticated tasks and accommodate complex environments, which may be limited in conventional control strategies.
- Reinforcement learning enables a robot to autonomously discover an optimal behavior through trial-and-error interactions with its environment.

Figure: Robot Hand Solving Rubik's Cube via RL

Source: https://openai.com/blog/solving-rubiks-cube/

Modelling the Continuum Robot



The modelling framework of a continuum robot that considers the tendon actuation to calculate the resulting configuration space parameters. The configuration space parameters can then be used to obtain the corresponding backbone shape in planar or spatial space.

Source: Rao, P., Peyron, Q., Lilge, S., and Burgner-Kahrs, J. (2021). How to Model Tendon-Driven Continuum Robots and Benchmark Modelling Performance.

Modelling the Continuum Robot

$$\kappa : [\tfrac{1}{m}],\ l : [m]$$

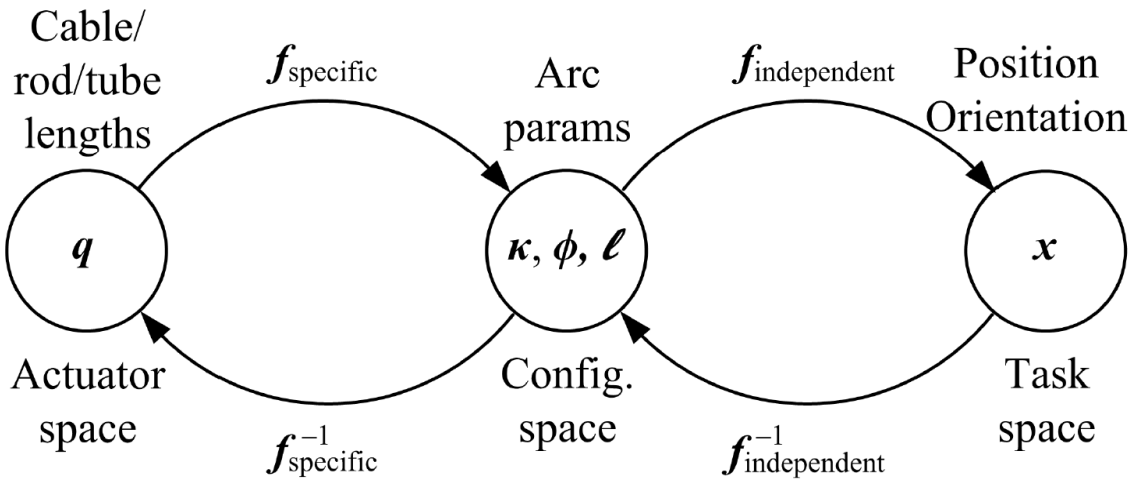
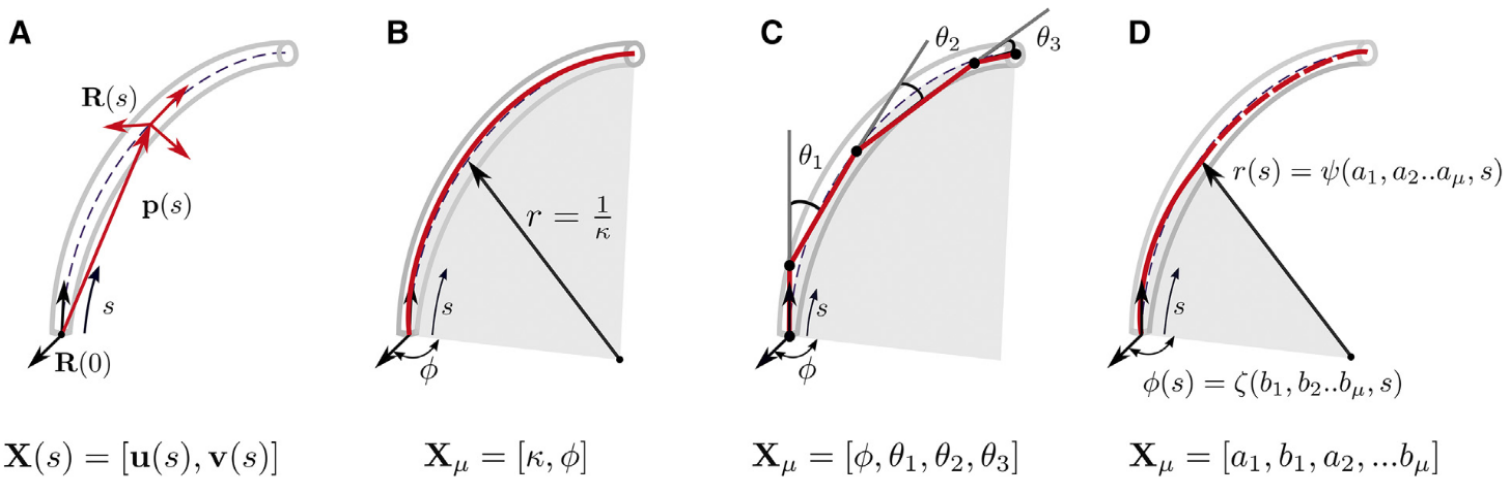


Figure: The three spaces and mappings between them, which define the kinematics of constant-curvature robots.

Source: Webster, R. J., and Jones, B. A. (2010). Design and Kinematic Modeling of Constant Curvature Continuum Robots: a Review.

Modelling the Continuum Robot



$\mathbf{X}(s) = [\mathbf{u}(s), \mathbf{v}(s)]$ $\quad \mathbf{X}_\mu = [\kappa, \phi]$ $\quad \mathbf{X}_\mu = [\phi, \theta_1, \theta_2, \theta_3]$ $\quad \mathbf{X}_\mu = [a_1, b_1, a_2, ...b_\mu]$

Diagrammatic representation of various kinematic frameworks used to describe the continuum robot's backbone.

Source: Rao, P., Peyron, Q., Lilge, S., and Burgner-Kahrs, J. (2021). How to Model Tendon-Driven Continuum Robots and Benchmark Modelling Performance.

Our Model Selection

Constant Curvature Approximation

This assumption forms the basis for a common approach to parameterizing the backbone.
Credit: Hannan, M. W., and Walker, I. D. (2003). Kinematics and the Implementation of an Elephant's Trunk Manipulator and Other Continuum Style Robots

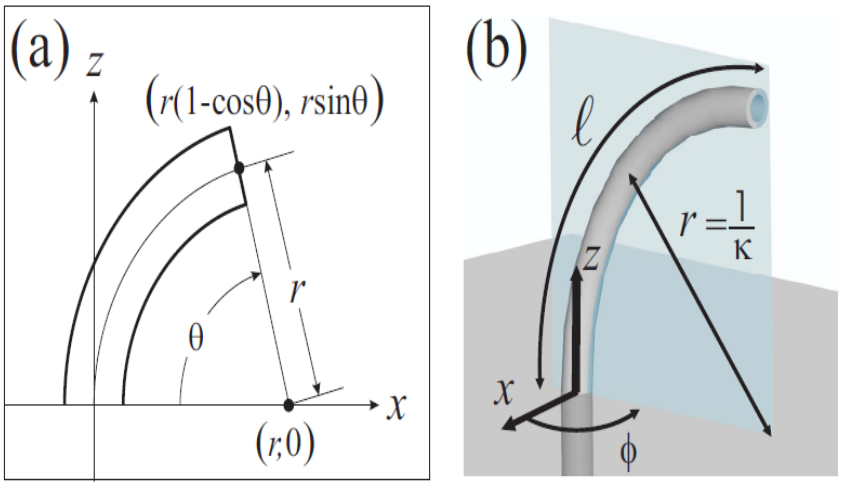


Figure: Constant Curvature Representation

Velocity Kinematics (Jacobian)

Velocity Kinematics and Jacobian Matrix

- Jacobian matrices are a super useful tool, and heavily used throughout robotics and control theory. Basically, a Jacobian defines the dynamic relationship between two different representations of a system.
- This relationship in our case is derivative of position and derivative of curvature.
- In velocity kinematics, the target is to describe the motion of end-effector (Tip in our case). This motion is highly constrained by joints (curvatures in our case), which means describing the states of motion at those joint is describing the motion of the end-effector.

Method for Reinforcement Learning

Deep Deterministic Policy Gradient

- Deep Deterministic Policy Gradient (DDPG) is a reinforcement learning algorithm for learning continuous actions.
- It combines ideas from DPG (Deterministic Policy Gradient), Actor-Critic method and DQN (Deep Q-Network). It uses Experience Replay and slow-learning target networks from DQN, and it is based on DPG, which can operate over continuous action spaces.
- It can solve very large action and state spaces because it uses neural networks separately for both action selection and Q value generation.
- These neural networks are for function approximation of Q value and action.

Training the Robot

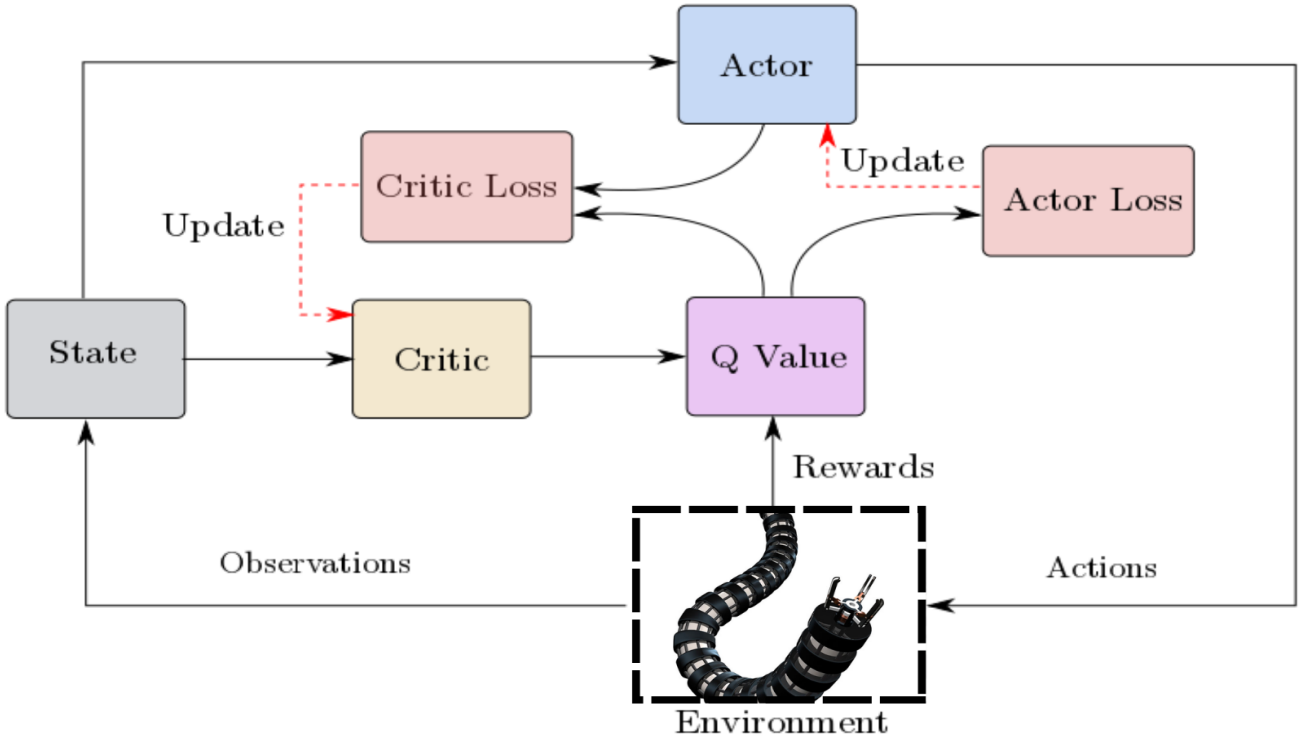


Figure: Deep Deterministic Policy Gradient Training Scheme

Testing the Robot

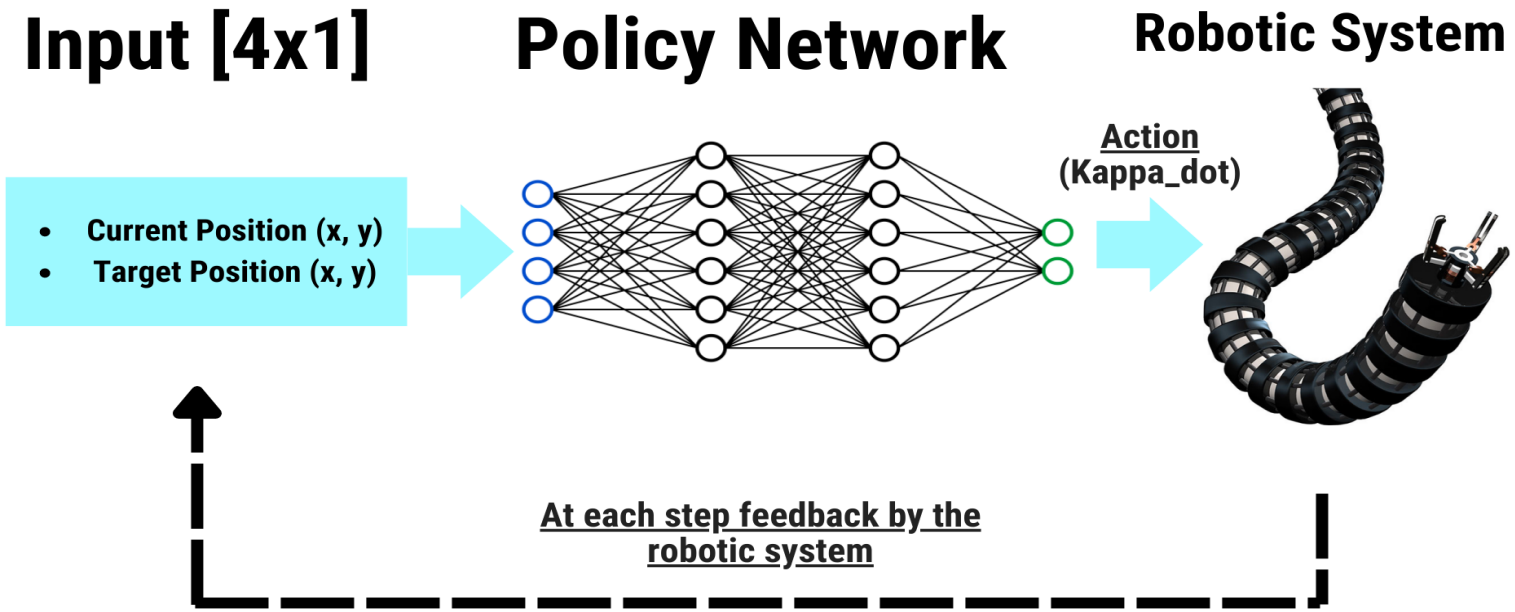
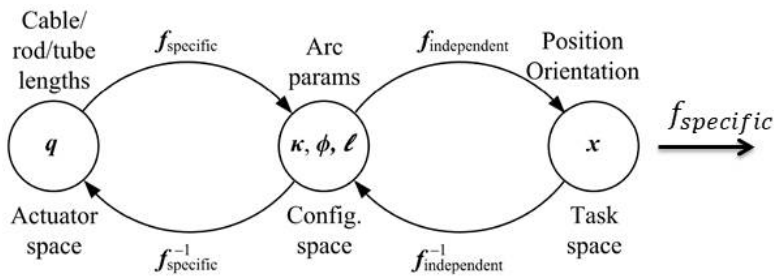
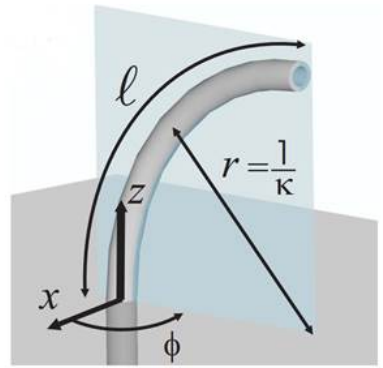


Figure: Control testing scheme

Planar Robot Forward Kinematics Formulation



$f_{specific}$



$f_{independent}$

Only for one section!!!!

$$\left[\begin{array}{c|c} \begin{pmatrix}\text{Rotation}\\ \text{Matrix}\end{pmatrix}_{old}^{new} & \begin{pmatrix}\text{Location}\\ \text{of Old}\\ \text{Origin}\end{pmatrix}_{new} \\ \hline \begin{pmatrix}0 & 0 & 0\end{pmatrix} & 1 \end{array}\right]$$

$$\begin{bmatrix} \cos(\kappa l) & -\sin(\kappa l) & 0 & \frac{1}{\kappa}\{\cos(\kappa l)-1\} \\ \sin(\kappa l) & \cos(\kappa l) & 0 & \frac{1}{\kappa}\sin(\kappa l) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

What about Robot with 3 sections?

$$T_0^3 = T_0^1 \cdot T_1^2 \cdot T_2^3$$

Figure: Forward Kinematics of the Three Sections Planar Continuum Robot

Let's just find the Tip of the Robot (Computationally Less Expensive)

$$T_0^3 = T_0^1 \cdot T_1^2 \cdot T_2^3$$

$$T_0^3 = \begin{bmatrix} \cos(\omega_1+\omega_2+\omega_3) & -\sin(\omega_1+\omega_2+\omega_3) & 0 & A_{14} \\ \sin(\omega_1+\omega_2+\omega_3) & \cos(\omega_1+\omega_2+\omega_3) & 0 & A_{24} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_{14} = \frac{1}{\kappa_1}\{\cos\omega_1 - 1\} + \frac{1}{\kappa_2}\{\cos(\omega_1+\omega_2) - \cos\omega_1\}$$
$$+\frac{1}{\kappa_3}\{\cos(\omega_1+\omega_2+\omega_3) - \cos(\omega_1+\omega_2)\}$$
$$A_{24} = \frac{1}{\kappa_1}\sin\omega_1 + \frac{1}{\kappa_2}\{\sin(\omega_1+\omega_2) - \sin\omega_1\}$$
$$+\frac{1}{\kappa_3}\{\sin(\omega_1+\omega_2+\omega_3) - \sin(\omega_1+\omega_2)\},$$

Figure: Tip calculation of the robot

Forward Kinematics Implementation (Planar Case)

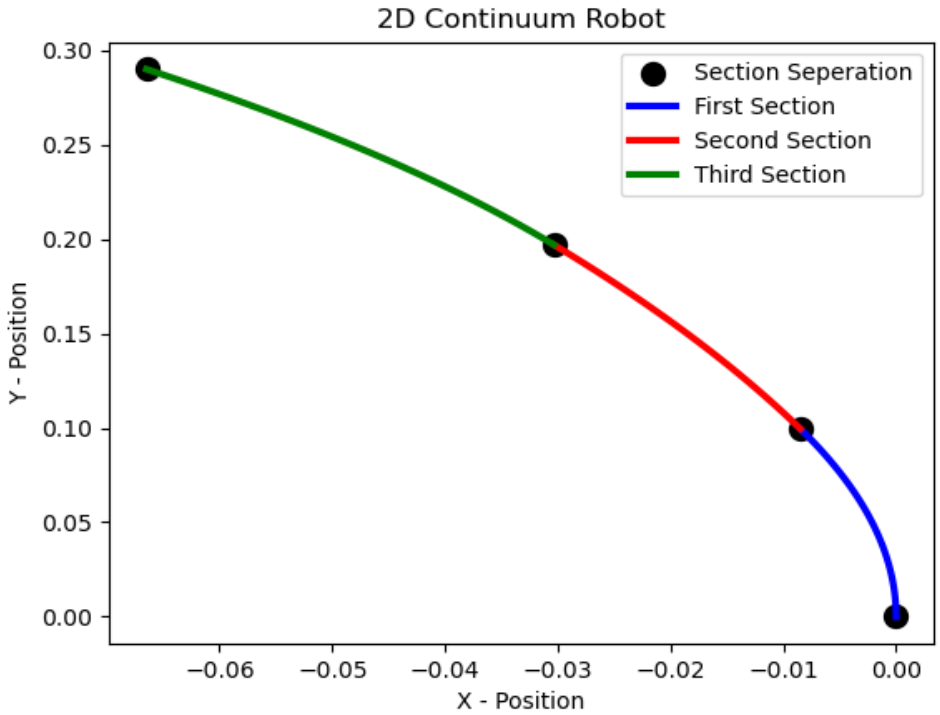


Figure: Planar Case for Continuum Robot

$$-4 \leq \kappa_1, \kappa_2, \kappa_3 \leq 16$$

Velocity Kinematics Formulation

$$\dot{x} = J\dot{q},$$

$$\mathbf{q} = [\kappa_1, \kappa_2, \kappa_3]^T,$$
$$\dot{\mathbf{q}} = [\dot{\kappa_1}, \dot{\kappa_2}, \dot{\kappa_3}]^T,$$
$$\dot{\mathbf{x}} = [\dot{x}, \dot{y}]^T$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} J_{11} & J_{12} \\ J_{21} & J_{22} \end{bmatrix} \begin{bmatrix} \dot{q_1} \\ \dot{q_2} \end{bmatrix}$$

$$\frac{dx}{dt} = \frac{\partial x}{\partial q_1}\frac{dq_1}{dt} + \frac{\partial x}{\partial q_2}\frac{dq_2}{dt},$$
$$\frac{dy}{dt} = \frac{\partial y}{\partial q_1}\frac{dq_1}{dt} + \frac{\partial y}{\partial q_2}\frac{dq_2}{dt}$$

$$\frac{dx}{dt} = \frac{\partial x}{\partial \kappa_1}\frac{d\kappa_1}{dt} + \frac{\partial x}{\partial \kappa_2}\frac{d\kappa_2}{dt} + \frac{\partial x}{\partial \kappa_3}\frac{d\kappa_3}{dt},$$
$$\frac{dy}{dt} = \frac{\partial y}{\partial \kappa_1}\frac{d\kappa_1}{dt} + \frac{\partial y}{\partial \kappa_2}\frac{d\kappa_2}{dt} + \frac{\partial y}{\partial \kappa_3}\frac{d\kappa_3}{dt}$$

$$J = \begin{bmatrix} \frac{\partial x}{\partial \kappa_1} & \frac{\partial x}{\partial \kappa_2} & \frac{\partial x}{\partial \kappa_3} \\ \frac{\partial y}{\partial \kappa_1} & \frac{\partial y}{\partial \kappa_2} & \frac{\partial y}{\partial \kappa_3} \end{bmatrix}$$

$$\begin{bmatrix} \frac{dx}{dt} \\ \frac{dy}{dt} \end{bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial \kappa_1} & \frac{\partial x}{\partial \kappa_2} & \frac{\partial x}{\partial \kappa_3} \\ \frac{\partial y}{\partial \kappa_1} & \frac{\partial y}{\partial \kappa_2} & \frac{\partial y}{\partial \kappa_3} \end{bmatrix} \begin{bmatrix} \frac{d\kappa_1}{dt} \\ \frac{d\kappa_2}{dt} \\ \frac{d\kappa_3}{dt} \end{bmatrix}$$

Algorithm for Simple Motion

Algorithm Describing a motion with velocity kinematics

1: Choose starting κ_1, κ_2, κ_3 as curvatures for each section and define l_1, l_2, and l_3 as length of each section.
2: Calculate Jacobian Matrix
3: Choose curvature velocities,$\dot{\kappa}$, for each κ
4: **while** True **do**
5: Calculate $\dot{x}$
6: Move the tip of the robot with velocities from step 3. $\triangleright$ $x_{t+1} = x_t + \dot{x}_t \cdot dt$
7: Update the jacobian as it is a function of the curvature, which have now changed

Velocity Kinematics Implementation

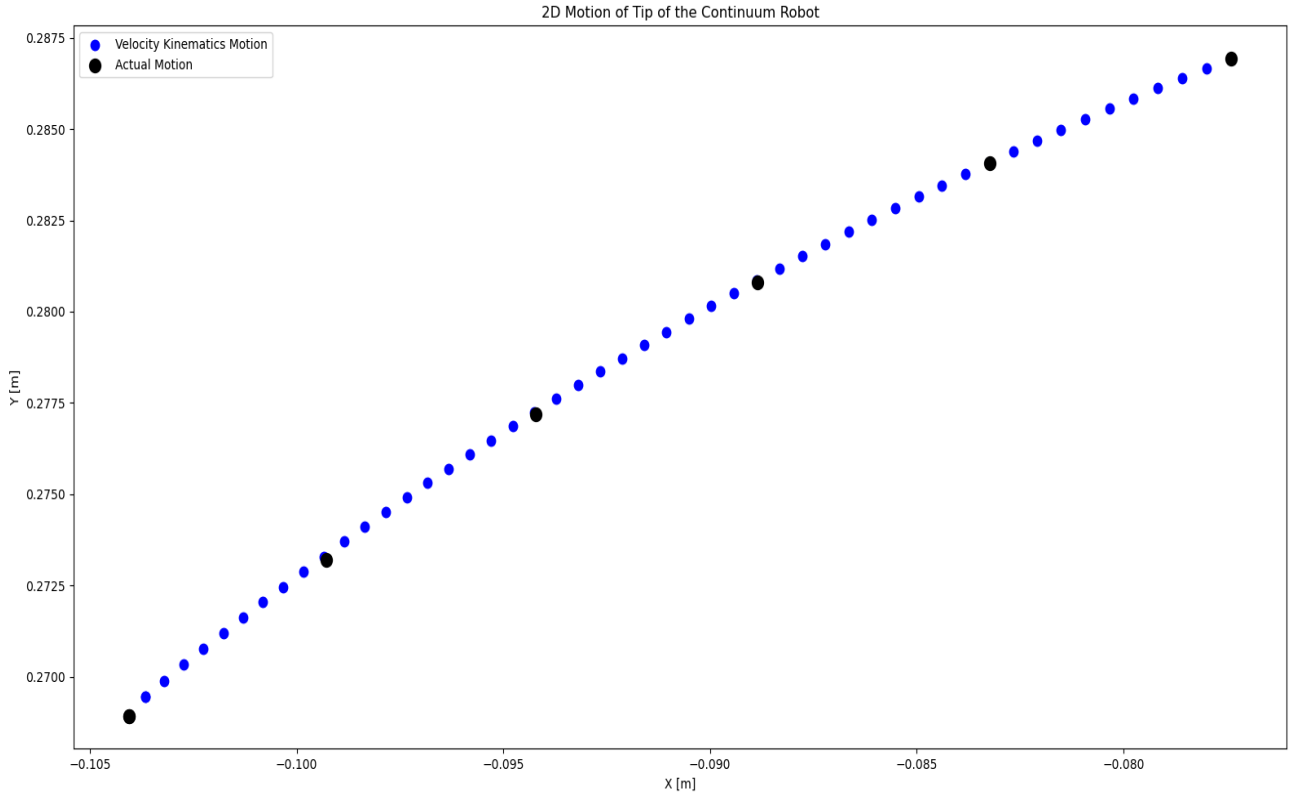


Figure: Velocity Kinematics for Continuum Robot

DDPG Algorithm

Algorithm Deep Deterministic Policy Gradient Algorithm

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\pi(s|\theta^\pi)$ with weights θ^Q and θ^π.
Initialize target network $\hat{Q}$ and $\hat{\pi}$ with weights $\theta^{\hat{Q}} \leftarrow \theta^Q$, $\theta^{\hat{\pi}} \leftarrow \theta^\pi$
Initialize replay buffer R
for episode = 1, M **do**
    Initialize a random process $\mathcal{N}$ for action exploration
    Receive initial observation state s_1
    for t = 1, T **do**
        Select action $a_t = \pi(s_t|\theta^\pi) + \mathcal{N}_t$ according to the current policy and exploration noise
        Execute action a_t and observe reward r_t and observe new state s_{t+1}
        Store transition (s_t, a_t, r_t, s_{t+1}) in R
        Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R
        Set $y_i = r_i + \gamma \hat{Q}(s_{i+1}, \hat{\pi}(s_{i+1}|\theta^{\hat{\pi}})|\theta^{\hat{Q}})$
        Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
        Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\pi} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\pi(s_i)} \nabla_{\theta^\pi} \pi(s|\theta^\pi)|_{s_i}$$

        Update the target networks:

$$\theta^{\hat{Q}} \leftarrow \tau\theta^Q + (1-\tau)\theta^{\hat{Q}}$$

$$\theta^{\hat{\pi}} \leftarrow \tau\theta^\pi + (1-\tau)\theta^{\hat{\pi}}$$

Figure: Click Here

Environment Design

- RL environment was created based on the kinematics in previous slides
- **Action Space:** $\dot{K}_1, \dot{K}_2, \dot{K}_3$
- **Observation Space (States):** x, y, x_{goal}, y_{goal}
- **Reward:** $-(d_u)^2$

where $d_u = e = \sqrt{(x - x_{goal})^2 + (y - y_{goal})^2}$

Task Space Analysis

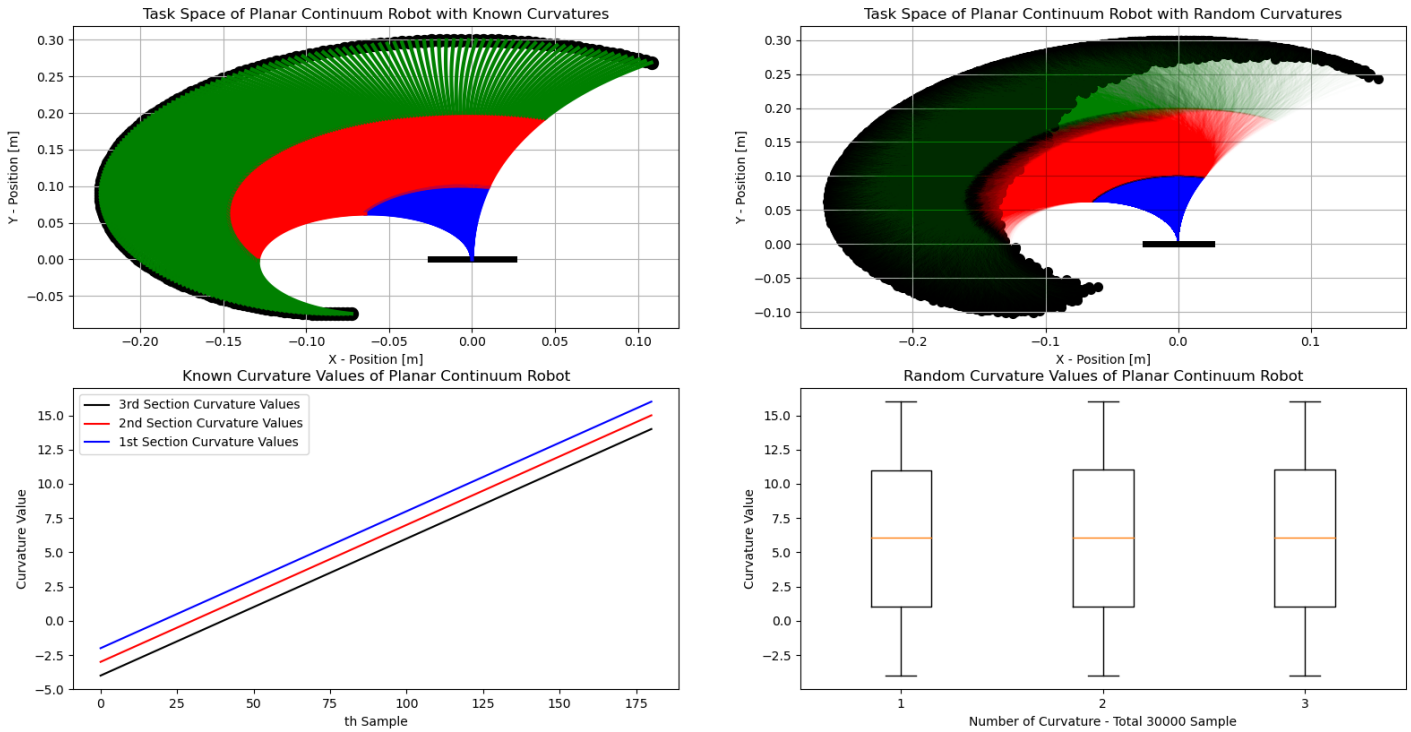
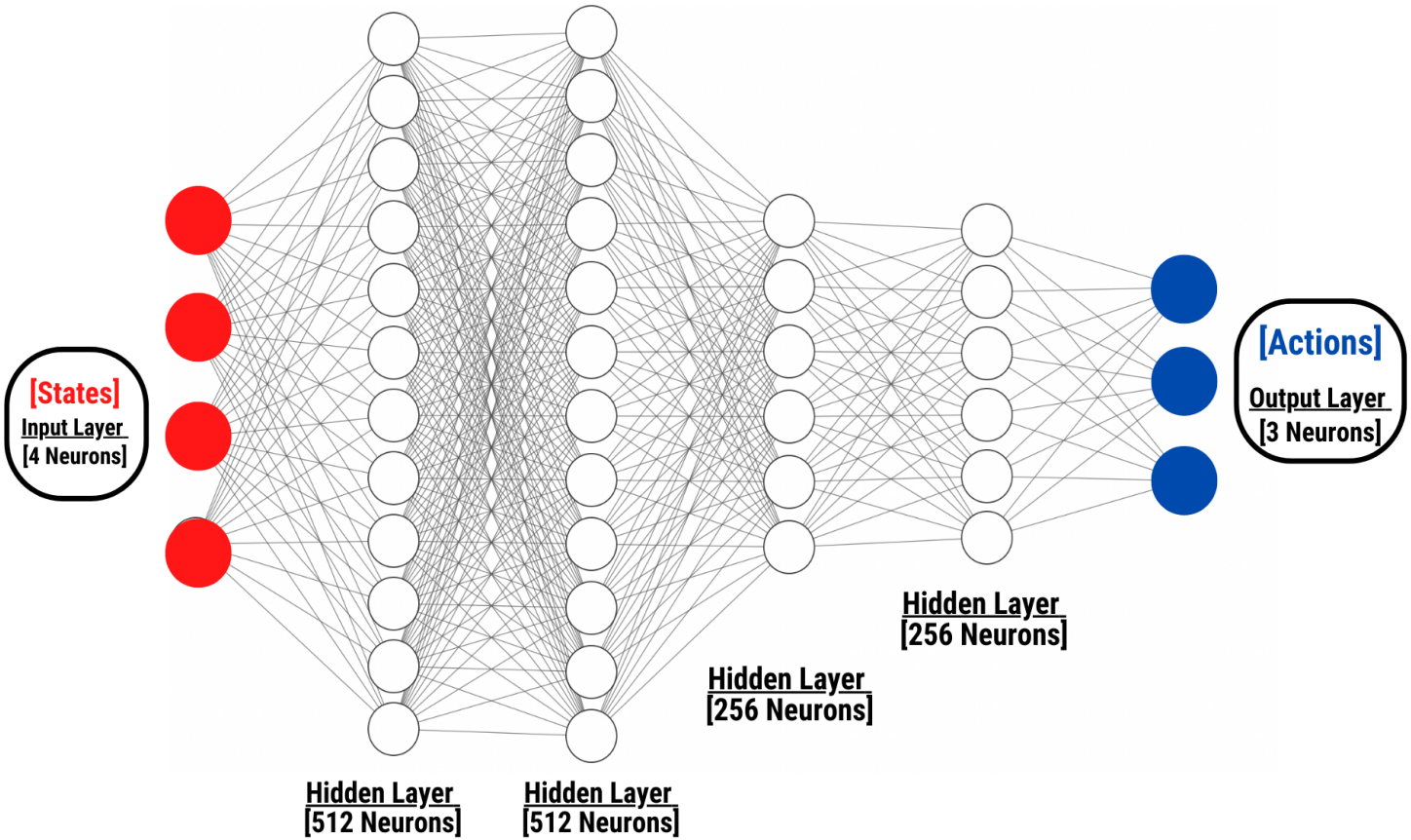


Figure: Simulation result of task space as well as the curvature values

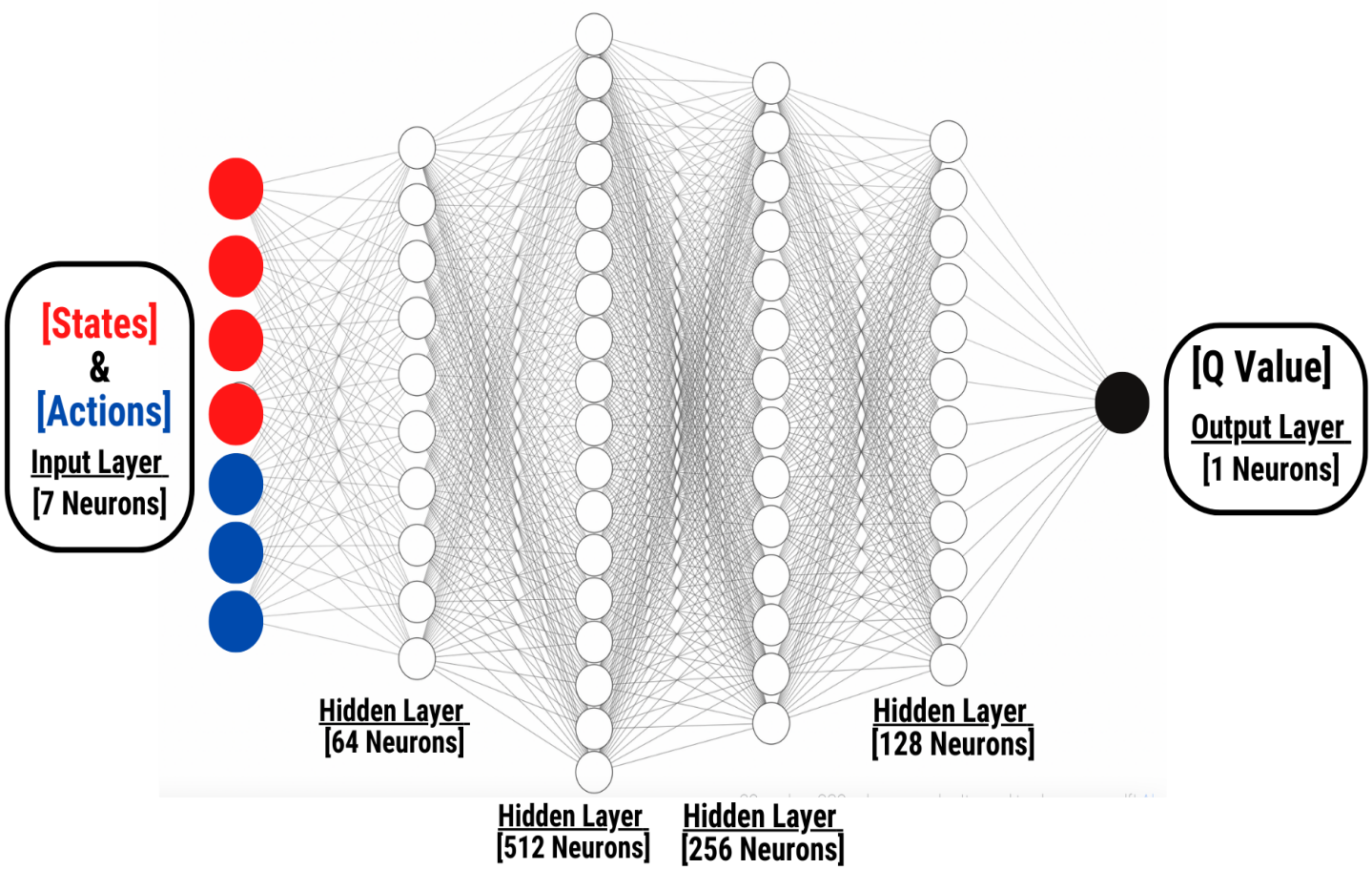
Hyper-parameters Tuning

Parameter	Value
Optimizer	Adam
Learning Rare Actor	0.0001
Learning Rate Critic	0.0003
Discount Factor (γ)	0.99
Batch Size	128
Replay Buffer Size	1000000
Actor Hidden Layer Number	4
Actor Hidden Layer Activation Function	Relu
Actor Output Layer Activation Function	Tangent Hyperbolic
Critic Hidden Layer Number	4
Critic Hidden Layer Activation Function	Relu
Critic Output Layer Activation Function	Linear
Maximum training steps	2000
Soft Update (τ)	0.001

Actor-Critic Architecture



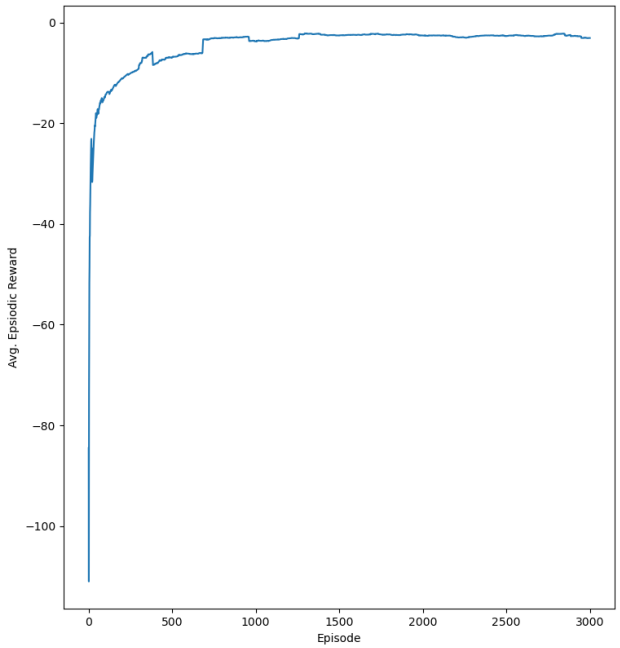
Actor-Critic Architecture

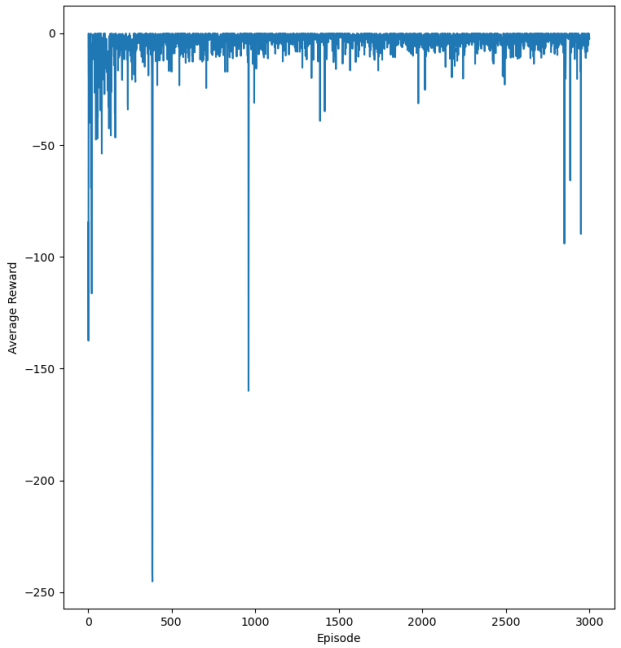


Results

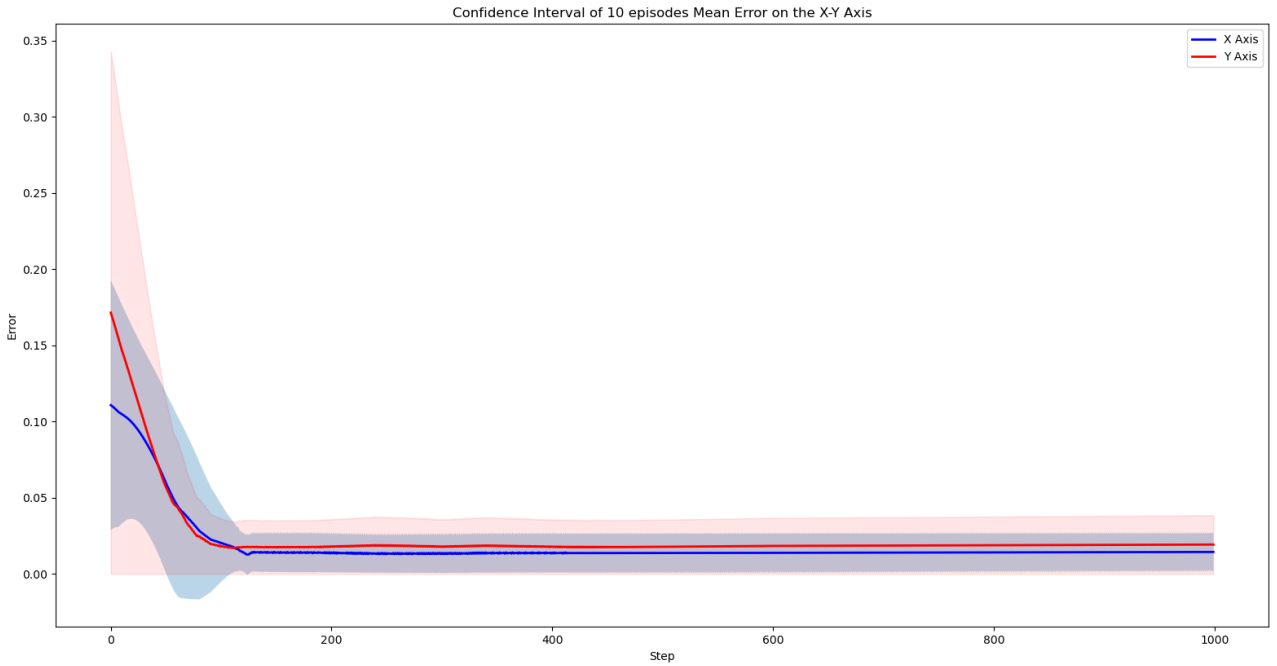
Artificial neural networks and other variables used in the algorithm were created using the Nvidia CUDA infrastructure. Thanks to the CUDA library, calculations are made faster on the computer's graphics card. The technical specifications of the computer used for the training of the algorithm are intel i7 6700HQ 2.60 Ghz processor, 16 GB ddr4 ram, Nvdia 950M graphics card. 3000 episodes are used for the algorithm to perform a successful training. Thus, as a calculation step, at the end of maximum 3,000,000 steps (because maximum training step is 1000), the robot performs a good learning on the targeted region in the workspace. With this computer and hardware used, the training took approximately 14 hours.

Reward Plots





Error Plots


Confidence Interval of 10 episodes Mean Error on the X-Y Axis
X Axis
Y Axis
Error
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0
200
400
600
800
1000
Step

Example Demo

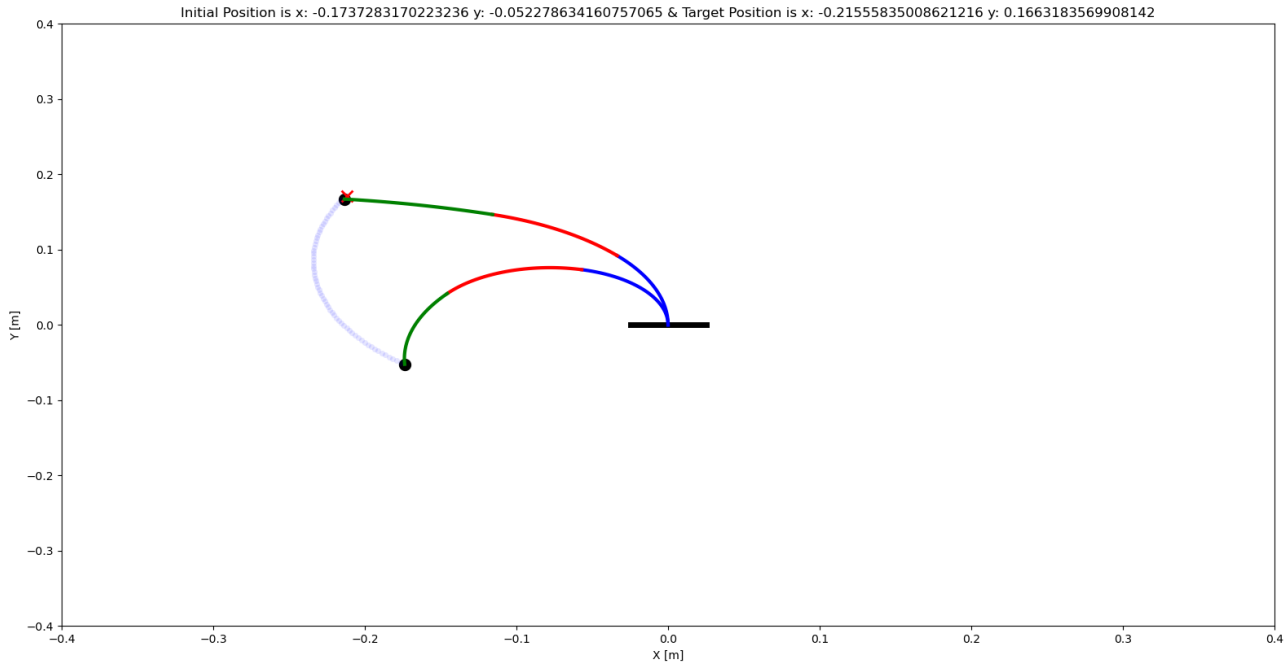
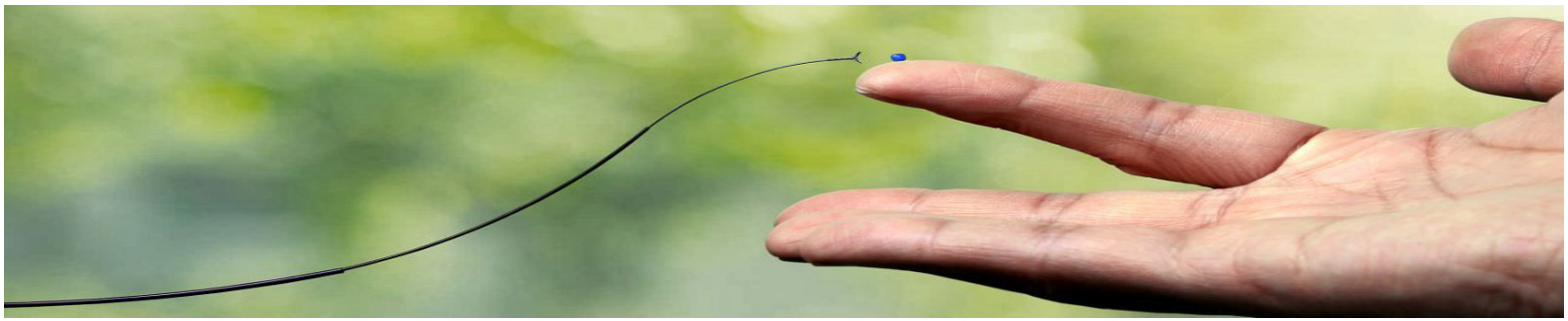


Figure: Click Here

Conclusion

- The forward kinematics model of the continuum robot was built.
- The velocity kinematics model of the continuum robot was built.
- Reinforcement learning environment was implemented.
- Action, states, and reward functions were chosen for the RL algorithm.
- RL (Deep Deterministic Policy Gradient) Algorithm was implemented.
- An appropriate reward function is written to provide point-to-point access.
- Robot is going to the random target point from random starting point successfully after training 3000 episodes.
- Future works will be implemented for other scenarios.



Other Possible Scenarios and Future Works

- Tracking of a randomly moving target
- For high-level control, the problem of the robot creating a path planning in the environment by adding obstacles to its environment can be solved with this algorithm.
- Maybe algorithm can be checked on real continuum robot.