Journal of Intelligent & Robotic Systems (2023) 109: (2023) 109:77
https://doi.org/10.1007/s10846-023-02003-0

REGULAR PAPER

A Reinforcement Learning Approach for Continuum Robot Control

Turhan Can Kargin[1] · **Jakub Kołota**[2]

Received: 22 April 2023 / Accepted: 16 October 2023 / Published online: 22 November 2023

Abstract
Rigid joint manipulators are limited in their movement and degrees of freedom (DOF), while continuum robots possess a continuous backbone that allows for free movement and multiple DOF. Continuum robots move by bending over a section, taking inspiration from biological manipulators such as tentacles and trunks. This paper presents an implementation of a forward kinematics and velocity kinematics model to describe the planar continuum robot, along with the application of reinforcement learning (RL) as a control algorithm. In this paper, we have adopted the planar constant curvature representation for the forward kinematic modeling. This choice was made due to its straightforward implementation and its potential to fill the literature gap in the field RL-based control for planar continuum robots. The intended control mechanism is achieved through the use of Deep Deterministic Policy Gradient (DDPG), a RL algorithm that is suited for learning controls in continuous action spaces. After simulating the algorithm, it was observed that the planar continuum robot can autonomously move from any initial point to any desired goal point within the task space of the robot. By analyzing the results, we wanted to recommend a future direction for research in the field of continuum robot control, specifically in the application of RL algorithms. One potential area of focus could be the integration of sensory feedback, such as vision or force sensing, to improve the robot's ability to navigate complex environments. Additionally, exploring the use of different RL algorithms, such as Proximal Policy Optimization (PPO) or Trust Region Policy Optimization (TRPO), could lead to further advancements in the field. Overall, this paper demonstrates the potential for RL-based control of continuum robots and highlights the importance of continued research in this area.

Keywords Reinforcement Learning · DDPG algorithm · Continuum robot

Highlights

- Implementing the forward kinematics and velocity kinematics model for the planar case of a continuum robot
- Reinforcement learning has been applied as a control algorithm for continuous robot control
- Validating results obtained by applying the Deep Deterministic Policy Gradient (DDPG) reinforcement learning algorithm

1 Introduction

The concept behind continuum robots is inspired by the natural abilities of certain biological organisms, such as snakes, elephant trunks, octopus tentacles, and similar creatures [1]. Unlike traditional rigid-joint manipulators, continuum robots bend individually for each section by creating a sequence of continuous non-joint arcs [2]. After extensive research and development in academic settings, these designs have recently been applied to large-scale grasping, movement, and positioning in industrial contexts, as well as search and rescue operations in confined spaces [8]. They have also gained popularity in surgical applications [9], as they offer adaptable qualities that are particularly advantageous in medical contexts. With the ability to manipulate objects with an infinite number of degrees of freedom (DOF) on small scales, continuum robots provide flexible access for the target and less intrusion for the patient.

Conventional rigid joint manipulators, also known as discrete manipulators, have a limited number of operable, series-independent, rigid connections [2]. Despite their limitations, they are highly practical for many tasks. However, in constrained environments, they may not be able to reach the desired end-effector position, resulting in failure to meet location criteria. Another disadvantage of discrete manipu-

✉ Jakub Kołota
Jakub.Kolota@put.poznan.pl

[1] Izmir, Turkey

[2] Poznan University of Technology, Institute of Automatic Control and Robotics, Piotrowo 3A, Poznan 60-965, Poland

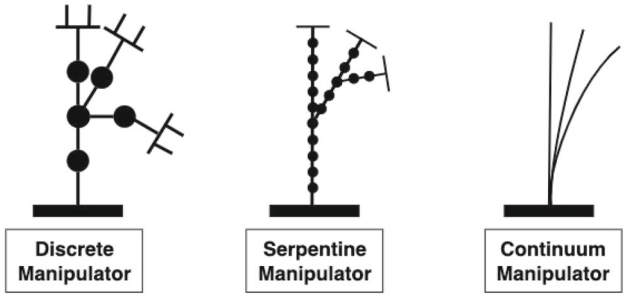

Fig. 1 Class of manipulators[2]

lators is that they require a specific tool, such as a gripper or hand, to manipulate an object. In some cases, it may be preferable to use a manipulator in sections, such as the elephant trunk, to hold objects. Adding more joints to discrete manipulators increases their flexibility and maneuverability, resulting in redundancy and maneuverability. This elevates them to the second category of manipulators, known as serpentine robots, which includes hyper-redundant manipulators. Continuum robots, on the other hand, lack rigid joints and bend continuously along their length, similar to biological bodies and tentacles. This makes continuum robots more maneuverable than discrete and serpentine manipulators, but also makes their kinematics more complex. See Fig. 1.

There are numerous methods for controlling continuum robots, such as PID and predictive control, which are based on explicitly implemented kinematics or dynamics [3–6]. Continuum robots, with their inherent flexibility and high degree of freedom, present unique control challenges that are not adequately addressed by existing techniques. As highlighted by [7], the soft and continuously deformable nature of continuum robots introduces significant complexities in their control. Traditional control methods often rely on precise mathematical models and struggle to handle the nonlinear dynamics and uncertainties associated with continuum robots. Furthermore, these methods are typically designed for rigid robots and do not fully exploit the potential of soft robots' embodied intelligence.

In contrast, Reinforcement Learning (RL) offers a promising alternative for controlling continuum robots. RL algorithms learn from interactions with the environment and adapt their control policies accordingly, making them well-suited to handle the complexities and uncertainties of continuum robots. Moreover, RL does not require a precise model of the robot and can learn effective control policies even in the presence of model inaccuracies. Therefore, we believe that RL holds significant potential for advancing the control of continuum robots and addressing the limitations of existing techniques.

RL is a cost-effective approach for controlling robots or solving complex control problems. It involves training a learnable architecture based on generated reward or penalty values through trial and error. RL has become increasingly popular due to its ability to handle control problems, make challenging sequential decisions, and solve a wide range of issues, even before being combined with deep learning. However, combining deep learning techniques with RL has increased the number of problems that have been solved, as well as their success rate [10]. This combination works better for high-dimensional state and action spaces. Deep reinforcement learning (Deep RL) has been used in many applications to surpass human levels. There have been several significant experiments using deep RL in games, including beating even the best poker players in the world [11, 12], playing video games from pixels [13, 14], and mastering the game of Go with programs like AlphaGo [15]. RL and Deep RL are used in a variety of academic and commercial contexts, including robotics [16–19], autonomous vehicles [20], banking [21], smart grids [22], and the energy sector [23]. RL in robotics allows robots to interact with their environment and learn the optimal path or behavior through trial and error. Instead of explicitly defining the mathematical details of a problem's solution, the control task designer in RL provides feedback in the form of a scalar objective function that measures the robot's performance over one step. There exist multiple RL techniques that can be employed to address the control problem in continuum robots [24–27]. For example, [26] employed a control strategy grounded in deep Q learning to successfully accomplish 2D position reaching. Additionally, [27] proposed a control method that does not rely on a specific model, utilizing Q learning. This method was then applied to a multi-segment soft manipulator in order to achieve 2D position reaching.

RL can be categorized as either model-based or model-free. Two different methods are used in the model-free approach, which are value-based and policy-based. In policy-based methods, we directly train a policy function, while in value-based methods, we train a value function that maps a state to the expected value of being in that state. The foundations of these RL approaches and methods are based on the Markov decision process. The Markov decision process is an approach to predicting the best next state ($s^{'}$) by dynamically computing the rewards ($Q(s,a) = E[R[s,a]]$) of possible action options (a) for discrete-time scholastic events with respect to the current state (s).

The deep deterministic policy gradient (DDPG) algorithm intersects two different methods, combining the ideas of the deterministic policy gradient (DPG) and the deep Q-network [28]. DDPG consists of two deep neural networks, similar to the Actor-Critic method. The actor suggests an action given a state, while the critic predicts whether the action is good or bad with a continuous numeric value given a state and an action. Therefore, DDPG can solve the continuous state and action space. DDPG has become a frequently used algorithm

in the literature [29–32]. For example, it has been used to stabilize the end-effector of a humanoid robot with 27 DOFs during its movement [29]. In another study, a specialized DDPG architecture called the twin-delayed DDPG structure was used to perform an inverse kinematic solution of a 5-DOF robot arm [30]. Detecting whether an object is in the workspace and touching an object in the workspace of a 7 DOF robot was achieved using a DDPG structure in another study [31]. Altitude control has been achieved not only in articulated robots, but also in systems such as unmanned aerial vehicles with the help of DDPG [32].

This paper aims to demonstrate the implementation of a forward kinematic model using a constant curvature representation and a velocity kinematics framework to simulate the planar continuum robot environment. The objective is to solve a 2D position-reaching task using the RL approach to control a continuum robot in a simulation environment. The DDPG algorithm is employed to learn controls in continuous action spaces, which is crucial for the intended control mechanism. The algorithm's features were examined through several simulations, and the results are presented. The simulations involved the continuum robot reaching a goal state while minimizing distance error. To our knowledge, the combination of the DDPG algorithm with our chosen kinematic model is a unique contribution to the field. This research not only underscores the immediate applicability of our approach but also paves the way for its potential expansion to more complex robotic systems in the future.

2 Kinematics Modeling of Continuum Robot

Real-world applications of continuum robotics require the usage of robot shape and motion models. To simulate the control based on RL described in this paper, it's necessary to define the forward kinematic model, which provides the correct shape of the continuum robot, and the velocity kinematics, which provide the correct motion. This section provides an overview of forward and velocity kinematics design for continuum robots.

The forward kinematics of continuum robots contains three spaces that can be decomposed into two mappings, which are a mapping between the robot's actuator space and the configuration space, and a mapping between the configuration space and the task space [1]. You can see the mapping in Fig. 2.

Several kinematic frameworks have been presented for continuum robot modeling. [33] discussed some of these models, such as variable curvature representation, constant curvature representation, pseudo-rigid body, and modal approach, in detail and shared the models implemented as open source. The planar constant curvature representation is chosen as the kinematic model in this paper due to its ease of implementation and a novel choice for the kinematic model in the context of RL-based continuum robot control. This simple model captures the key features of continuum robots, like their continuous structure and high degree of freedom. By focusing on the basics of continuum robot control, we could test our RL approach. The planar model is also relevant to real-world scenarios where continuum robots operate in two-dimensional spaces. The insights we gained from this study apply not only to the planar model, but also to more complex continuum robots.

2.1 Forward Kinematics with D-H Method

In this section of the paper, the solution for the geometric shape of a three-section continuum robot using the Denavit-

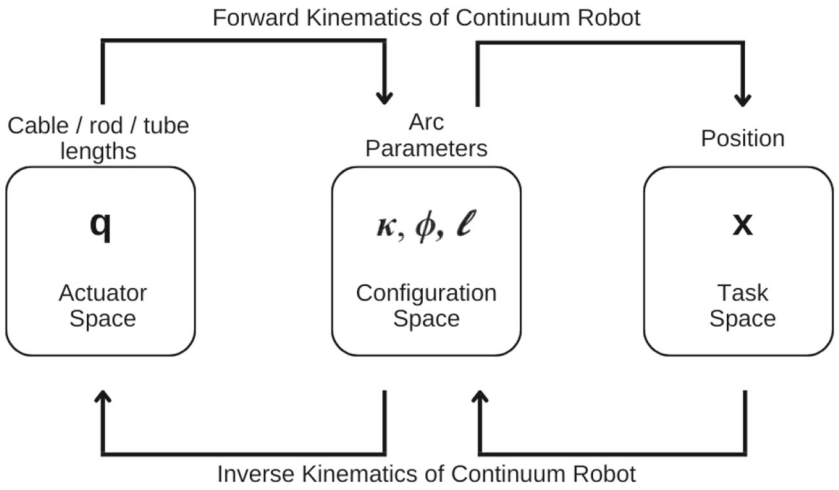

Fig. 2 Three space of continuum robot and the mapping directions

Hartenberg (DH) method, which is one of the forward kinematics frameworks, is presented. The purpose of deriving the forward kinematics model of the continuum robot is to calculate the new position of the robot as a result of the actions taken by the robot in the control problem with the RL approach. In order to infer the shape of continuum designs, it is necessary to obtain a description of their movement. Because of the continuous nature of the design, joint angles and joint lengths do not provide a simple and physically feasible description of the manipulator's shape. Instead, a kinematic model that describes the manipulator's geometry using curvatures can offer a more realistic and understandable description. When using a constant curvature approach, a planar curve can be thought of as moving in three coupled movements.

1. rotation by an angle θ
2. translation by an amount of $\| x \|$, and
3. rotation by the angle θ once more,

where **x** is referred to as the position vector of the curve's endpoint. With the above information, we have a discrete motion description of how the curve moves. [34] obtained the homogeneous transformation matrix in Eq. 1 using a modified D-H approach for forward kinematic analysis, just as is applied for conventional rigid joint manipulators.

$$A_0^3 = A_0^1 A_1^2 A_2^3 = \begin{bmatrix} \cos(\kappa l) & -\sin(\kappa l) & 0 & \frac{1}{\kappa}\{\cos(\kappa l) - 1\} \\ \sin(\kappa l) & \cos(\kappa l) & 0 & \frac{1}{\kappa}\sin(\kappa l) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

The homogeneous transformation matrix for the curve from frame 0 to frame 3 can be written in terms of the curvature κ and the total arc length l. This representation gives us the shape of only the first section. In order to know the endpoint position of the three-section robot, the product of the matrix of each partition must be calculated. The frames for a three-section continuum robot can be setup as a combination of three sections. The homogeneous transformation matrix used for forward kinematics is the product of three matrices of the form of Eq. 1. Therefore, the transformation matrix from frame 0 to frame 9 is shown in Eq. 2.

$$\begin{aligned} A_0^9 &= A_0^3 A_3^6 A_6^9 \\ &= \begin{bmatrix} \cos(\omega_1+\omega_2+\omega_3) & -\sin(\omega_1+\omega_2+\omega_3) & 0 & A_{14} \\ \sin(\omega_1+\omega_2+\omega_3) & \cos(\omega_1+\omega_2+\omega_3) & 0 & A_{24} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{2}$$

where κ_i and l_i are the curvature and total arc length, respectively for section i and $\omega_i = \kappa_i l_i$ for $i = \{1, 2, 3\}$. The following are the elements of the matrix that were not previously defined:

$$\begin{aligned} A_{14} =& \frac{1}{\kappa_1}\{\cos\omega_1 - 1\} + \frac{1}{\kappa_2}\{\cos(\omega_1+\omega_2) - \cos\omega_1\} \\ &+ \frac{1}{\kappa_3}\{\cos(\omega_1+\omega_2+\omega_3) - \cos(\omega_1+\omega_2)\} \\ A_{24} =& \frac{1}{\kappa_1}\sin\omega_1 + \frac{1}{\kappa_2}\{\sin(\omega_1+\omega_2) - \sin\omega_1\} \\ &+ \frac{1}{\kappa_3}\{\sin(\omega_1+\omega_2+\omega_3) - \sin(\omega_1+\omega_2)\} \end{aligned} \tag{3}$$

Equation 3 captures the spatial relationships and transformations as the continuum robot moves in the planar space. The components A_{14} and A_{24} of the transformation matrix are derived from the robot's configuration and kinematic constraints. A_{14} represents the x-coordinate transformation and A_{24} represents the y-coordinate transformation. Note that the total arc length for $i = \{1, 2\}$ must be used so that Section 3 is properly positioned, but any arc length can be used for the final section depending on where the point of interest lies in the section. The total arc length for the final section gives the kinematics in terms of the end point. To provide a clearer understanding of the planar continuum robot used in our study, we present the kinematics parameters in Fig. 3 for a single section of the continuum robot. r defines the radius of the resulting arc, which is equal to $\frac{1}{\kappa}$.

2.2 Velocity Kinematics

Positions are not enough to create movement. Therefore, velocities are needed for a better interaction. Given curvature velocities, we can calculate how fast the endpoint moves and how fast each curvature must move to guarantee a desired endpoint velocity using the velocity kinematics framework.

The velocity kinematics of a continuum robot can be expressed similar to traditional kinematic analysis as Eq. 4

$$\dot{x} = J\dot{q} \tag{4}$$

where $x \in \mathbb{R}^{m\times 1}$ is the vector of the task space that corresponds to position and / or orientation, and the dot notation represents differentiation with respect to time, denoting the time derivative of the position and/or orientation variables. For the sake of simplicity, we have not taken orientation into account in our current work. While orientation is an important aspect of robot control, our primary focus in this study is on position control. By simplifying the control task to exclude orientation, we aim to establish a foundational understanding of the robot's planar movement dynamics using our RL

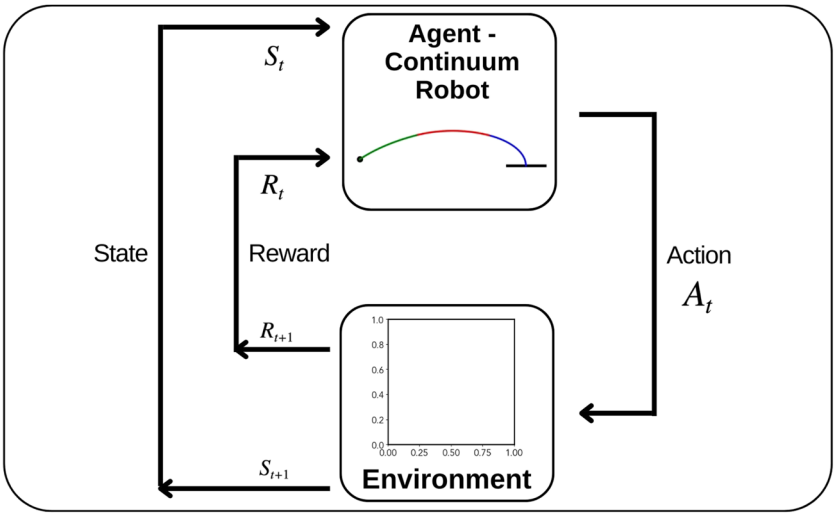

Fig. 3 Illustration of specified kinematics parameters

approach. However, we acknowledge that in real-world applications, orientation can play a crucial role, and we plan to incorporate it in our future work to provide a more comprehensive control solution. To be clear for our three-section planar robot:

$$\dot{\mathbf{q}} = [\dot{\kappa_1}, \dot{\kappa_2}, \dot{\kappa_3}]^T \ , \ \ \dot{\mathbf{x}} = [\dot{x}, \dot{y}]^T \tag{5}$$

The matrix J called as Jacobian matrix, and is a function of the 'joint' variables $\mathbf{q}$. The Jacobian matrix represents the linear sensitivity of the tip point of the continuum robot velocity $\dot{x}$ to the curvature velocity $\dot{\kappa}$, and it is a function of curvature variables κ. When taking partial derivatives of forward kinematic equations, as in Eq. 6, one can obtain the components of the Jacobian J_{ij}:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} J_{11} & J_{12} \\ J_{21} & J_{22} \end{bmatrix} \begin{bmatrix} \dot{q_1} \\ \dot{q_2} \end{bmatrix} \tag{6}$$

Equation 7 will result from writing it using the chains rule.

$$\begin{aligned} \frac{dx}{dt} &= \frac{\partial x}{\partial q_1}\frac{dq_1}{dt} + \frac{\partial x}{\partial q_2}\frac{dq_2}{dt} \\ \frac{dy}{dt} &= \frac{\partial y}{\partial q_1}\frac{dq_1}{dt} + \frac{\partial y}{\partial q_2}\frac{dq_2}{dt} \end{aligned} \tag{7}$$

If this case in Eq. 7 applied to the continuum robot, Eq. 8 is obtained.

$$\begin{aligned} \frac{dx}{dt} &= \frac{\partial x}{\partial \kappa_1}\frac{d\kappa_1}{dt} + \frac{\partial x}{\partial \kappa_2}\frac{d\kappa_2}{dt} + \frac{\partial x}{\partial \kappa_3}\frac{d\kappa_3}{dt} \\ \frac{dy}{dt} &= \frac{\partial y}{\partial \kappa_1}\frac{d\kappa_1}{dt} + \frac{\partial y}{\partial \kappa_2}\frac{d\kappa_2}{dt} + \frac{\partial y}{\partial \kappa_3}\frac{d\kappa_3}{dt} \end{aligned} \tag{8}$$

We can extract the Jacobian Matrix elements as below:

$$J = \begin{bmatrix} \frac{\partial x}{\partial \kappa_1} & \frac{\partial x}{\partial \kappa_2} & \frac{\partial x}{\partial \kappa_3} \\ \frac{\partial y}{\partial \kappa_1} & \frac{\partial y}{\partial \kappa_2} & \frac{\partial y}{\partial \kappa_3} \end{bmatrix} \tag{9}$$

Finally, we obtain Eq. 10.

$$\begin{bmatrix} \frac{dx}{dt} \\ \frac{dy}{dt} \end{bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial \kappa_1} & \frac{\partial x}{\partial \kappa_2} & \frac{\partial x}{\partial \kappa_3} \\ \frac{\partial y}{\partial \kappa_1} & \frac{\partial y}{\partial \kappa_2} & \frac{\partial y}{\partial \kappa_3} \end{bmatrix} \begin{bmatrix} \frac{d\kappa_1}{dt} \\ \frac{d\kappa_2}{dt} \\ \frac{d\kappa_3}{dt} \end{bmatrix} \tag{10}$$

Robotics and control theory often rely on Jacobian matrices. In short, a Jacobian helps to explain how two different system representations interact dynamically. In our specific example, this connection is based on both position and curvature. The aim of velocity kinematics is to describe the motion of an endpoint. However, the motion of the endpoint is heavily restricted by curvatures. As a result, describing the motion of the endpoint also means describing the states of motion at those curvatures.

The velocity kinematics plays a pivotal role in generating movement. In our approach, the action is defined as the derivative of curvature with respect to time ($\dot{\kappa}$). By incorporating the Jacobian matrix derived from the forward kinematics, we are able to calculate the movement at each time step, thus facilitating a smooth and controlled movement of the robot. This integration of velocity kinematics is crucial in creating a dynamic and responsive control system.

3 Reinforcement Learning

Both humans and animals learn best through trial and error. They interact with their surroundings, make small changes,

and observe the outcomes. For example, consider how a newborn learns to walk. It may take a few days of attempting different actions before they can stand up and walk without falling. During this process, the baby learns from rewards and penalties - they are rewarded for making progress and penalized for falling [35]. This incentive system is innate to humans and animals and encourages us to pursue behaviors that result in positive rewards and avoid behaviors that result in negative ones. This type of learning is called reinforcement learning and is modeled mathematically based on human and animal learning patterns. In a reinforcement learning framework, an agent interacts with the environment E by receiving observations s_t and executing actions a_t. After each t time step, the agent receives a scalar reward r_t and a new state s_{t+1}. Figure 4 illustrates this process.

Algorithms for reinforcement learning can be broadly divided into two groups. Model-based and model-free approaches are the names for these. Model-based algorithms are constructed using the environment's prior knowledge or a mathematical model that represents it. Algorithms for model-free reinforcement learning are unaware of their environment. By simulating the environment with a Markov decision process, which is typically a random process, algorithms are created and learned. The agent investigates the environment, acquires knowledge about them, and then chooses the best course of action. This procedure is an example of reinforcement learning that involves trial and error as well as reward accumulation. Today, reinforcement learning is used in many different fields, and one of the key factors in its success is that it doesn't need an environment model. It interacts with the environment in this way to find solutions to a variety of engineering issues. However, it has often been used in robotics and control problems.

Basic RL algorithms such as Q learning and Sarsa learning are called tabular methods in the literature [37]. For discrete state and action spaces, these techniques are appropriate. The control problem's state space and the control signal are actually stated in continuous time. The derivative values of the curvature, which is our control signal, in the continuum robot control problem discussed in this paper are defined continuous. By discretizing the continuous state and action spaces, Q learning and Sarsa learning algorithms can be used as practical implementations of reinforcement learning, however in reality, the errors caused by discretization constitute a concern in control applications. Another problem with the tabular method of reinforcement learning is the computational cost. Increasing the size of our discrete state and action space leads to an increase in computational complexity and computational cost.

As the amount of accessible data grows exponentially every year and the hardware quality of computers improves (especially GPUs), deep learning methods have started to develop rapidly, and thus deep reinforcement learning was born from the combination of artificial neural networks with reinforcement learning [10]. Thus, algorithms that can operate in continuous state and action space have been developed. The application of RL to the control of continuum robots offers several potential advantages and disadvantages compared to other existing control methods. Pros of RL for Continuum Robot Control as follows:

- Adaptability to changing environments and system dynamics.
- Flexibility for complex and nonlinear systems.
- Autonomous learning without relying on human-engineered controllers.
- Scalability to high-dimensional state and action spaces.
- Generalization to similar tasks or robot platforms.

Cons of RL for Continuum Robot Control as follows:

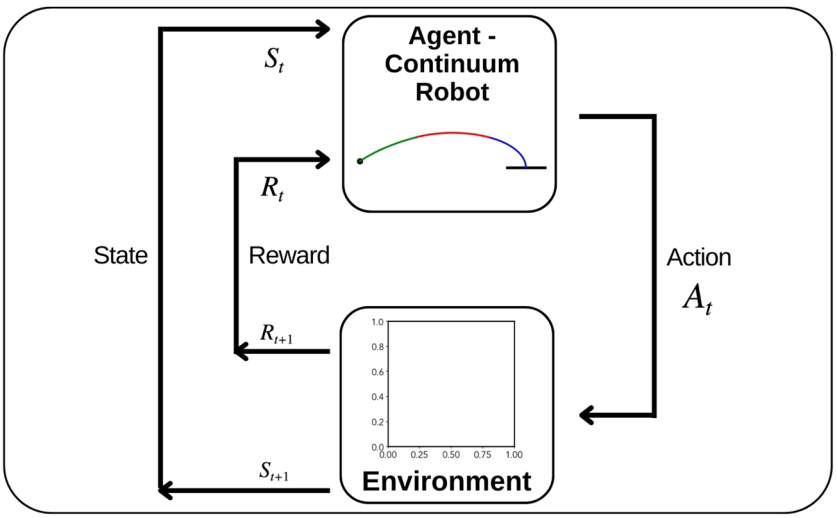

Fig. 4 Agent and Environment interaction in Reinforcement Learning

- Sample inefficiency, requiring numerous interactions with the environment.
- Exploration-exploitation trade-off, challenging in continuous and flexible systems.
- Safety concerns due to trial and error learning.
- Hyperparameter tuning for optimal performance.
- Lack of interpretability in learned policies.

In conclusion, while RL offers potential advantages for the control of continuum robots in terms of adaptability, flexibility, autonomy, scalability, and generalization, it also has limitations in terms of sample efficiency, exploration-exploitation trade-off, safety concerns, hyperparameter tuning, and interpretability. Careful consideration of these pros and cons is necessary when applying RL to the control of continuum robots, and further research and development are needed to address the challenges and improve the performance and safety of RL-based control methods for continuum robots.

3.1 Deep Reinforcement Learning

Deep reinforcement learning combines deep neural networks with reinforcement learning algorithms to solve continuous state control problems. While table-based algorithms like Q-learning do not work well for these types of problems, deep reinforcement learning methods are often used. One popular algorithm is deep Q-learning, but it is not effective for continuous action space due to its reliance on a single neural network. To address this issue, new algorithms with two neural networks that work well for both continuous state space and continuous action space have been introduced [28, 38, 39].

As mentioned earlier, the environment of the continuum robot is characterized by continuous control action and state. This is an important factor to consider when choosing a reinforcement learning algorithm. To address this issue, we have employed the DDPG algorithm, which is capable of handling continuous action and state space. The DDPG algorithm uses neural networks approximations to represent the actor-critic algorithm. Based on the policy π, which is determined by the present state, the actor chooses the current action. The critic takes the current state and action as input, and is described by an action-value function **Q**. In this paper, a deep neural network with several layers is used to approximate the functions **Q** and π.

A DDPG algorithm is a model-free off-policy algorithm formed by combining the deterministic policy gradient and deep Q Learning. It is built on DPG, which can operate over continuous action spaces, and uses Experience Replay and slow-learning target networks from deep Q learning. To give a detailed mathematical background of the DDPG algorithm, robot performs an action $a_t \in \Re^N$ in the environment where the agent is in discrete time steps and records s_{t+1}, r_t. The aim of the agent is to maximize the sum of all rewards $R_t = \sum_{t=1}^{T} \gamma^{(t-1)} r_t\,(s_t, a_t)$ that it will create in the future. Depending on our π policy, the action we take in a given state and the reward values we will obtain accordingly change. The expression of the policy can be deterministic or stochastic. Equation 11 describes the expected return of taking an action a_t in a certain state s_t, expressed by following a policy π. This return is represented by the action value function [28].

$$Q^{\pi}\,(s_t, a_t) = E\,[R_t \mid s_t, a_t] \tag{11}$$

Extending Eq. 11 with the Bellman equation as a recursive approximation, we obtain Eq. 12 for deterministic politics.

$$Q^{\pi}\,(s_t, a_t) = E\left[r\,(s_t, a_t) + \gamma\, Q^{\pi}\,(s_{t+1}, \pi\,(s_{t+1}))\right] \tag{12}$$

The $\pi\,(s_{t+1})$ policy in the off-policy algorithms like Q learning algorithm is the $\varepsilon-$ greedy approach. Parametric writing of Eq. 12 to minimize the loss is shown in Eq. 13.

$$L\left(\theta^{Q}\right) = E\left[\left(Q\left(s_t, a_t : \theta^{Q}\right) - y_t\right)^2\right] \tag{13}$$

where

$$y_t = r\,(s_t, a_t) + \gamma\, Q\left(s_{t+1}, \pi\,(s_{t+1}) : \theta^{Q}\right) \tag{14}$$

It is possible to approximate Eq. 13 with multi-layer neural networks. Equation 13 cannot be calculated in high-dimensional action space with the Q learning algorithm. The actor-critic approach is used to solve this problem. The deterministic policy gradient maps states to actions using a deterministic parametric policy function $\pi\,(s \mid \theta^{\mu})$. Using the Critic $Q(s, a)$ Bellman equation, Q is calculated as in learning. The actor updates his parameters using a gradient-based update using Eq. 15.

$$\begin{aligned}\nabla_{\theta^{\pi}} J \;=\; & E\left[\nabla_a Q\left(s, a \mid \theta^{Q}\right) | s = s_t,\right. \\ & \left. a = \pi\,(s_t)\, \nabla_{\theta_{\pi}} \pi\left(s_t \mid \theta^{\pi}\right)\right| s = s_t\Big]\end{aligned} \tag{15}$$

Equation 15 is called the policy gradient. The $\pi\,(s_t \mid \theta^{\pi})$ function is created with the deterministic policy gradient algorithm and the deep Q learning algorithm approach. Deep neural networks are used to approximate the $\pi\,(s_t \mid \theta^{\pi})$ function. The values $D = (s_t, a_t, r_t, s_{t+1})$ are saved in the created memory. The s_t states are the current position and goal position values for our robot. These different quantitative characteristics in the data cause different scaling. By providing normalization of the data formed in the memory, it is ensured that the artificial neural networks to be trained

Table 1 Noise parameters of the DDPG Algorithm

Parameter	Value
μ	0.00
θ	0.15
σ	0.20

optimize their parameters. The biggest challenge in continuous action space is to discover the environment in which the agent is located by applying appropriate actions. Since the DDPG algorithm is an off-policy, it allows us to create the discovery problem independently of the algorithm. To provide exploration, an Ornstein-Uhlenbeck process is added to generate noise to the policy value produced by the actor, as demonstrated in Eq. 16.

$$\pi'(s_t) = \pi\left(s_t \mid \theta_t^{\pi}\right) + N \tag{16}$$

You can find the parameters of the Ornstein-Uhlenbeck process in Table 1.
where μ represents the mean value of the process, θ is a positive constant that determines the speed of mean reversion, and σ is a positive constant that determines the volatility of the process.

3.1.1 Network Structure of the Reinforcement Learning Approach

In our endeavor to develop a robust control mechanism for planar continuum robots, we have adopted the DDPG algorithm, a model-free, off-policy actor-critic algorithm that operates in continuous action spaces. This section delineates the structure of the actor and critic networks and the synergies between them, offering a comprehensive insight into our RL approach. The RL approach is grounded on a dual neural network architecture comprising two primary components: the Actor network and the Critic network. This architecture is designed to seamlessly integrate the continuous control capabilities of the actor network with the value estimation proficiencies of the critic network, fostering a harmonized learning environment.

The Actor network, responsible for determining the optimal policy, is structured as a feedforward neural network. It maps the current state of the robot to a specific action, aiming to maximize the expected cumulative reward. The network consists of:

- **Input Layer:** Accepts the state variables, initiating the process of policy determination.
- **Hidden Layers:** Incorporates four hidden layers with 128, 256, 256, and 128 neurons respectively, facilitating the learning of complex patterns in the data.
- **Output Layer:** Utilizes a tanh activation function to generate actions within a specified range, aligning with the continuous action space of the problem.

The Critic network, on the other hand, evaluates the value of the chosen action in the current state, aiding in the optimization of the policy developed by the actor network. Its structure is as follows:

- **Input Layer:** Receives both the state variables and the actions proposed by the actor network as inputs.
- **Hidden Layers:** Comprises four hidden layers with a similar configuration to the actor network, enhancing the ability to learn intricate relationships between states and actions.
- **Output Layer:** Produces a single output representing the estimated Q-value of the chosen action-state pair.

ReLU (Rectified Linear Unit) activation functions are employed in the hidden layers of both networks, promoting efficient learning by mitigating the vanishing gradient problem. The learning process is facilitated through the backpropagation algorithm, complemented by the Adam optimizer, fostering efficient convergence during the training phase. Training involves minimizing a loss function that represents the difference between the predicted and target Q-values, a process orchestrated by the critic network. To prevent overfitting and foster model generalization, batch normalization and dropout techniques have been implemented. A set of optimized hyperparameters, including actor learning rate of 0.0001, critic learning rate of 0.0003 and a

Table 2 Hyperparameters of the DDPG Algorithm

Parameter	Value
Optimizer	Adam
Learning Rate 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	1000
Soft Update (τ)	0.001

discount factor of 0.99, were determined through preliminary experiments to ensure stable and efficient learning. The performance of the networks was assessed based on convergence speed and the precision in achieving the desired goal states in the simulation environment. Further insights into performance metrics are elucidated in the "Simulations and results" section. The Table 2 displays the hyperparameters to summarize the discussion in this section.

3.2 Environment Design

The interaction between the environment, actors, and critics is well shown in the DDPG diagram in Fig. 5. In DDPG based reinforcement learning, the environment responds to an actor's action by sending the critic an observation along with a reward. The environment's internal state encompasses all observations made, and the reward reflects the action's success. To utilize this approach, two elements are required. The first is the actor and critic networks, which are two distinct deep neural networks that determine the action and Q value for the robot. The second is the environment, which is a task or simulation that the Actor-Critic must solve in the robot's task space. This section provides details on our design of the environment for implementing the DDPG algorithm, which is an essential component of the process.

The aim of this task is to achieve the 2D position reach by moving a three-section continuum robot from its initial state q_{ini} to the goal state q_{goal}. The initial position of the robot, denoted as (x_{ini}, y_{ini}), and the target position, denoted as (x_{goal}, y_{goal}), are randomly selected within the task space, which is illustrated in Fig. 7. The task space is defined for a continuum robot with a section length of 0.1 $[m]$ and curvature values ranging from $-4[\frac{1}{m}]$ to $16[\frac{1}{m}]$. Additionally, the base of the continuum robot is mounted at the point (0,0).

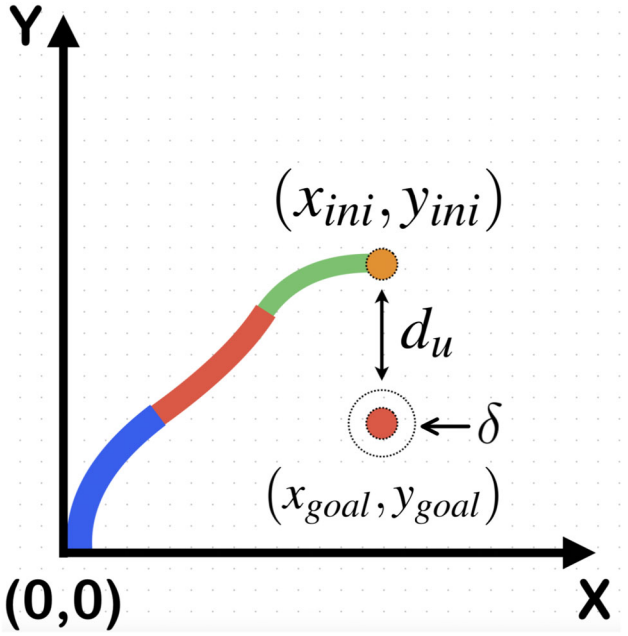

Fig. 6 Three-section continuum robot with initial (orange) and goal (red) state, goal tolerance δ. First section (blue), middle section (red) and third section (red)

Figure 6 is included in the paper to provide readers with a visual representation of the task that the robot is required to perform. In this figure, d_u represents the Euclidean distance between the robot's current state and the target point, while δ represents the tolerance or target circle that the robot must enter in order to reach the target. This graphical representation helps readers better understand the criteria for successful completion of the task (Fig. 7).

The environment specification is a crucial component of the reinforcement learning methodology. In our study, we directly apply the forward and velocity kinematics of the robot, as covered in Section 2, to describe the planar con-

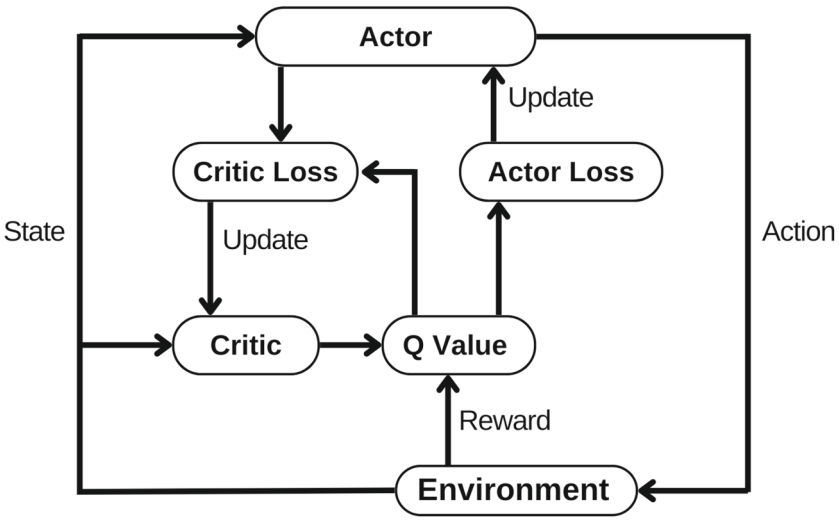

Fig. 5 Schematic diagram of Deep Deterministic Policy Gradient

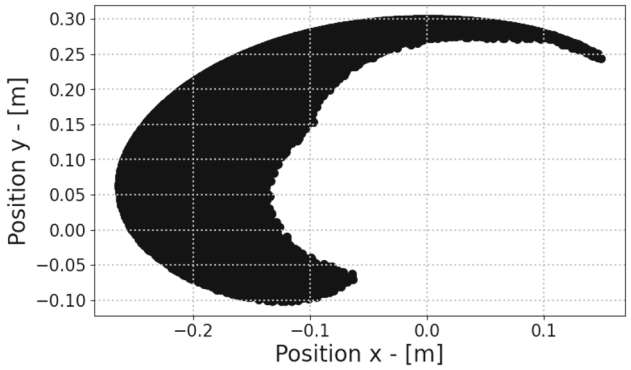

Fig. 7 Task Space of three-section continuum robot

tinuum robot in the environment. The state consists of the current position (x and y) and the goal position (x_{goal} and y_{goal}) of the continuum robot in the planar coordinate system. The state space is designed to be within the task space of the robot. As the continuum robot moves in the environment, it will have three different curvature values (κ) since it is a three-section robot. Therefore, we use $\dot{\kappa}$ for each section as the action space, with each $\dot{\kappa}$ limited to the range of $[-1, +1]$.

To complete the environment, the reward function must be designed. The proper design of the reward function is crucial for the agent to take meaningful action in reinforcement learning problems. Given that our aim is to control the continuum robot for point-to-point access, it makes sense to create a reward function that calculates the distance to the goal position at each time step. To establish the reward function, we started by computing the Euclidean distance to the goal state utilizing Eq. 17. The resulting distance value was then multiplied by negative one, as demonstrated in Eq. 18. This function multiplies the Euclidean distance between the robot's current state and the goal state by negative one to increase the reward value as the robot gets closer to the target. The reward is negative to discourage the robot from deviating from the goal state, with the penalty increasing as the distance increases. This reward function is the foundation of the RL algorithm for controlling the continuum robot, and shapes its behavior during the learning phase.

$$d_u = \sqrt{(x - x_{goal})^2 + (y - y_{goal})^2} \tag{17}$$

$$r = -1 \cdot (d_u)^2 \tag{18}$$

4 Simulations and results

This section discusses the results of the RL algorithm used in simulations. At the start of each simulation episode, the robot is randomly spawned within the task space. The base of the continuum robot is mounted at point (0,0), and each section of the three-section robot starts from a random position relative to this point. The simulation uses a time step of ΔT equal to 0.1 s. To accomplish the intended task, the robot was trained using the DDPG algorithm with the keras-rl framework [36].

To successfully train the algorithm, at least 5000 episodes are required, with a maximum of 1000 steps per episode to prevent the robot from getting stuck. As a result, the robot needs a maximum of 5,000,000 steps to perform good learning on the targeted region in the workspace. After training the continuum robot on the NVIDIA TESLA P40 Graphical Processing Unit (GPU) accelerator using cudatoolkit version 11.2 and cudnn version 8.1.0 for approximately 14 hours, the 2D position reach task was achieved. To effectively train the neural networks of Actor and Critic, it is important to input appropriate hyperparameters.

A 250-episode moving average of the rewards, along with the rewards received for each episode, were saved to evaluate the performance of the RL algorithm (See Fig. 8). The use of a 250-episode moving average facilitates the interpretation of the plots by smoothing out any fluctuations.

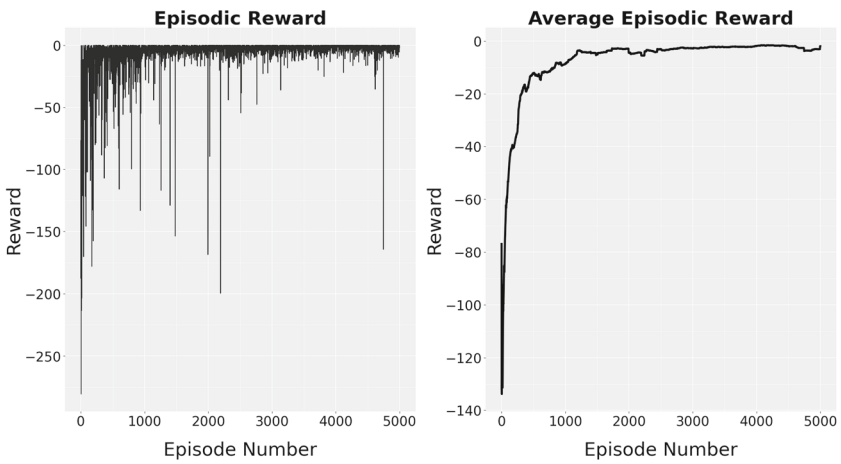

Fig. 8 Moving average and episodic reward per episode

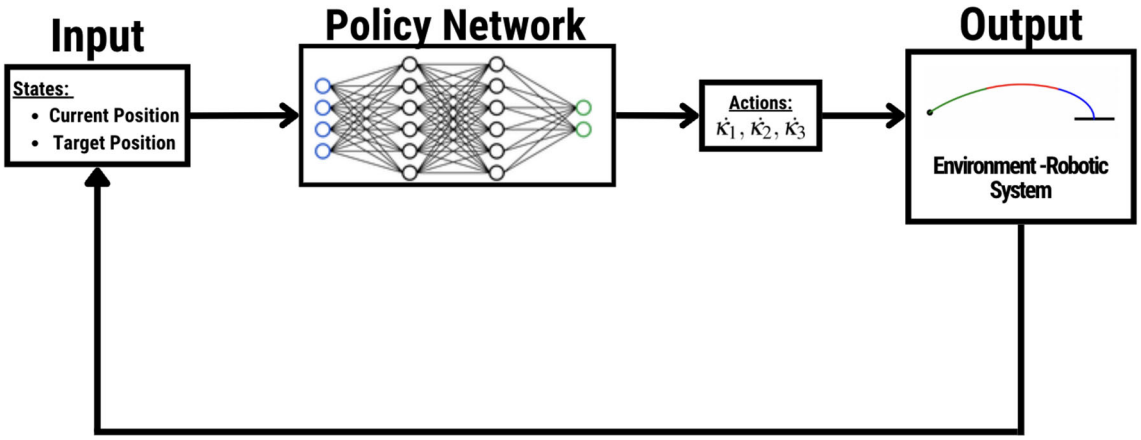

Fig. 9 General control scheme

After the training phase was completed, the neural network weights were saved and made ready for use in the control application during the testing phase. The control scheme for the robot is presented in Fig. 9, where an actor network, also known as the policy network, takes the current and goal states as inputs to produce the action $\dot{\kappa}$ for each of the three robot sections at every time step.

Figure 10 depicts some examples of the robot's motion on the X-Y plane during test simulations. To evaluate the performance of the control algorithm, multiple simulations were conducted. Figure 11 (a) and (b) illustrate the total error value and the error values for the X-axis and Y-axis at each step of the continuum robot, respectively. The confidence bands for these figures provide a statistical measure of the consistency of the results across the multiple simulations. The figures demonstrate that the robot is capable of reaching the goal point and maintains its position in subsequent simulation steps. As a result, the error converges to nearly zero after a certain point.

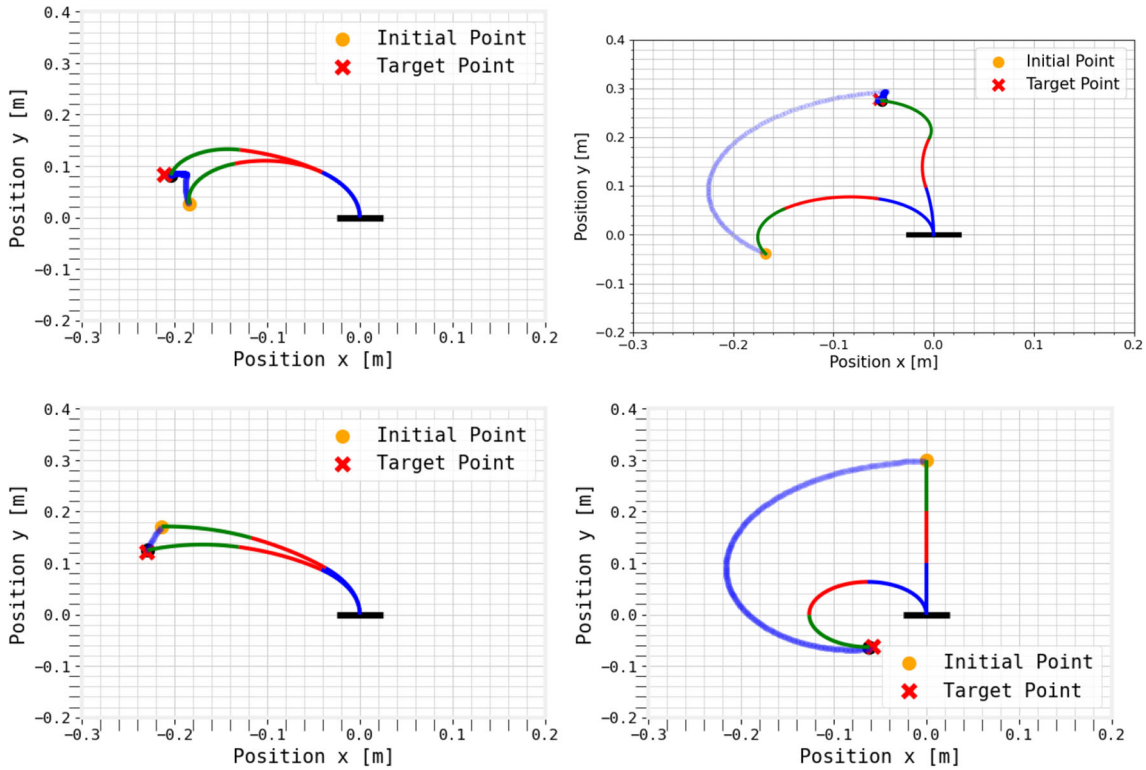

Fig. 10 Frames of a few sample episodes with the robot's trajectory

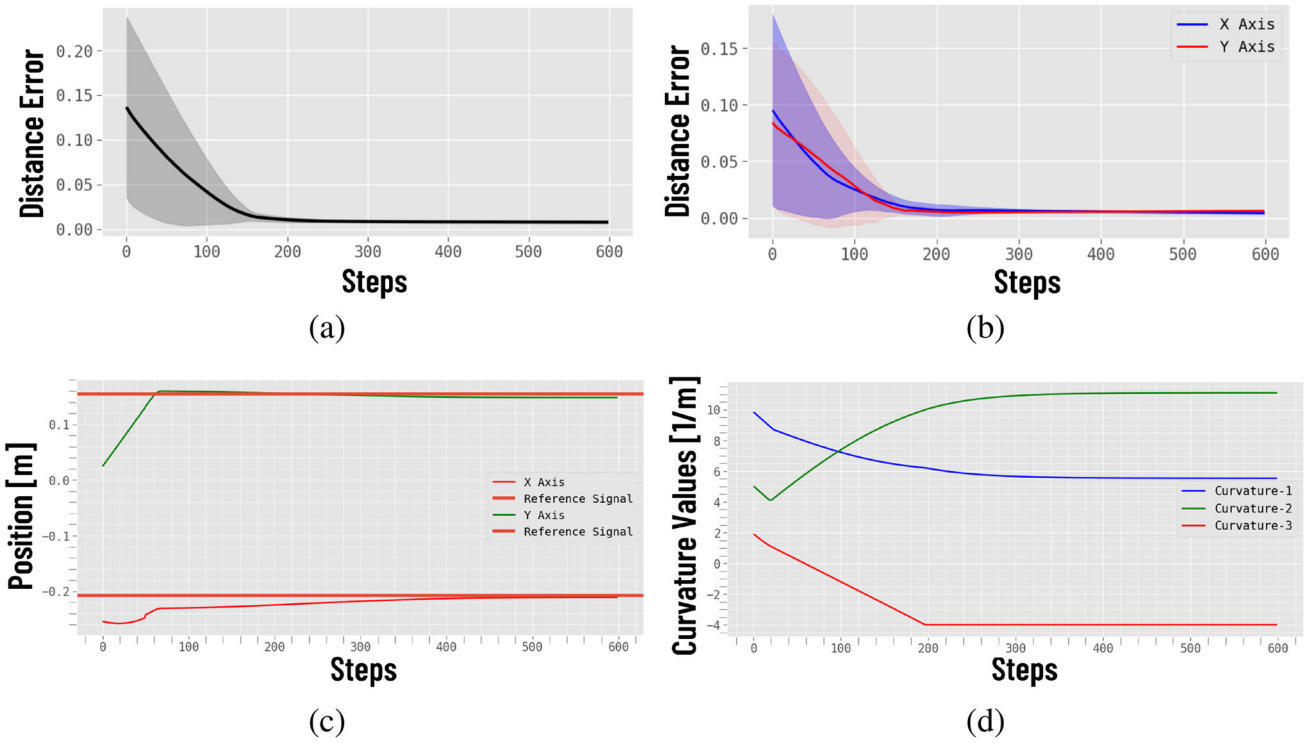

Fig. 11 Total error of the continuum robot (a), error in X and Y axis (b), position in X and Y axis (c), curvature values for each step (d)

Further analysis of the results can be observed in Fig. 11 (c) and (d) that are obtained from single test simulation. Figure 11 (c) shows the position for the X-axis and Y-axis at each step, providing additional insight into the robot's trajectory. Figure 11 (d) displays the curvature value for the three different robot sections at each step during a single simulation. These figures offer a more detailed understanding of the robot's movement and provide evidence of the effectiveness of the control algorithm in achieving the desired results. Overall, the significant point of using RL for a continuum robot is its ability to enable autonomous learning and adaptability in the environment, providing a promising approach for advancing the capabilities and applications of continuum robots.

5 Conclusions

Reinforcement Learning (RL) has evolved from being a term in psychology to becoming a common method of learning in computer science. RL is often observed in nature, such as when a cheetah runs at high speeds hours after birth without any instruction. In summary, this paper has demonstrated the application of reinforcement learning in the context of the planar continuum robot control. This study presented a pioneering approach to continuum robot control by melding a specific forward kinematics and velocity kinematics model with an RL control algorithm. The results underscore the efficacy of our method, especially in the context of the planar continuum robot. While our immediate findings are promising, they also pave the way for future research. The novelty of our approach is evident in its adaptability and potential scalability to more complex robotic systems. As the field of robotics continues to evolve, we believe that such integrative methods will play a pivotal role in addressing the multifaceted challenges of robot control. The use of the constant curvature representation for forward kinematics to describe a shape as well as velocity kinematics for the robot's motion allowed for the creation of an environment for RL to learn the motion of the robot from an initial point to a goal point. Traditional reinforcement learning algorithms are not suitable for this problem due to its continuous state and action spaces, which led to the application of the DDPG algorithm for the control of the continuum robot. The paper also discussed the selection of appropriate choices for the reward function, state space, and action space, which are essential components of reinforcement learning systems. Tuning the hyperparameters of the two neural networks used in the DDPG algorithm was also highlighted as necessary to obtain a suitable solution. The results of the simulations show that the DDPG algorithm was able to learn the control of the continuum robot and move it from an initial point to a goal point efficiently. Our future work involves extending our current model to

include a spatial movement platform and orientation, which will allow us to capture the full complexity of continuum robot control and make our model more realistic. We will also investigate the use of other reinforcement learning algorithms, like PPO or TRPO, and integrate sensory feedback to enhance the robot's ability to navigate complex environments. Overall, this paper contributes to the growing body of knowledge on reinforcement learning and its application in robotics, particularly in the control of continuum robots. The results demonstrate the potential of RL-based control of continuum robots and highlight the importance of continued research in this area to achieve significant breakthroughs in robotics technology.

Acknowledgements This work was supported by the statutory grant No. 0211/SBAD/0121.

References

1. Webster, R.J., Jones, B.A.: Design and kinematic modeling of constant curvature continuum robots: a review. Int. J. Robot. Res. **29**(13), 1661–1683 (2010). https://doi.org/10.1177/0278364910368147
2. Robinson, G., Davies, J.B.C.: Continuum robots - a state of the art. Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C). **4**, 2849–2854 (1999). https://doi.org/10.1109/ROBOT.1999.774029
3. Gravagne, I.A., Rahn, C.D., Walker, I.D.: Large deflection dynamics and control for planar continuum robots. IEEE/ASME Transactions on Mechatronics **8**(2), 299–307 (2003). https://doi.org/10.1109/TMECH.2003.812829
4. Bailly, Y., Amirat, Y.: Modeling and control of a hybrid continuum active Catheter for Aortic Aneurysm Treatment. In: Proceedings of the 2005 IEEE International Conference on Robotics and Automation. 924–929 (2005). https://doi.org/10.1109/ROBOT.2005.1570235
5. Best, C.M., Gillespie, M.T., Hyatt, P., Rupert, L., Sherrod, V., Killpack, M.D.: A new soft robot control method: using model predictive control for a pneumatically actuated humanoid. IEEE Robot. Autom. Mag. **23**(3), 75–84 (2016). https://doi.org/10.1109/MRA.2016.2580591
6. Penning, R.S., Jung, J., Ferrier, N.J., Zinn, M.R.: An evaluation of closed-loop control options for continuum manipulators. In: 2012 IEEE International Conference on Robotics and Automation. 5392–5397 (2012). https://doi.org/10.1109/ICRA.2012.6224735
7. Della Santina, C., Duriez, C., Rus, D.: Model-based control of soft robots: a survey of the state of the art and open challenges. IEEE Control Syst. Mag. **43**(3), 30–65 (2023). https://doi.org/10.1109/MCS.2023.3253419
8. Jones, B.A., Walker, I.D.: Practical kinematics for real-time implementation of continuum robots. IEEE Trans. Robot. **22**(6), 1087–1099 (2006). https://doi.org/10.1109/TRO.2006.886268
9. Burgner-Kahrs, J., Rucker, D.C., Choset, H.: Continuum robots for medical applications: a survey. IEEE Trans. Robot. **31**(6), 1261–1280 (2015). https://doi.org/10.1109/TRO.2015.2489500
10. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.A.: Playing Atari with deep reinforcement learning. CoRR (2013). http://arxiv.org/abs/1312.5602
11. Brown, N., Sandholm, T.:Libratus: The superhuman AI for no-limit Poker. Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17, 5226–5228 (2017)
12. Moravčík, M., Schmid, M., Burch, N., Lisý, V., Morrill, D., Bard, N., Davis, T., Waugh, K., Johanson, M., Bowling, M.: DeepStack: expert-level artificial intelligence in heads-up no-limit poker. Science 356(6337), 508–513 (2017) https://doi.org/10.1126/science.aam6960
13. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K.: Ostrovski: Human-level control through deep reinforcement learning. Nature. **518**, 529–533 (2015). https://doi.org/10.1038/nature14236
14. OpenAI: Openai five. OpenAI, OpenAI (2021) https://openai.com/five/
15. Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., Hassabis, D.: Mastering the game of Go with deep neural networks and tree search. Nature **529**(7587), 484–489 (2016). https://doi.org/10.1038/nature16961
16. Levine, S., Finn, C., Darrell, T., Abbeel, P.: End-to-end training of deep Visuomotor Policies. J. Mach. Learn. Technol. **17**(1), 1334–1373 (2015). https://arxiv.org/abs/1504.00702
17. Gandhi, D., Pinto, L., Gupta, A.: Learning to fly by crashing (2017) https://arxiv.org/abs/1704.05588
18. Pinto, L., Andrychowicz, M., Welinder, P., Zaremba, W., Abbeel, P.: Asymmetric actor critic for image-based robot learning (2017). https://arxiv.org/abs/1710.06542
19. OpenAI: Solving rubik's Cube with a robot hand. OpenAI (2022). https://openai.com/blog/solving-rubiks-cube
20. Pan, X., You, Y., Wang, Z., Lu, C.: Virtual to real reinforcement learning for autonomous driving (2017). https://arxiv.org/abs/1704.03952
21. Deng, Y., Bao, F., Kong, Y., Ren, Z., Dai, Q.: Deep direct reinforcement learning for financial signal representation and trading. IEEE Trans. Neural Netw. Learn. Syst. **28**(3), 653–664 (2017). https://doi.org/10.1109/TNNLS.2016.2522401
22. François-Lavet, V.: Contributions to deep reinforcement learning and its applications in smartgrids, ULiège - Université de Liège (2017)
23. Liu, T., Tan, Z., Xu, C., Chen, H., Li, Z.: Study on deep reinforcement learning techniques for building energy consumption forecasting. Energy Build. **208**, 109675 (2020). https://doi.org/10.1016/j.enbuild.2019.109675
24. Liu, J., Shou, J., Fu, Z., Zhou, H., Xie, R., Zhang, J., Fei, J., Zhao, Y.: Efficient reinforcement learning control for continuum robots based on Inexplicit Prior Knowledge. CoRR abs/2002.11573 (2020). https://arxiv.org/abs/2002.11573
25. Thuruthel, T.G., Falotico, E., Renda, F., Laschi, C.: Model-based reinforcement learning for closed-loop dynamic control of soft robotic manipulators. IEEE Trans. Robot. **35**(1), 124–134 (2019). https://doi.org/10.1109/TRO.2018.2878318

26. Wu, Q., Gu, Y., Li, Y., Zhang, B., Chepinskiy, S.A., Wang, J., Zhilenkov, A.A., Krasnov, A.Y., Chernyi, S.: Position control of cable-driven robotic soft arm based on deep reinforcement learning. Information. **11**(6), 310 (2020). https://doi.org/10.3390/info11060310 https://www.mdpi.com/2078-2489/11/6/310
27. You, X., Zhang, Y., Chen, X., Liu, X., Wang, Z., Jiang, H., Chen, X.: Model-free control for soft manipulators based on reinforcement learning. In: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2909–2915 (2017). https://doi.org/10.1109/IROS.2017.8206123
28. Lillicrap, Timothy P. and Hunt, Jonathan J. and Pritzel, Alexander and Heess, Nicolas and Erez, Tom and Tassa, Yuval and Silver, David and Wierstra, Daan: Continuous control with deep reinforcement learning (2015) https://arxiv.org/abs/1509.02971
29. Phaniteja, S., Dewangan, P., Guhan, P., Sarkar, A., Krishna, K.: A deep reinforcement learning approach for dynamically stable inverse kinematics of humanoid robots (2018). https://arxiv.org/abs/1801.10425
30. Shi, X., Guo, Z., Huang, J., Shen, Y., Xia, L.: A distributed reward algorithm for inverse kinematics of arm robot. 2020 5th International Conference on Automation, Control and Robotics Engineering (CACRE), 92–96 (2020)
31. de la Bourdonnaye, F., Teulière, C., Chateau, T., Triesch, J.: Within reach? learning to touch objects without prior models, 2019 Joint IEEE 9th International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob), 93–98 (2019)
32. Ghouri, U. H., Zafar, M.U., Bari, S., Khan, H., Khan, M.U.: Attitude control of quad-copter using deterministic policy gradient algorithms (DPGA). 2019 2nd International Conference on Communication, Computing and Digital systems (C-CODE) 149–153 (2019)
33. Rao, P., Peyron, Q., Lilge, S., Burgner-Kahrs, J.: How to Model Tendon-Driven Continuum Robots and Benchmark Modelling Performance. Frontiers in Robotics and AI **7**(630245), 20 (2021). https://doi.org/10.3389/frobt.2020.630245
34. Hannan, M.W., Walker, I.D.: Kinematics and the implementation of an Elephant's Trunk manipulator and other continuum style robots. J. Robot. Syst. **20**(2), 45–63 (2003). https://doi.org/10.1002/rob.10070
35. Wang, S., Chaovalitwongse, W., Babuska, R.: Machine learning algorithms in Bipedal Robot control. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) **42**(5), 728–743 (2012). https://doi.org/10.1109/TSMCC.2012.2186565
36. Plappert, M.: keras-rl: Deep reinforcement learning for Keras. GitHub (2019). https://github.com/keras-rl/keras-rl
37. Sewak, M.: Temporal difference learning, SARSA, and Q-Learning. Deep Reinforcement Learning: Frontiers of Artificial Intelligence. 51-63 (2019)
38. Haarnoja, T., Zhou, A., Abbeel, P., Levine, S.: Soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor. CoRR abs/1801.01290 (2018). http://arxiv.org/abs/1801.01290
39. Fujimoto, S., van Hoof, H., Meger, D.: Addressing function approximation error in actor-critic methods. CoRR abs/1802.09477 (2018). http://arxiv.org/abs/1802.09477

Turhan Can Kargin received the M.Sc. degree from Poznan University of Technology, Poland, in 2022 (Faculty of Automatic Control, Robotics and Electrical Engineering). In 2020 he graduated Izmir Katip Celebi University (Bachelors of Science, Electrical & Electronics Engineering). His area of interests are AI, Robotics, Electronics, Deep Learning, Control Theory, Machine Learning and Computer Vision.

Jakub Kołota received the M.Sc. degree from Poznan University of Technology, Poland, in 2005, and the Ph.D. degree from the same university in 2009. He is presently a research scientist at the Faculty of Control, Robotics and Electrical Engineering, Poznan University of Technology in Poland. His main research area is control theory in smart materials. Currently he is working with electroactive polymers like IPMC and DEAP.