TECHNICAL UNIVERSITY OF KOŠICE
FACULTY OF MECHANICAL ENGINEERING

Redundant robots control by machine learning

Dissertation Thesis

2024 **Peter Jan Sincak, Ing.**

Redundant robots control by machine learning

Dissertation thesis

Ing. Peter Jan Sincak

Supervisor: prof. Ivan Virgala

Department of Industrial Automation and Mechatronics
Faculty of Mechanical Engineering
Technical University of Košice

This dissertation is submitted for the degree of
Doctor of Philosophy

April 2024

Student Card:

Name : Ing. Peter Jan Sincak
Year : 4
Supervisor: prof. Ivan Virgala
Guarantor: prof. Michal Kelemen
Department: Department of Industrial Automation and Mechatronics
Institute: Institute of Automation, Mechatronics, Robotics and Production Systems
Study program: Industrial Mechatronics
Field of study: Mechanical Engineering
Thesis title: Redundant robots control by machine learning

Funding

This research was funded by VEGA 1/0436/22 Research on modeling methods and control algorithms of kinematically redundant mechanisms, KEGA 030TUKE-4/2020 Transfer of knowledge from the area of industrial automation and robotics to the education process of Mechatronics study, 04/TUKE/2023 Control of kinematically redundant robots for minimally invasive surgeries.

Declaration

I hereby declare that, the contents of this dissertation are original and are my own work, except where specific reference is made to the work of others. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements.

Ing. Peter Jan Sincak
April 2024

Acknowledgements

I would like to thank my parents, family, and friends whose encouragement and support helped me during my studies. Many thanks also goes to my colleagues at ARM lab and at Department of Industrial Automation and Mechatronics, especially to my supervisor prof.Virgala, whose patience and wisdom guided me where I am today, also special thanks goes to head of our department prof.Kelemen, for providing valuable advices and place for work.

Abstract

The aim of this thesis is to advance the understanding and utilization of cable-driven continuum robots through the application of machine learning methods for control. A versatile testing bench was developed specifically for cable-driven continuum robots, emphasizing modularity and ease of use. This bench facilitates validation and testing of kinematic structures of cable-driven continuum robots and their control algorithms, as demonstrated in experimental validations. The test bench was instrumental in evaluating different kinematic structures and control algorithms. Notably, two machine learning-based control algorithms were introduced. The first algorithm employed a reinforcement learning framework with a deep Q-learning agent to optimize cable lengths based on proximity to a target position. The second approach focused on supervised learning, employing regression modeling to predict cable displacements necessary to reach a specified target point. A comprehensive dataset comprising 30,000 action combinations within the robot's operational space was generated using a standardized initial position and motion capture system, forming the basis for training the models.

Keywords:
Continuum robots, Supervised learning based controller, Continuum robot dataset, Redundant robots

Abstrakt

Cieľom tejto práce je posunúť pochopenie a využitie káblových kontinuum robotov prostredníctvom aplikácie metód strojového učenia na riadenie. Bolo vyvinuté univerzálne testovacie zariadenie špeciálne pre káblové kontinuum roboty, kde sa kladie dôraz na modularitu a jednoduché použitie. Toto zariadenie umožňuje validáciu a testovanie kinematických štruktúr káblových kontinuum robots a ich riadiacich algoritmov, čo bolo preukázané v experimentoch. Testovacie zariadenie bolo dôležitý pri hodnotení rôznych kinematických štruktúr a riadiacich algoritmov. Zvlášť boli predstavené dva riadiace algoritmy založené na strojovom učení. Prvý algoritmus využíval metódu učenia s odmeňovaním s hlbokým Q-učiacim agentom na optimalizáciu dĺžok káblov na základe vzdialeností k cieľovej pozícii. Druhý prístup sa sústredil na kontrolované učenie a použil regresné modelovanie na predikciu posunov káblov potrebných na dosiahnutie určenej cieľovej pozície. Rozsiahly dataset zahŕňajúci 30 000 kombinácií akcií v operačnom priestore robota bol vytvorený pomocou štandardizovanej počiatočnej pozície a systému na zachytávanie pohybu, čím sa vytvoril podklad pre trénovanie modelov.

Kľúčové slová:
Kontinuum roboty, Riadenie na báze kontrolovaného učenia, Súbor údajov kontinuum robota, Redundantné roboty

Table of contents

List of figures

List of tables

List of Algorithms

Nomenclature

Acronyms / Abbreviations

CC	Constant curvature
CR	Continuum robot
DNN	Deep neural networks
DQN	Deep Q networks
MDP	Markov decision processes
ML	Machine learning
mocap	Motion capture system
PCC	Piecewise constant-curvature approximation
RL	Reinforcement learning
SARSA	State-action-reward-state-action
TRPO	Trust Region Policy Optimization

Chapter 1

Introduction

1.1 Motivation

The continuum robotics is a rapidly progressing field that is developing across many different application fields such as minimally invasive surgery, jet engine inspections, general inspection of tight spaces, and various manipulation and collaborative tasks. The task of controlling the continuum robot in these challenging applications is, therefore, crucial.

Currently, there is an increasing trend of research of soft pneumatically controlled continuum robots and their control. This trend is also pushing the research of other types of continuum robots that are more rigid, such as concentric tube continuum robots or continuum robots that are controlled by cables. These robots are the focus of our research. At our laboratory the research of continuum robots has naturally evolved from the research of in-pipe robots and snake robots towards kinematically redundant and continuum robots. The focus of our research has been on the development of new kinematic structures, actuation principles, and control strategies of these robots. This focus is driven by the wide variety of the applications that these robots can be used for, however the most interesting application to us is the application in medicine. The challenges in the minimally invasive surgery like tight spaces for the movement of the robots, small scale of the robots and lack of sensing are some of our key motivations for the research of control algorithms for cable driven continuum robots.

1.2 PhD thesis overview

This work is divided into seven chapters. Chapter 1 is dealing with the introduction of continuum robots and their classification based on the actuation principle. The sensors

that are used in continuum robots and their importance for the control of the robots are also classified in this chapter. Chapter 2 is dealing with the preview of machine learning, its classification and the introduction of selected methods as well as, the mathematical modelling of continuum robots. In chapter 3 the continuum robot hardware is introduced. The testing stand, the actuation principles and different structures of continuum robots that were developed by our research group are introduced here. In chapter 4, the algorithms that were used and implemented for the control of continuum robots are described, together with their working principles, features and details. In chapter 5 the experimental setup is shown and experiments that were conducted are presented. Chapter 6 deals with the results of these experiments and their performance and comparison. Lastly, chapter 7 summarises the work and the achieved results.
The main points of this work are as follows:

- Classification of continuum robots and sensors used in this domain
- Experimental setup that allows for testing of various kinematic structures of tendon driven continuum robots and different control algorithms
- Implementation of reinforcement learning base control algorithm for tendon driven continuum robots
- Implementation of supervised learning base control algorithm for tendon driven continuum robots
- Experimental verification of kinematic structures of continuum robot
- Experimental verification of proposed control algorithms
- Summary of the experimental verification

1.3 Continuum robots

The continuum robots are biologically inspired group of robots, whose main characteristic is large number of joints that correlate with number of degrees of freedom. Unlike conventional robotic manipulators these joints can be either passive or active depending on whether they are actuated or not. Due to these factors and because they have more degrees of freedom that they need for accomplishing their tasks, the continuum robots can be regarded as redundant or hyper-redundant robots. Their capabilities are based on snakes, elephant trunks and all kinds of tentacles that can be found in nature [64, 73]. The large number of joints allows

them to move in various complicated environments and adapt to obstacles in the way just like snakes or tentacles. Together with their small size and great adaptability and dexterity in the

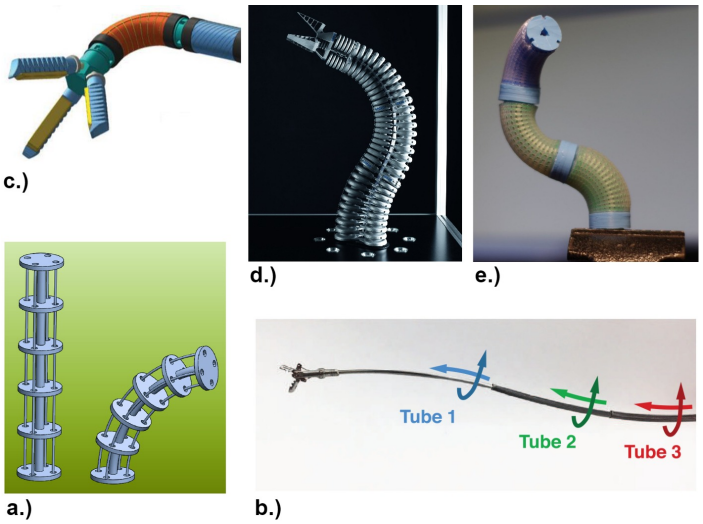


Fig. 1.1 a.)Tendon driven CR[3], b.)tubular CR [6], c.)hydraulic driven CR [103], d.)Festo bionic hand [16], e.)soft CR [23]))

environment, these robots are often chosen for very restricted and fragile environments where conventional robots or robotic manipulators could not be used. One of the application fields where continuum robots are very useful is the inspection of various tight spaces [84, 90] e.g. jet engines inspections[17, 18]. Another area where continuum robots are used are different manipulation tasks [43, 21], to move objects in different environments. The most popular area for the application of continuum robots is the medical field [7, 19, 100]. Mainly in minimally invasive surgeries the importance of small diameter robots is important, these robots are used here instead of laparoscopic tools, since they can have one or more functions [40]. These robots can carry a camera to perform inspection and/or have other tools such as clamps to carry objects in and out or scissors for cutting [10, 83]. The main advantage of using these robots lies in a lower number of incisions necessary for surgery and therefore a decrease in the healing time of the patient. Some of the various types of continuum robots can be seen in Fig. 1.1. The tendon driven continuum robot controlled by four tendons can be seen in Fig. 1.1a.), the soft robots can be seen in Fig .1.1c.),d.),e.) and tubular continuum robots can be seen in Fig. 1.1b.). There are three main categories of continuum robots based on their actuation and mechanical design as can be seen in Fig. 1.2. All categories are based on the most common mechanical design and actuation principle. The main three groups of continuum robots are tubular, soft and cable-driven continuum robots.

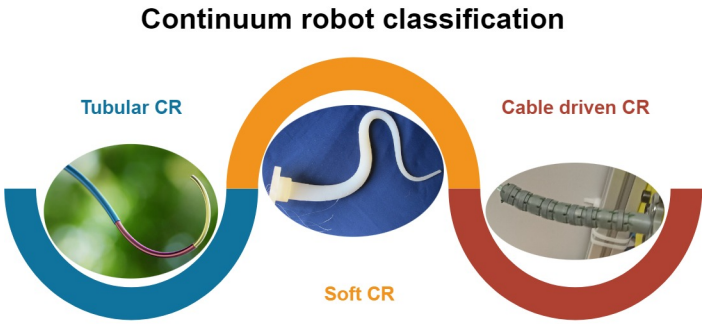


Fig. 1.2 Continuum robot classification [75]

1.3.1 Soft CR

One of the most popular continuum robots are soft robots, which are a subset of continuum robots. The advantage of such robots is their softness and therefore natural predisposition to work in collaboration with humans or fragile objects [20]. Soft continuum robots are often made of silicone or other similar material, where as a manufacturing process, casting has to be used. This allows various shapes, sizes, and cavities to be created to accommodate the actuation. Many soft robots are often actuated with pressurised air or other fluids that flow through the channels or cavities inside of the robot body, as can be seen in [34, 96, 2], and therefore move the robot in the direction and shape that is desired. Some of the soft robots like in [2], combine the fluid based actuation with tendons to further improve the movement parameters of the robot like load capacity, speed, precision or dexterity of the robot.

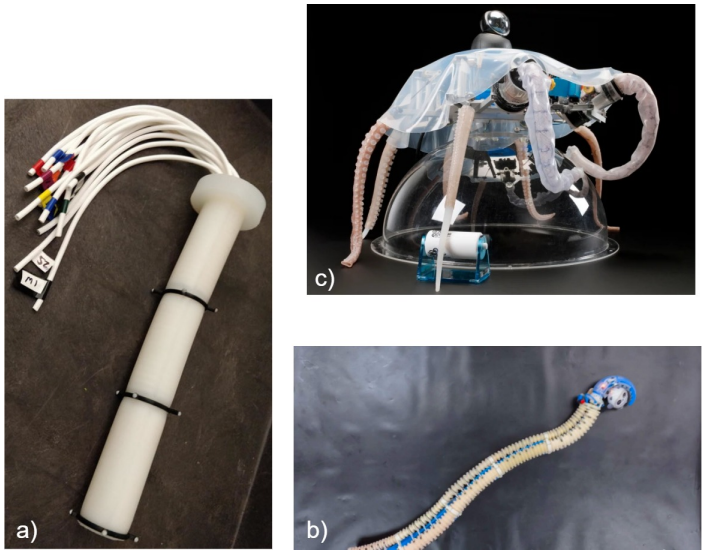


Fig. 1.3 A few of the soft robots: a)Pneumatically controlled soft arm prototype [41], b) Octopus inspired soft arm [95], c) Octopus robot with soft tentacles [74]

1.3.2 Tubular CR

The other category of continuum robots, is based on concentric tubes [72, 22, 91, 92, 81]. This type of robot is often used as a needle or cannula in medical applications. The main principle on which these robots are based on, are the precurved concentric tubes, which are then rotated and pushed in and out in a way, so that the robot reaches the desired position. This is visible in [88], where the robot is using super-elastic precurved tubes as active cannulas. These cannulas or needles can be then used for smart drug delivery or inspection

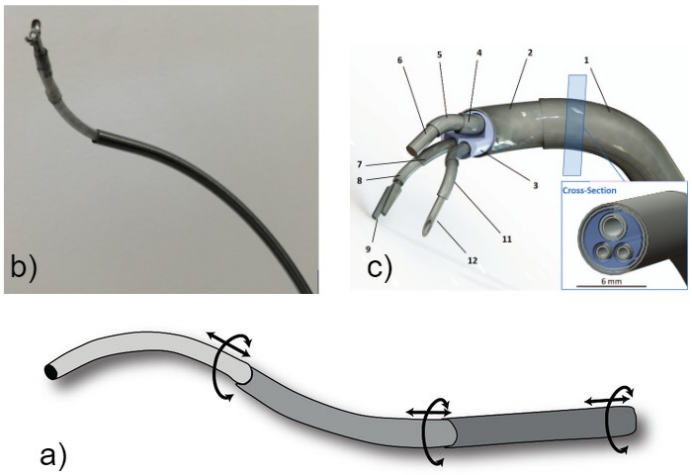


Fig. 1.4 A few of the concentric tube continuum robots: a)Simple concentric tube robot[7], b) Concentrit tube robot with a gripper [60], c) Concentric tube robot with multiple arms [53]

of veins and other parts of human body, where very small sizes are needed. The tubular continuum robots have the advantage of having a small diameter that can be useful in other applications as well. These applications might require precise manipulation, which is a difficult task.

1.3.3 Cable-driven CR

Another category of continuum robots are cable-driven continuum robots. These robots consists of multiple smaller units that are interconnected and form a single unit, which is controlled by the number of tendons or cables. This can be seen in [101, 57, 62, 29]. These single units can be multiplied and attached to each other to form multi-segment continuum robots. By using multiple segments, the dexterity, adaptability and reach of the robot is multiplied and extended. These kinds of continuum robots are highly dexterous, however, at the same time most of the cable-driven continuum robots have the load capacity quite limited due to the lower stiffness which is a trade-off with the high flexibility of the segments.

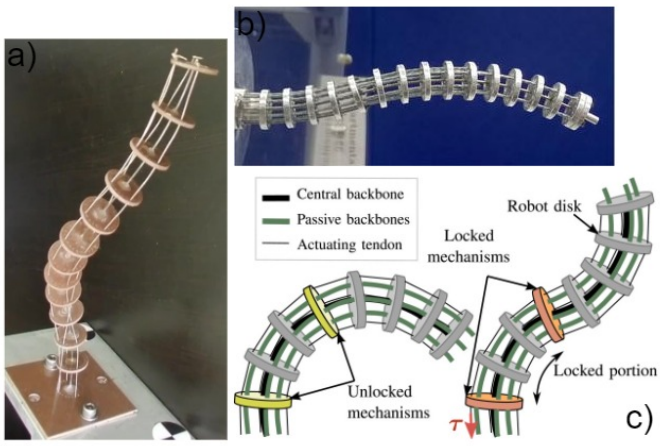


Fig. 1.5 Selection of the cable-driven continuum robots: ta)Three cable-driven continuum robot [80] b) Cable-driven continuum robot for minimally invasive surgery [58] c)Cable-driven continuum robot prototype [62]

1.4 Continuum robot sensors

Different types of continuum robots are creating different challenges when it comes to control, sensing and modeling of these robots. The continuum robot sensing is a challenging area of research that along the research of novel kinematic structures is being rapidly developed. The importance of sensing is highlighted when precise manipulation is needed. Based on what is being sensed, two main groups of sensors are known as shape sensing sensors and environment sensors. The shape sensors are focusing on sensing of the robot's shape. By tracking various physical properties that change with the shape of the robot, the correlations are being observed and used for the shape tracking. On the other hand the need of tracking the environment around the robot and it's interactions with it is also important.

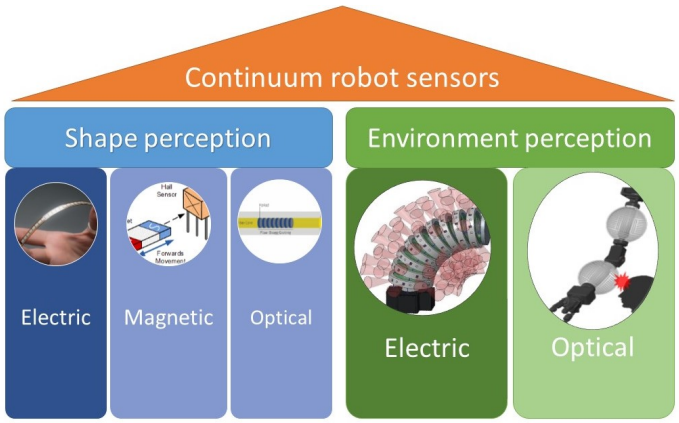


Fig. 1.6 Continuum robot's sensors classification [75]

1.4.1 Shape perceptions

When it comes to the control of continuum robots it is possible to have control without feedback, however, sensing feedback might help to achieve the desired goal faster and with higher precision. The information about the shape of the robot can be obtained by integrating the sensors or part of the measuring chain in the robot. The most commonly used shape sensing sensors are tracking physical properties that correlate with the shape of the robot. These properties are based on three reacuring physical principles that the sensors most commonly use, for sensing the shape of the robot. Based on this, the shape sensing sensors could be categorised into three main groups: electric, magnetic and optical based sensors [75]. The groups are based on the physical property that the sensors sense in order to get information about the change in the robot's shape.

Electric base shape sensors

One of the most popular types of sensors used in this domain are the sensors based on measuring electrical properties [75]. These sensors are based on the change of resistance or capacitance that corresponds with the change of the shape. Most of the sensors are custom made for an application of a concrete continuum robot. However, an application of commercially available bend sensor is shown in [26]. In this paper, a commercially available

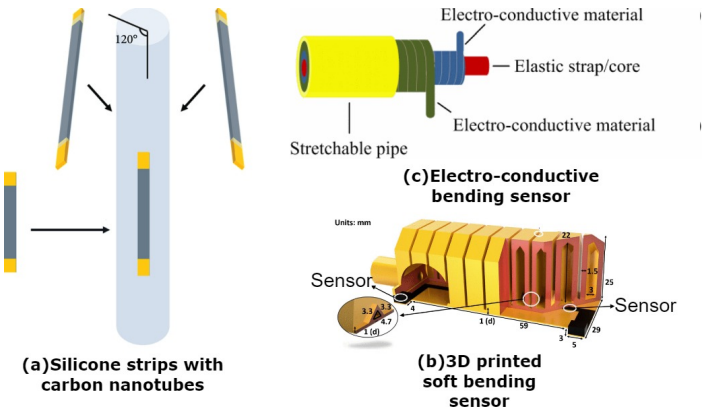


Fig. 1.7 A few of the electric based shape sensors: a)Custom made silicone stripes with carbon nanotubes [99], b) Embedded 3D printed electric based soft sensor [32], c)Stretcheble electric based shape sensor [93, 94]

flexible potentiometers are integrated in a continuum robot structure. The sensor allows for single axis bending and outputs an analog signal, just like regular potentiometer. There are two sensors opposite each other, embedded in the centre part of a pneumatically actuated continuum robot. The advantage of using this sensor is the high resolution of the analog

signal, however, the disadvantage is the one axis bending, since the sensor doesn't allow for bending in more then one axis, which would damage the sensor. In [9], authors showcased an array of sensors placed in an outer diameter of continuum robot structure. The sensors are in the form of flexible and conductive stripes, made of a mixture of polyurethane and carbon black. Carbon particles ensure the conductivity of the flexible stripes and therefore the resistance of the stripes during the movement is changing, which corresponds with the change in the shape of the robot. Similarly in [32], the author used a commercially available conductive filament for 3D printing. This was used during the manufacturing of a flexible pneumatic actuator, which was 3D printed. During this process, the sensor or the sensing part of the robot was printed as well and integrated in the structure of the robot. The sensor is in the shape of a straight line along the backbone of the soft actuator with two contact points for electrical connections. Another case of flexible sensor was introduced in [68]. The sensor in this case is a conventional spring that is connected to a electrical circuit, where it acts as an inductor and the peak-to-peak voltage is measured. This changes based on the elongation of the spring. The calibration curve of the sensor was determined before the integration in tendon driven continuum robot and, therefore, is independent from the continuum robot structure. The sensor embedded in the robot's structure was shown in [59]. In this case, the sensor was showcased on a general pneumatic actuator. In the structure of the pneumatic actuator, cavities were formed and filled with conductive liquid. This liquid then acted as a variable resistor, measuring the curvature indirectly by measuring the strain.

Magnetic based shape sensors

Another group of sensors that are tracking the shape are using the magnetism to do so. This can be seen in [13], where a soft robot with embeded magnets and magnetic sensors is discussed. The soft robot is composed of three segments and each has one magnet. The magnetic sensors were placed arbitrarily in the robot structure. Their information is then used to predict the shape of the robot using neural networks. In [31], the author proposed a similar setup integrating permanent magnets in the kinematic structure of the continuum robot and magnetic flux density sensors. By using these sensors, it was possible to sense the bending angle while assuming a constant bending of single section continuum robot. A different approach of placing coils on a catheter was shown in [36]. The coils are made of copper wire, wrapped around the plastic catheter. One set of larger coils are used for generating magnetic field and second set of smaller coils are used for sensing. The sensing is based on the mutual inductance of the sensing coils and coils generating magnetic field. This is dependent on the bending of the catheter and this relationships is used for shape reconstruction.

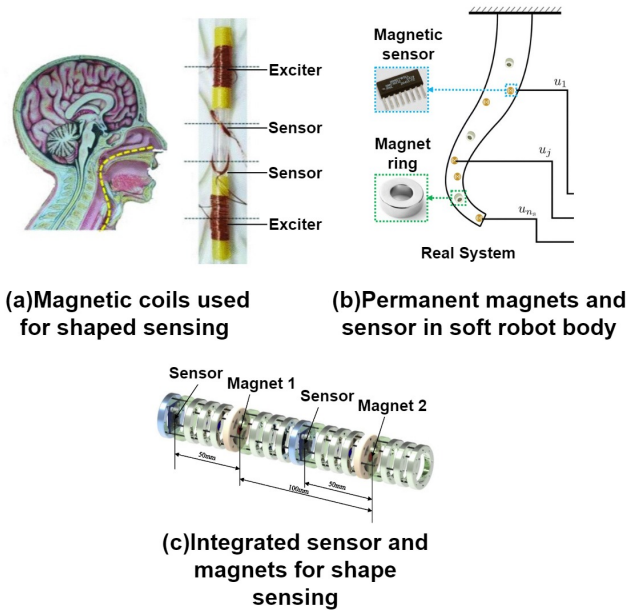


Fig. 1.8 A few of the magnetic based shape sensors: a)Magnetic coils used for shape sensing [36], b) Embedded permanent magnets and magnetic sensors used for shape sensing [4], c) Magnets and magnetic sensor within the kinematic structure [31]

Optical base shape sensors

The group that is using the properties of travelling light is also very dominant in the domain of sensing continuum robots. These sensors often use optical fibers, lasers, cameras and LEDs to measure the bending or shape of continuum robots. One of those methods was used in [47].

The authors in this paper showed a cable driven continuum robot moving in 2D. The bending was sensed by using optical fibers. The optical fiber was placed between two nitinol cables so that it follows the same path as the cables used for actuation. The authors opted to use an approach called fiber bragging grating, or FBG. This technique is using optical fibers that have groves in predefined locations and these grooves ensure that the travelling light is changing its wavelength depending on the applied strain, which can be recalculated to the bending angle. This optical fiber can then be used as a sensor for detecting the bending angle of such a continuum robot. A different approach was used in [45], where a cable-driven continuum was used to test a shape sensing method based on passive cables and camera. In this case, the robot was actuated with the cable, but in addition it had passive cables that were connected to each of the sections of the robot and the other end was free, so that the cables would move as much as the robot would bend. Just before the actuation unit a section was made where these passive cables were routed and markers were placed on each of the cables.

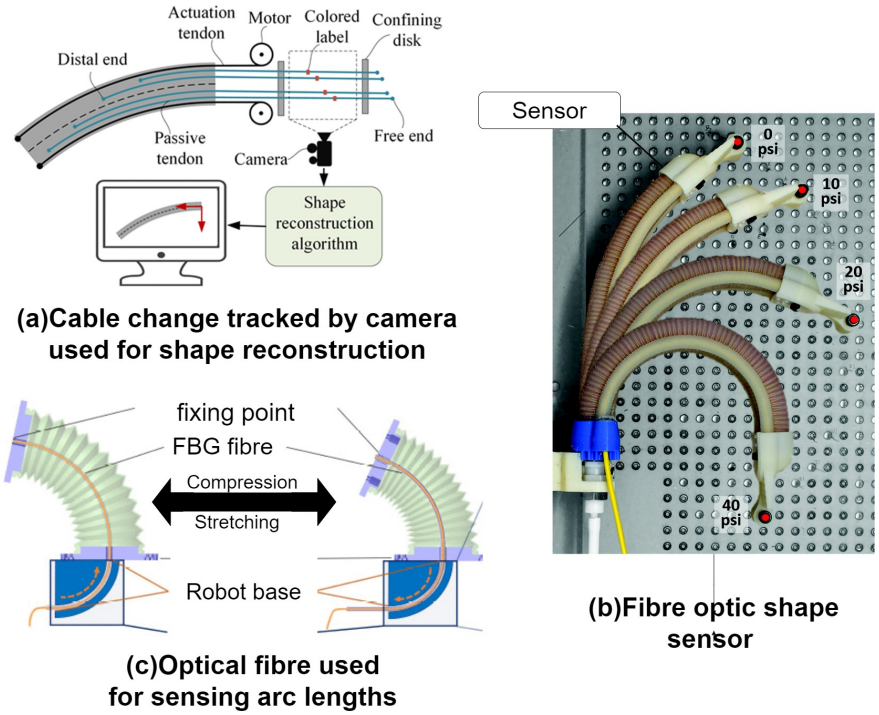


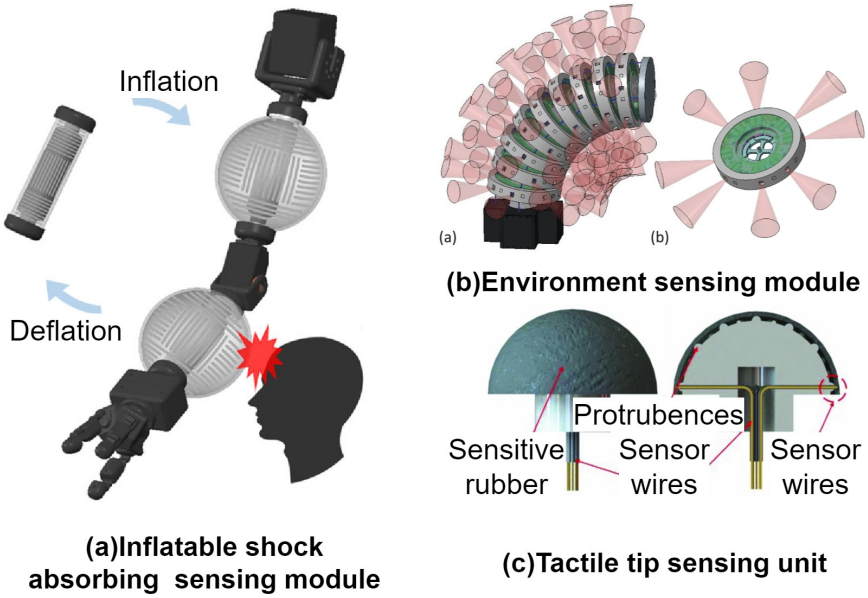
Fig. 1.9 A few of the optical based shape sensors: a)Magnetic coils used for shape sensing [45], b) Optical fibre wrapped around the outer shape of the robot [24], c) Optical fibre used for arc measurement [50]

These were then observed with the camera and shape reconstruction algorithm was used to determine the shape of the robot. In [71], the method that is using the travelling light through cavities in a dedicated sensing unit connected to a soft robot bending unit, was shown. The bending unit has three cables that have a reflective surface at the connection with the sensing unit. As the robot bends the reflective surface moves and the travelling light in the sensing unit has to travel a different distance and the shape of the robot can be recalculated based on these data from the thee channels that are in the sensing unit.

1.4.2 Environment perception

The use of shape sensing sensors is advantageous and important when control algorithms are used. However, when it comes to motion planning, it is important to have the knowledge, about the environment and possible obstacles that might be in the way. In this case, the environment sensors are used to detect possible obstacles or interactions of the robot with the obstacles. This can be especially useful in applications where the avoidance of obstacles is necessary or the interactions have to be careful and limited. Using a similar principle for categorisation as in section 1.4.1, the sensors are divided into two main groups using the

properties of electricity and the properties of travelling light to sense obstacles and detect collisions with them.



(a)Inflatable shock absorbing sensing module

(b)Environment sensing module

(c)Tactile tip sensing unit

Fig. 1.10 A few of the Environment sensors: a)Contact sensing modules [42], b) Optical based environment sensors [1], c)Tactile tip unit [98, 97]

The group of sensors that use the properties of electricity is often used for detecting a contact with the sensor that is integrated on an outer layer of the robot structure. The integration of the sensor into an other layer was shown in [42]. The authors proposed inflatable modules made of elastic silicone. During the molding process of the elastic silicone channels were made and filled with conductive liquid. The channels made in the elastic material are made in specific pattern to act as strain gauges. The pattern divides the module into three sections so that the contact location can be detected as well. When the module is inflated, it acts as a bumper and at the same time, when it collides with an object it acts as a sensor. The liquid changes its resistance, when the contact is detected. The soft material is also acting as a shock absorbing element and prevents the damage to the robot and the obstacles. A similar approach was also shown in [86], where a soft tactile sensor was introduced and applied on a continuum robot. The sensor is in the shape of a dome which consists of a piezoresistive array. During the contact of the dome with an object the the local resistivity changes and the contact location is detected using electrical impedance tomography using eight electrodes. The electrical impedance tomography is an imaging technique using the change in conductivity to display the object and in this case the location of the contact.

By using the principles of travelling light to detect objects, collisions and map the environment, it is possible to overcome the issues of noise in the signal that is often introduced by the presence of other electrical devices or magnetic fields. The noise can not influence the travelling light, only the signal from the sensor to the acquisition system, which is less prone to the introduction of the noise. A tactile sensing system was shown in [5], using optical fibres to emit and detect the travelling light. An array of nine small domes made of soft material was created to act as sensing element. Each dome is hollow and has a connecting channel to a pair of optical fibers, where one acts as source of the light and the other one detecting the reflected light. Based on the applied force, the dome is deformed and the light is travelling smaller distance and the intensity of the light is measured. In [1], a different approach was shown using multiple time of flight sensors to track the environment around the continuum robot. The continuum robot is made of multiple discs and each disc contains eight sensors scanning the environment around the robot. In addition to the time of light sensors, permanent magnets and hall sensors were placed on a deformable ring around the disc, leaving an air gap between the hall effect sensor and magnets. These are used for contact and force detection.

1.5 Chapter summary

In this chapter, a theoretical background of continuum robots is given. The classification of continuum robots can be based on various criteria, however the actuation principle and mechanical design are the two most common features that connect the various continuum robots. Based on these, the continuum robots can be grouped in tubular, soft, and cable-driven continuum robots. This does not mean that there are no combinations of the features that represent each group, but these are the most dominant groups. The tubular continuum robots are made of precurved concentric tubes with small diameters, which makes them well suited for medicine applications. The group of soft robots are robot that are made of soft materials which allows them to work in cooperation with human, or to manipulate fragile or hard to grasp objects. The cable-driven continuum robots can be characterised by the cables that are most of the time embedded within the kinematic structure of the robot. By shortening or elongating one or more cables, the change in the shape and position of the continuum robot can be achieved. Each of the groups has its challenges that come with their specific design and nature of actuation or materials that are used for their manufacturing. The tubular continuum robots have an advantage of small size, therefore they are often used in medicine. However, this can also be a problem when shape sensing or environment sensing is required. When it comes to the soft robots, their biggest advantage is their soft material, which comes

handy when hard to grasp object are being picked or in cases when the environment is fragile. On the other hand, the lifespan of these robots is often limited compared to tubular robots. The cable-driven continuum robots are also useful in cases when adaptability in a complicated environment is needed, but their load capacity can be limited depending on the number of cables used for actuation or the material used for their manufacturing. In this section, the sensors that are used in the domain of continuum robots were also described and classified based on the principle of sensing and what is being sensed. The main division of sensors is based on what is being sensed, either the shape or the environment surrounding the robot during the movement. Based on physical principles, shape sensors can be grouped in to electric, magnetic, and optical-based sensors. The environment sensor seems to be less explore and can be grouped to electric and optical-based sensors.

Chapter 2

Machine learning and continuum robots modeling approaches

The machine learning(ML) field is a progressive field of research that is being influenced by all engineering and non-engineering fields. Depending on the availability of the data, there are multiple learning scenarios or approaches that are used in the domain of machine learning [52]. The three main approaches used in machine learning are supervised learning, unsupervised learning and reinforcement learning(RL). The supervise learning approach

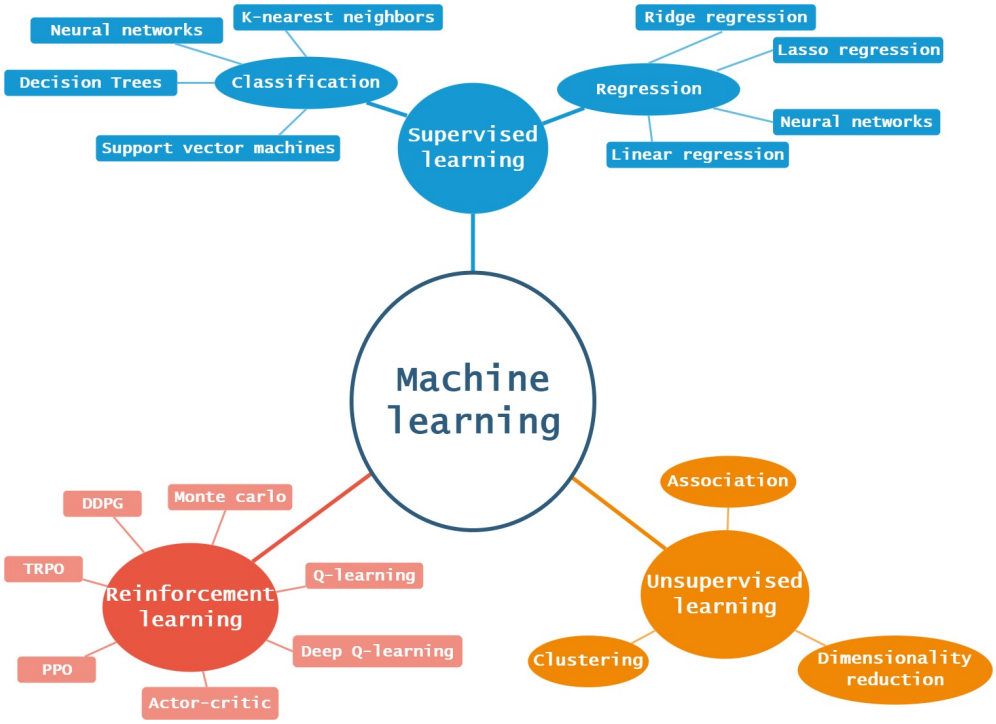


Fig. 2.1 Machine learning methods

is using labelled data, meaning that the input are paired with the outputs and this is used to learn a function that approximates this relation through all the given data [67]. On the other hand, unsupervised learning is using unlabelled data, and the algorithm is training to find patterns, similarities or other information from these data without any guidance or instructions [67, 28]. The third approach, called reinforcement learning, is based on interacting with an environment. Based on the interaction, it is rewarding or punishing an agent which is responsible for these interactions.

2.1 Supervised and unsupervised learning

The supervised learning is the most widely used approach today in machine learning. This approach is based on the use of labelled data, which means that the inputs are mapped to the appropriate outputs during the learning process[11, 49]. The best interpretation of this approach would be to imagine a supervisor telling the algorithm that at certain input x the correct output would be y and based on certain number of training inputs and outputs supervised algorithm has to learn to associate the inputs with the outputs so that it can predict the output of the data that were not used during training [14]. Often the training data set can be collected automatically, e.g. trying random inputs and measuring the outputs, however, in some cases the supervisor is necessary to add the labels to the data, e.g. labelling objects in images for image recognition. Generally, the problems that are being solved by supervised learning can be divided into either classification or regression problems. This can be seen in Fig. 2.1. The classification algorithms are trained to assign predefined labels, categories, or outputs to the inputs. After training is complete, the goal is to predict the correct labels or classes for the new unseen data. The key attribute of classification algorithms is that the number of labels, classes or outputs is a finite number, e.g. yer or no problem. When the number is not finite, e.g. temperature prediction, it is called regression. In regression tasks, the algorithm is tasked to predict a continuous numerical value based on the given input.

Similarly, in unsupervised learning, the algorithms require data. However, in this case, the labels are not available and the supervisor does not know them. Based on the inputs given, the algorithm is trying to figure out the patterns or find the correlations in the data that are not visible at first glance [11]. The most common algorithms are based on so-called clustering, dimensionality reduction and association [38]. The clustering is based on receiving raw data and is trying to group the data into clusters, based on common traits that these data might have [44]. The association approaches use rule-based learning to find relationships between points in the dataset. This approach is often used for the analysis of shopping habits or prediction of illnesses based on the symptoms presented [12]. Lastly, dimensionality

reduction problems focus on reducing the number of features or dimensions of the dataset to easier visualise the data or to preprocess the data for future use. The algorithms focus on extracting the important parts of the dataset would not compromise the overall results.

2.2 Reinforcement learning

To learn how to map the states to the actions, while maximising the reward would well interpret reinforcement learning[77]. The biggest difference between supervised / unsupervised learning and reinforcement learning lies in the fact that the first two methods require a data set, which is either labelled or unlabelled, whereas the RL uses an agent to interact with an environment and learn directly from these interactions [63]. In this chapter, the Markov decision processes(MDP) as the basic framework used in reinforcement learning will be explored and characterized, as well as application of reinforcement learning and a few reinforcement learning algorithms.

2.2.1 The Markov Decision Processes in regards to reinforcement learning

The Markov decision process is a mathematical framework that is used to describe the interactions within the reinforcement learning to achieve a set goal. The so-called agent in RL is the one that is learning and making the decisions by interacting with so called environment [85]. The interaction between the agent and the environment is carried out in discrete time steps t, where at each time step the environment is in a certain state s and the agent chooses an action a based on the learned policy π. As a consequence to the taken action the environment updates its state to the new state s' and a numerical value called reward r is calculated based on whether the action taken has had a positive impact on the new state or not. The agent is learning by adjusting its policy so that the optimal action selection policy π^*, maximizes the total sum of rewards received during the learning process.

In most cases, at the time step t the environment state transition is dependent on all previous actions that were taken during the interaction of an agent and the environment. At this time step t, the probability can be defined[77]

$$Pr\{r_{t+1} = r, s_{t+1} = s' | s_0, a_0, r_1, a_1, s_1, ..., a_{t-1}, s_{t-1}, r_t, s_t, a_t\} \tag{2.1}$$

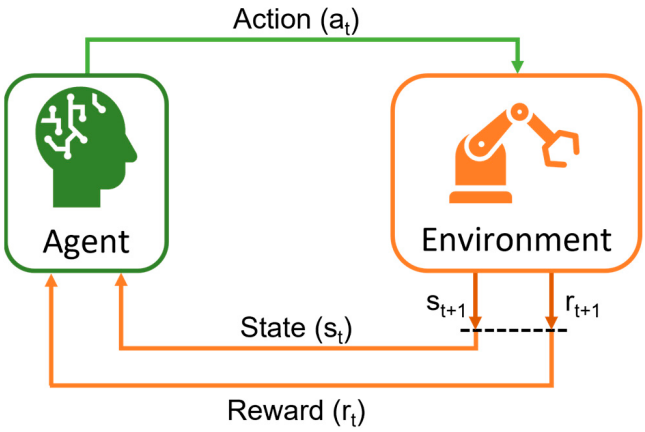


Fig. 2.2 Interaction between the agent and environment[15]

If the MDP is finite and have finite sets of elements such as state, action, reward, the state transition is only dependent on one previous step before the action was selected:

$$p(s',r|s,a) = Pr\{r_{t+1} = r, s_{t+1} = s'|s_t, a_t\} \tag{2.2}$$

If this is true then state signal has so called *Markov property* and this state is called a *Markov state*. This means that for all s' and r, equations 2.1 and 2.2 must be equal and the function 2.2 defines the dynamic of the MDP. The one-step dynamics of the environment not only allows for predicting the next state based on current state and taken action, but also all the future states and rewards by using the equation 2.2. The optimal policy π^*, is therefore approximated by selecting the action with the highest expected cumulative reward from state s.

If we consider the finite MDP we are accounting for the finite number of all possible states s of the environment as well as the finite number of all possible actions a that an agent can take. The MDP is defined by a tuple $< S,A,R,T >$, where S is the state space and contains all possible states of the environment $s \in S$, A is the action space and contains all possible actions that the agent can take $a \in A$, $R : S \times A \rightarrow r$ is the reward function, which is a numerical value or signal that changes according to the action taken by the agent and its influence on the state of the environment[77]. In general, the agent is trying to maximise the cumulative reward and not the immediate one after a single pass[77]. The transition function $T : S \times A \rightarrow t$ calculates the probabilities of transitioning to every new state s' based on the taken action by the agent and current state s. Based on the current state s and the action

selected by the agent, the expected reward can be computed as[77] :

$$r(s,a) = \mathbb{E}[R_{t+1}|S_t = s, A_t = a] = \sum_{r\in R} r \sum_{s'\in S} p(s', r|s, a). \tag{2.3}$$

The equation 2.3 only defines reward after one interaction of agent and environment. As mentioned before, the agent's goal is to maximise the total sum of rewards during the interaction, this can be calculated from *value function*, which is information for the agent how good it is to choose certain action being in the current state s or why it is good to be in this state in context of the expected reward [78]. The value function v of state s is calculated as expected return for starting in state s and following the policy π [77].

$$v_\pi(s) = \mathbb{E}_\pi[G_t|s_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}|s_t = s\right], \tag{2.4}$$

The $\mathbb{E}_\pi$ denotes the expected value of return G_t. This value is the expected return from state s and following the policy π. The discount factor $\gamma \in [0,1]$, is parameter used to balance the influence of the rewards. If this parameter is set to 0 the value function considers only the immediate rewards, while it is set to 1 all the rewards have the same importance. The value of taking an action a, being in state s and following the policy π is defined by *action-value function* $q_\pi(s,a)$ [77].

$$q_\pi(s,a) = \mathbb{E}_\pi[G_t|s_t = s, a_t = a] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}|s_t = s, a_t = a\right], \tag{2.5}$$

The *value function* v_π in equation 2.4 and *action-value function* q_π in equation 2.5 are unknown functions during the learning process. With the interactions between the environment and agent, the learning algorithm estimates these functions[77]. In cases, where the agent follows policy π and keeps track of average of every state, that has occurred during the interaction with the environment, will converge to the state's value $v_\pi(s)$, the same is true for the separate averages of actions taken in each state, and they will also converge to the action values $q_\pi(s,a)$[77]. These estimation methods estimate real values by averaging over many samples, in RL they are called *Monte Carlo methods*.

The goal of the reinforcement learning is to find a so called *optimal policy*. This means that there is a policy π that is equal or better than a policy π', in case its expected return is greater or equal in all the states that occurred by following these policies, so therefore $\pi \geq \pi'$ if $v_\pi(s) \geq v_{\pi'}(s)$ for all the $s \in S$. The *optimal policy* is always better or equal than other policies. Optimal policies share the same *state-value function* denoted v_* as well as

action-value function denoted q_* called optimal for all $s \in S, a \in A$.

$$v_*(s) = \max_{\pi} v_\pi(s) \tag{2.6}$$

$$q_*(s,a) = \max_{\pi} q_\pi(s,a) \tag{2.7}$$

The equation 2.7 can be simplified since the agent is taking an action a in state s and following the optimal policy, the expected return of taking this action is [77].

$$q_*(s,a) = \mathbb{E}[R_{t+1} + \gamma\, v^*(s_{t+1}) | s_t = s, a_t = a] \tag{2.8}$$

2.2.2 Selected RL methods

The focus of this section will be on the introduction of couple RL methods that are widely used. First, the Q-learning method is explored as one of the most popular model-free methods. Next, the SARSA algorithm is described. And finally, the deep learning method called deep Q-learning.

Q-learning method

The Q-learning is one of the model-free off-policy methods used in RL. In this case, the agent is learning how to act properly by interacting with the environment and experiencing the consequences of actions[87, 35]. The policy that is being followed is independent from the learned action-value function Q which directly approximates the optimal action-value function q_*, however the policy is used for action selection depending on the state, and unless state-action pairs will be updated the Q-learning will converge[77]. This is a minimal requirement for convergence.

$$Q(S_t,A_t) \leftarrow Q(S_t,A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1},a) - Q(S_t,A_t)] \tag{2.9}$$

In equation 2.9 the $Q(S,A)$ is the Q value of the state-action pair. t represents the time step and α represents the learning rate. The a_t represents the action taken based on the state S and received immediate reward R. The discount factor γ is used to balance the influence of $maxQ(S',a)$ the highest expected value in state S'. As mentioned previously, the minimal requirement for the convergence of the Q-learning is that the state-action pairs continue to be updated. The algorithm for Q-learning can be seen in algorithm 1. The main advantage and popularity of the Q-learning comes from the simplicity of the algorithm. However, it has certain limitations. Since the algorithm learns based on interaction with the environment,

Set algorithm parameters : step size $\alpha \in (0,1], small\ \varepsilon > 0$
Initialise $Q(s,a), for\ all\ s \in S^+, a \in A(s), arbitrarily\ except\ that\ Q(terminal,.) = 0$
for *Each episode* **do**
  Initialise state S
  repeat
    Choose action A from state S using policy derived from Q(e.g., $\varepsilon - greedy$)
    Take the chosen action A, observe reward R and new state S'
    $Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma max_a Q(S',a) - Q(S,A)]$
    $S \leftarrow S'$
  until *S is terminal*
end

Algorithm 1: Q-learning[77]

first it has to explore the environment and fill the Q-table, after that the agent exploits the environment and starts taking better action. Due to this, the size of the environment, or number of states the agent can achieve, can grow quickly, which influences the learning time and increases the Q-table and storage capacity. Same is true, for the number of actions that the agent can take. This is often considered when the algorithm is being considered for some application. Often the environment has a limited number of states as well as a limited number of actions to keep the size of the Q-table manageable.

SARSA method

Similarly as Q-Learning, SARSA is a model-free method of RL, however, unlike Q-learning it is a on-policy method and it was introduce in [66]. The abbreviation SARSA stands for state-action-reward-state-action. Since it is a on-policy, it means that the update of the policy is based on the actions taken during the interaction of an agent and environment. The main function for the Q-value can be seen in 2.10.

$$Q(S_t,A_t) \leftarrow Q(S_t,A_t) + \alpha[R_{t+1} + \gamma Q(S_{t+1},A_{t+1}) - Q(S_t,A_t)] \tag{2.10}$$

This function is similar to the one in Q-Learning, however, in this case the *maxQ* is changed to the $Q(S_{t+1},A_{t+1})$. The $Q(S_{t+1},A_{t+1})$ represents the Q-value chosen in new state as well as action selected based on the new state by following the policy π. As the abbreviation suggests, the Q function depends on the current state S_t, the action chosen by the agent A_t, the reward received after taking the action R_{t+1}, the new state caused by the previous action S_{t+1} and the new action A_{t+1} based on the new state. This quintuple $(S_t,A_t,R_{t+1},S_{t+1},A_{t+1})$ is used as bases for the abbreviation SARSA. The convergence of the SARSA algorithm

occurs with probability 1 to an optimal policy under the condition that all state-action pairs are visited infinite number of times and the policy converges to an ε *greedy policy* [77].

Set algorithm parameters : step size $\alpha \in (0,1], small\ \varepsilon > 0$
Initialise $Q(s,a), for\ all\ s \in S^+, a \in A(s), arbitrarily\ except\ that\ Q(terminal,.) = 0$
for *Each episode* **do**
  Initialise state S
  Choose action A from state S using policy derived from Q(e.g., $\varepsilon - greedy$)
  repeat
    Take the chosen action A, observe reward R and new state S'
    Choose new actionA' based on new stateS' using policy derived from Q(e.g., $\varepsilon - greedy$)
    $Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma Q(S',A') - Q(S,A)]$
    $S \leftarrow S'$; $A \leftarrow A'$
  until *S is terminal*
end

Algorithm 2: SARSA[77]

Deep Q-learning method

As its name suggests, deep Q-learning is the combination of deep learning and Q-Learning. The simple algorithm of Q-Learning allows for fast implementation in cases, when the environment is relatively simple and does not have high number of states as well as too many actions, that can be taken by the agent. Otherwise, when there are millions of possible states and actions, Q-Learning becomes impractical, and the Q-table becomes too big. This problem was addressed in [54] by approximating the Q function with the use of deep neural networks(DNN). In this case, the neural network called deep Q-network(DQN) was initialised with random weights and trained to approximate the action value function $Q(S,A,\theta)$ [54]. The number of output nodes in this deep Q-network is equal to number of actions that the agent can take, or to the action-space size and the number of input nodes was dependent on the state representation of the environment(E.g. in [54] raw video data input). The DQN is trained by minimising the sequence of loss functions $L_i(\theta_i)$ that changes at each iteration i [54].

$$L_i(\theta_i) = \mathbb{E}_{S,A\sim\rho(\cdot)}\left[(y_i - Q(S,A;\theta_i))^2\right] \tag{2.11}$$

The $y_i = \mathbb{E}_{S'\sim\varepsilon} = [r + \gamma max_{A'} Q(S',A';\theta_{i-1})|S,A]$ is known as target for the i iteration and $\rho(S,A)$ is the probability distribution over states S and actions A that is referred to as *behaviour distribution* [54]. The presented target is dependent on the weights of the network.

The gradient of the loss function $\nabla_{\theta_i} L_i(\theta_i)$ is then:

$$\nabla_{\theta_i} L_i(\theta_i) = \mathbb{E}_{S,A\sim\rho(\cdot);S'\sim\varepsilon}\left[(y_i - Q(S,A,\theta_i))\nabla_{\theta_i} Q(S,A;\theta_i)\right] \tag{2.12}$$

In equation 2.12, the θ_i is the vector that contains the weights of the neural network at the current iteration and θ_{i-1} are the weights of the previous iteration. The $Q(S,A,\theta)$ is the value function that is approximated by the neural network and $\nabla_{\theta_i} Q$ is the gradient of this function. The deep Q-Learning algorithm introduced in [54] can be seen in algorithm 3.

Initialize replay memory D to capacity N
Initialize action-value function Q with random weights
for *episode = 1,M* **do**
  Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$
  for *t = 1, T* **do**
    With probability ε select random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_{t+1} and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in D
    Sample random minibatch of transitions $(\phi_j, a_j, r_{j+1}, \phi_{j+1})$ from D
    Set $y_j = \begin{cases} r_{j+1} & \text{for terminal}\phi_{j+1} \\ r_{j+1} + \gamma \max_{a_{t+1}} Q(\phi_{j+1}, a_{t+1}; \theta) & \text{for non-terminal}\phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation2.12
  end
end

Algorithm 3: Deep Q-Learning[54]

One of the approaches used in deep Q-Learning, is a technique known as experience replay [54]. This technique acts as a memory of experiences or a buffer called *replay memory*. The experiences are stored at every time step. During training, experiences are sampled randomly in so called *minibatch* updates(containing multiple experiences) from the replay memory. The purpose of this technique is to minimize the oscillations and avoid local minima. Except the experience replay technique, the separate target network as stabilisation was introduced in [55]. In this case, the agent contains two identical networks, where one network does the training and the target network is regularly updated from the training network and is used in temporal difference target estimation. In between the update intervals,

the target network is fixed. The loss function is then defined as follows.

$$L_i(\theta_i) = \mathbb{E}_{(s_t,a_t,r_{t+1},s_{t+1})}\left[(r_{t+1} + \gamma \cdot \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}; \theta_i^-) - Q(s_t, a_t; \theta_i))^2\right] \tag{2.13}$$

Actor-critic method

Another method, which will be briefly mentioned is the actor-critic method. Unlike previously mentioned methods, which are based on the calculation of the expected value of executing an action in a state(value-based methods), this method is a policy-based method. The actor critic method has two networks, the actor decides which action is taken, while the critic provides the knowledge of how good the action taken was [30]. In this case, the actor handles for the policy function and the critic evaluates the current policy through the value function.

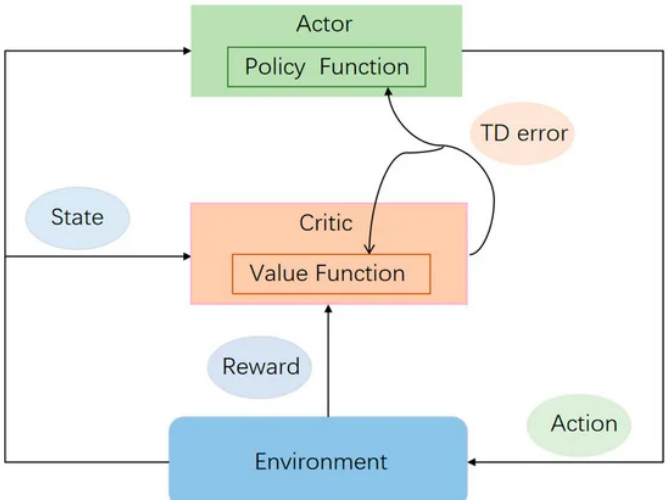


Fig. 2.3 Scheme of actor-critic method[46]

Presented RL methods are one of the most commonly used methods in RL. One of the objectives of this thesis is to use one of the RL methods for the control task of a continuum robot. In this task, the state space is not discrete, and therefore regular Q-Learning would not be suitable for such a task. This was addressed in deep Q-Learning, where Q function is approximated by neural network. By using the so-called deep Q-network(DQN) the state space which is not discrete is not a problem. Therefore, it was decided to use the DQN to construct a control algorithm based on a single DQN agent that would learn on a mathematical model, which will be described in the following chapters.

2.3 Mathematical modeling of continuum robots

The necessary component of every robotic system is the controller responsible for actually giving the commands to the actuators to steer the system into the desired positions. This can be a challenging task depending on the complexity of the system and the complexity of the task.

To do this task well it is important to have some kind of a knowledge about the system. To describe the kinematics of continuum robots it is important to map the relations between multiple coordinate systems. The approaches that focus on different kinds of mappings and extraction of the knowledge about the continuum robot kinematics can be categorised into model-based methods and model-free methods. The model-based methods focus on the use of classical analytical methods to map the relations, whereas the model-free methods employ learning to figure them out either from the robot directly of the datasets.

2.3.1 Model-based methods

The model-based methods use more traditional analytical approaches to describe the kinematics of continuum robots. One of the commonly used approaches is called the constant curvature (CC) approximation method [61]. The premise of this method is based on parameterisation of the continuum robot's configuration space as a constant curve, assuming that the robot shape follows this curve and this curve can be represented by thee variables [33]. By simplifying and reducing the complexity this model gains computation speed which can be beneficial when fast calculations are needed. One of the restrictions of this methods is in the actuation that is used. It assumes that the robot is controlled by cables that are equally placed around the center line of the robot. The robot also has to have uniform shape as well, so that symetrical actuation of the robot can be achieved. Similarly in [39, 89], so called piecewise constant-curvature approximation(PCC) method was shown. This method simplifies the robot as a number of smaller serially connected tangent sections and each section is further represented as constant curvature arc [61]. That can be represented by parameters κ, ϕ, representing curvature of the robot and angle of rotation of bending plane of the robot. From these parameters, the task space coordinates can be obtained using by using multiple approaches like Frenet-Serret frames, Arc geometric mapping, integral representation and others that can be found in more details in [89]. Another approach that is widely used in this domain is the pseudo-rigid body representation[65]. This model is approximating the continuum robot backbone as a series of rigid links. These links are interconnected with torsion springs. The length and stiffness of each of the links is highly dependent on the forces and moments acting on the robot[76], therefore it is crucial to set these parameters correctly.

One of the more elaborate models is based on the Cosserat rod theory, which is described in [82]. In this model, the authors take into account the effect of material nonlinearities, when the robot has to position itself into complex shapes. It is also geometrically exact for these complex shapes, torsion and shear[82].

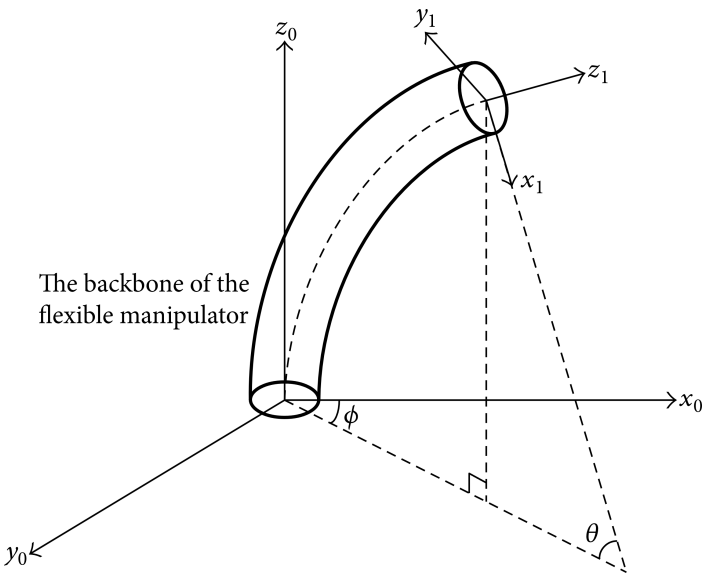


Fig. 2.4 Kinematic approximation of flexible manipulator[79]

These methods represent traditional analytical approach to the the modeling and kinematic description. Currently one of the most commonly used approx is the constant curvature method. Even though, it is not the most precise, it is simple understand, implement and fast to compute. The advantages and disadvantages of the presented models may vary depending on the situations and unique parameters of the robots. Different kinematic structures, loading scenarios or number of robot's segments can be an important factor when choosing the model.

2.3.2 Model-free methods

If the kinematic structure is too complex and complicated the use of model-free methods can be more suitable than the traditional model-based approaches relying on the simplifications and analytical methods. The model-free methods are based on learning the mapping from gathered or generated data [51]. Depending on the data forward or inverse kinematics can be learned, possibly with high precision. One of the concept used for such learning are neural networks. This approach was shown in [27], where neural networks were used to learn the inverse kinematics of soft three cable driven continuum robot. After the learning this approach has shown good result of 4.2 mm of mean error. The disadvantage of this approach

is that this model can only be used for specific continuum robot that was presented in this paper. To use this model for different robot, new data would be needed for the training of the model. A different approach was shown in [102], using reinforcement learning to learn how to actuate a soft robot, which is pneumatically actuated. The goal was to learn how to pressurise the in order to manipulate the robot into a desired place. The robots movement was allowed only in one axis. An approach called Q-learning was used to train an agent to navigate the robot from any given state, by selecting an action that moves the robot towards the random desired position, ending the training episode when the agent reached the target with an acceptable error. The agent is then acting based on the learnt policy during the training phase. The agent can select from four predefined actions and the state space in which the robot moves is also divided into finite number of small spaces. In [25] a forward dynamic model was trained using recurrent neural networks. This was used to generate sample trajectories for the final policy training. This way a closed loop predictive controller was trained.In [48], the authors propose a reinforcement learning control of tendon driven continuum robot based on in-explicit prior knowledge. The inexplicit knowledge serves as guidance to the robot with the approximate direction in which it should start to move. By using this knowledge, the exploration of the space can be shorter and learning process faster. In this way, the policy is guided to the approximate direction in which lies the target point. Use of Deep Q-learning was also shown in [69], controlling a soft continuum robot. The robots actuation was based on pneumatics and the bending was restricted in one axis. The Deep Q-learning was also shown in [37] used for control of cable driven continuum robot. The robot was controlled by four cables, where each pair was controlled by one agent trained to position the robot in one axis in order to position the robot into overall desired position. In order to improve the performance adaptive actions depending on the euclidean distance to the target were introduced. The proposed adaptive actions improved the performance a shortened the training time. In [56], the authors created reinforcement learning based control algorithm. The robot used to test this algorithm consists of three segments, each of which consists of three pneumatic artificial muscles. The proposed algorithm is using Ensemble Lightweight model-free reinforcement learning Network. This network learns the control policy by using an actor-critic method with multiple actors and critics. In this case, the state space and the action space are continuous with a discrete time step. In the proposed algorithm, the agent takes an action based on Q-value using multiple policies. Similar approach of using the reinforcement learning was also shown in [8]. The authors proposed a control algorithm based on a deep reinforcement learning approach called Trust Region Policy Optimization (TRPO) [70]. The learning process was done on simulation of pneumatically actuated soft robot, where the model was learnt by Long-short Term Memory network. In this paper, the

TRPO is actualy learning the controlling task by having a trust region that avoids the large updates of the policy network which in case of misstep can undo all the learning up to that point.

The application of model-free techniques in the continuum robotics domain is less explored and has not been as thoroughly investigated as model-based approaches. These techniques rely on data and utilize it to understand the kinematics of continuum robots. Their primary benefit lies in situations where the mathematical model is overly complex or unavailable and so learning based methods may be faster to develope. Despite the time-consuming nature of the learning process, subsequent implementation is rapid, leading to model with relatively high accuracy.

2.4 Chapter summary

In chapter 2 a theoretical background of machine learning and its use in the domain of continuum robotics is given. Machine learning is divided in three main groups called supervised learning, unsupervised learning and reinforcement learning. The supervised and unsupervised learning is based on using collected or generated data. Supervised learning uses labelled data, which means that inputs are paired with outputs, and supervised learning algorithms are trained to associate inputs with correct outputs. The two main approaches in supervised learning are classification and regression. The classification problems aim to assign or predict the correct labels to the input values, where the number of labels is a finite number. In regression problems, the aim is to predict a value. In unsupervised learning, the data that are used are unlabelled. In unsupervised learning, the algorithms are trained to determine the patterns or associations within the data. The last group of machine learning algorithms are reinforcement learning algorithms. The reinforcement learning is based on training an agent to select the proper action for the controlled system. The agent is learning from the interactions with so called environment, which contains the controlled system. To describe these interactions Markov decision process is being used, which is defined by a tuple $< S, A, R, T >$. This tuple all possible states that can occur in the environment, all possible actions that an agent can take, reward signal which is a feedback for the agent from an environment, how the action influenced the system and finally the the transition function calculating the probabilities of transitioning to new state. Finally, the modeling of continuum robots was discussed, where model-based methods and model-free methods are used. Model-based methods use classical analytical approaches do describe the relations between the actuation and the robot position, whereas the model-free method uses learning-based techniques to learn these relations from data or the robot itself. Model-free

methods use machine learning techniques to obtain these relations. The biggest advantage of these methods is especially in the cases when the analytical model is too complicated or nonexistent. The data-based method can learn from the robot's data or interactions with the robot to learn how to position and navigate the robots with high precision even in cases when the analytical models cannot.

Chapter 3

Hardware

To carry out the experiments in the real world, hardware to do so is necessary. For this reason, this chapter presents a design of test bench that will be used for experimental verification of multiple kinematic structures and control algorithms of continuum robots. This chapter also shows the different kinematic structures of continuum robots that will be used for the experiments.

3.1 Experimental setup design

To test the proposed control algorithms, kinematic structures of cable-driven continuum robots have been developed by the ARM-Lab team during the MISBOT project. During this project a concept of minimally invasive surgery robot workspace was developed, where collaborative robotic manipulator was paired with a continuum robot used for inspection purposes equipped with endoscopic camera. To test the developed kinematic structures a

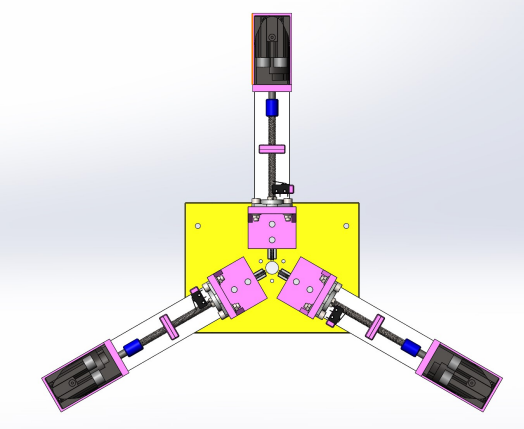


Fig. 3.1 Back view of testing stand

testing setup has been made. The purpose of this setup is to easily change the kinematic structure and be able to test its properties by applying various control algorithms. The testing setup consists of a yellow front plate, on which the kinematic structure and motor units are mounted on. There are three motor units, each controlling one cable of the robot. Motor units are identical and can be added if different structure with more cables will be tested in the future. The unit consists of a servo motor, which is connected to a threaded rod with a pitch of 8mm per rotation. The servo motor is placed in a 3D printed motor mount and the end of the threaded rod is mounted in a bearing mounted in a 3D printed mount as well. A runner is moving on the threaded rod based on its rotation and the cable that is controlling the robot is connected to it. The runner also has a guide on the bottom part to prevent rotation and so that the runner can only perform a linear movement. The cable can then be pulled back and forth according to the control algorithm. These motor units can be easily adjusted in case

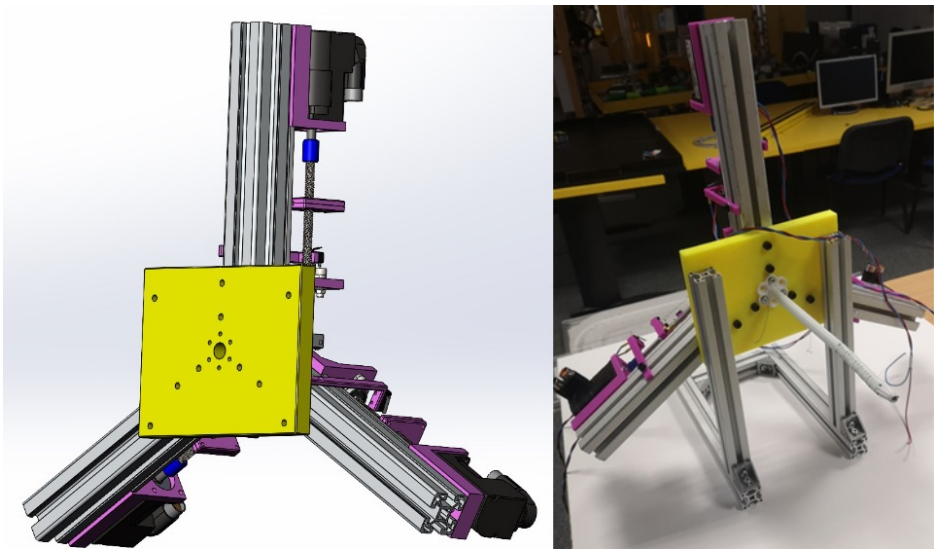


Fig. 3.2 Testing stand, front view

more linear movement is necessary, by extending the threaded rod. At the end of the linear path of the runner a limit switch is added to prevent the movement out of the set boundaries. The premise of the movement is that each kinematic structure starts with a straight position, parallel to the ground. All cables are tensed so that the kinematic structure can be straight. In this initial position, all runners are in the centre of its boundaries for their movement. Based on the actions, the motors are spinning so that the desired shortening or extension of the cables is achieved and the robot can be positioned to the desired location. The last important part of the testing setup are the cable guides. These are placed right after the motor units. Their purpose is to guide the cables from the moving runner on the motor unit, through the central hole in the front plate, to the kinematic structure of tested continuum robot. The

guides are made of v grooved bearing to limit the friction of the cables to the minimum to ensure smooth movements and longevity of the used cables. To ensure the proper control

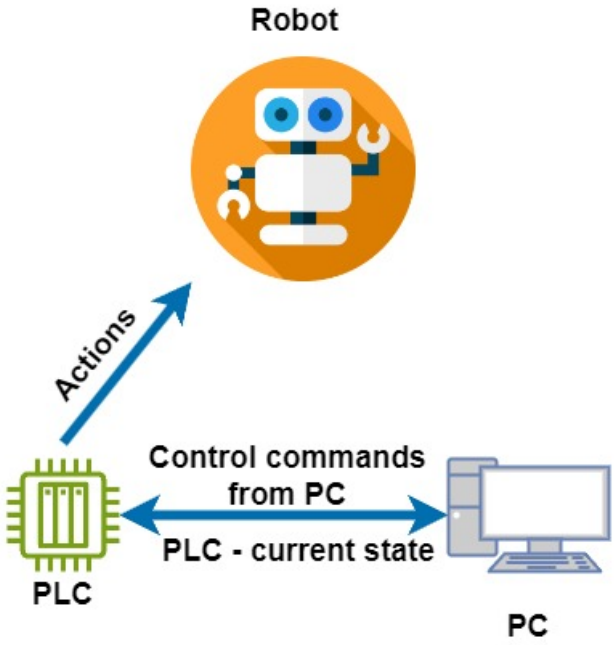


Fig. 3.3 Control scheme

of the testing bench, the motors are connected to the PLC, which is controlling the motors and receiving the signal from the limit switches in case of the collision with the runner. The PLC receives the linear distance that the runner should travel as well as the direction, it is then responsible for calculating the rotations necessary for achieving the desired linear travel. The control commands are sent from MATLAB by serial communication to the PLC. Once the PC sends the commands, the PLC confirms that the commands were received and the commands are being executed. After the command execution, the PLC sends message that the execution is done and it is waiting for another set of commands. It is programmed to have two sets of commands. One receiving the linear travel for each motor and the second one to go back to the initial position that it started from. Using these two commands, it is possible to conduct all necessary experiments with various kinematic structures of continuum robots and use this testing bench for the research of novel kinematic structures as well as control algorithms for cable-driven continuum robots.

3.2 Kinematic structures

To test the proposed control algorithms, real hardware is necessary. The development of multiple kinematic structures of cable-driven continuum robots was part of the MISBOT project. During this project, we have worked on the concept of a robotic system that would assist the surgeon during minimally invasive surgeries. The purpose of the project was to combine a collaborative robotic manipulator and a continuum robot prototype as its effector. The continuum robot, was then equipped with an endoscopic camera and the system was

Fig. 3.4 MISBOT project

showcased on a figurine. During the development process of the project, three different kinematic structures were developed. All had the same length of 110 mm and diameter of 12 mm. The first of three structures can be seen in Fig. 3.5. This kinematic structure is made of three flexible smaller sub-segments. Each sub-segmnet structure is 3D printed using flexible TPU and have top and bottom part which is made of rigid PLA. The three sub-segments are then connected with one central backbone made of silicone tube. The controlling cables are routed through all the three segments. The cables are equidistantly placed around the centre axis of the robot. This way each cable should influence the robot's movement evenly. The flexible structure is made with holes in the structure that are regularly placed, creating a pattern. During the movement of the robot, the structure is deformed and acts as a spring. By creating this kind of force, that is acting against the cables that are being pulled, a better reliability and repeatability of the movement of proposed structure can be achieved. The

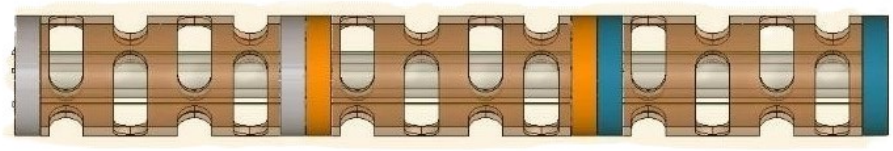


Fig. 3.5 CR structure 1

second kinematic structure is made of multiple rigid sub-segments. The sub-segments are 3D printed using PLA. The top part of a sub-segment is made to fit into the bottom part of another sub-segment. By adding a central flexible backbone made of flexible tube, same as in structure number one, the movement of sub-segments is limited but still possible in respect to each other. The rotation of sub-segments is not possible, since the cables are routed through

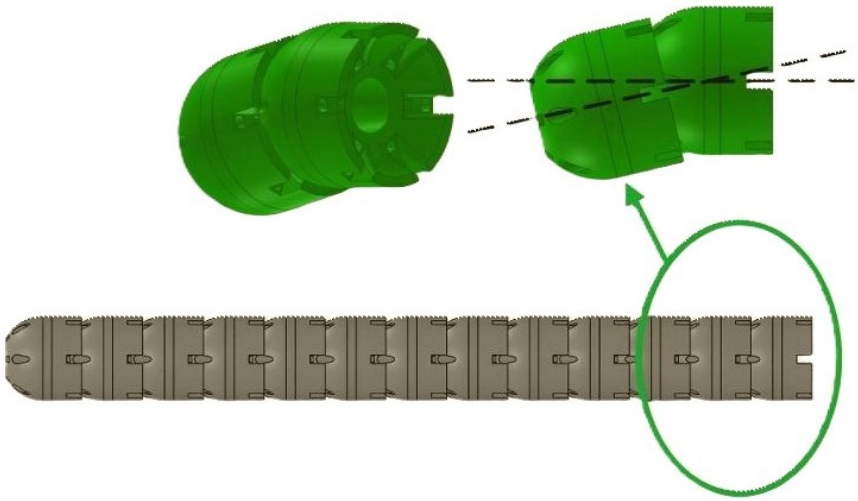


Fig. 3.6 CR structure 2

all segments to the last segment at the tip position. The cables are also placed equally around the central axis of the robot to ensure equal influence of each cable on the control of the robot. The last kinematic structure is also made of multiple rigid sub-segments, which are made to fit into each other. The bottom part of the sub-segment has a V groove, in which another sub-segment fits in. This groove also allows for the rotation of sub-segment in one axis perpendicular to the groove. The direction of the grooves is rotated by 90 degrees at every sub-segment, so that the movement into every direction is possible. As in the previous cases, the central backbone is made of flexible tube. Around the central backbone, three

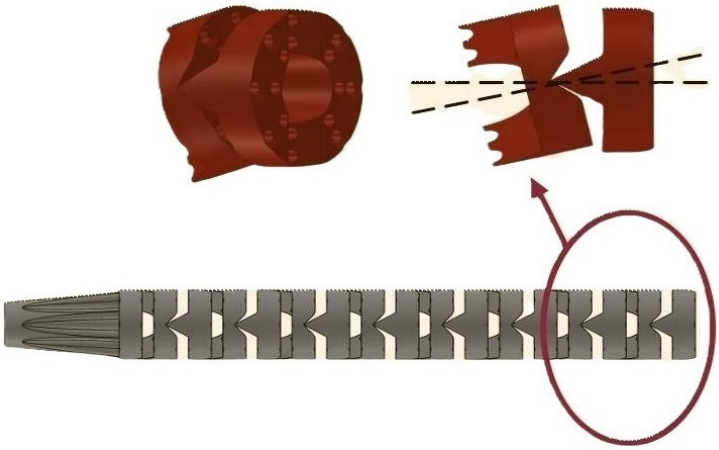


Fig. 3.7 CR structure 3

cables are evenly distributed and routed through all sub-segments all the way to the last segment at the tip of the robot.

3.3 Chapter summary

In chapter 3, a testing bench was presented. The test bench contains universal motor units. These contain servo motors and a mechanism that translates the rotational movement of the motor to the translational movement of a runner, which is connected to a controlling cable of a cable-driven continuum robot. The servo motors were selected for their high precision and wide variety of options. In the future, these motors allow for the monitoring of electric currents, which might give some feedback on the tension of the cables. Each motor unit controls one cable of three cable driven continuum robot, so currently there are three motor units, however, if needed more can be added in the future. The bench is controlled by a PLC which communicates with the PC using MATLAB script and receives the elongations or shortenings of the cables. The purpose of the test bench is to easily test and evaluate different kinematic structures of continuum robots and different control algorithms. Three different kinematic structures were also presented. These were made by the ARM-Lab team during the MISBOT project. Each of the structures has its advantages and disadvantages, which will be tested later in this work.

Chapter 4

Proposed methods

One of the objectives of this thesis is the development of a control algorithm using learning-based methods. To begin with in this chapter, we will start with the implementation of the constant curvature model. This model will be used as part of a reinforcement learning based controller, which is one of the proposed controllers in this chapter. The second approach proposed in this chapter will be based on the method called supervise learning. This approach is using labeled data to train a neural network model for predicting actions of the robot.

4.1 Modelling of continuum robot

In this chapter, the implementation of the mathematical model that was mentioned in the previous chapter 2.3.1, is going to be presented. This mathematical model is going to be implemented as part of one of the control algorithms for controlling the continuum robots. The modelled robot is a 1 segment continuum robot, consisting of three flexible sub-segments that are interconnected with central backbone. The robot is controlled by 3 tendons that are shortened or extended in order to position the robot in the Cartesian coordinate system.The modelled robot can be seen in Fig. 4.1. The cables controlling the robot are placed equally around the central axis of the robot, distanced 5 mm away from the central axis. The total length of this robot is 110 mm in total. The maximal cable displacement is set to 10 mm. The three cable configuration is a minimal number of cables necessary for the movement in 3D.

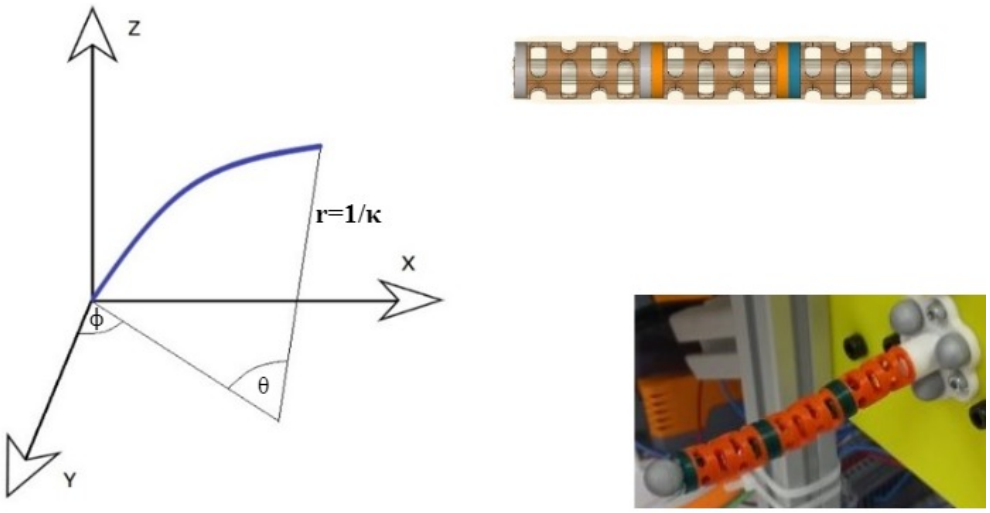


Fig. 4.1 Prototype of the modelled robot

4.1.1 Mathematical model of continuum robot

The control algorithm will be trained on a mathematical model of the continuum robot, therefore, one of the criterion of the mathematical model is its speed of computation. Due to this condition, it was opted to use the constant curvature approach, which simplifies the robot model and is fast to compute. This model, is based on mapping between three coordinate systems, actuator coordinate system, configuration coordinate system and task coordinate system. The robot is approximated as a curve that is described by the arc parameters, that are part of the configuration coordinate system. The actuator coordinate system consists of cable lengths, that are actuating the robot. The configuration coordinate system, contains the arc parameters κ, ϕ, θ. The task coordinate system, consists of the Cartesian coordinates of the end point at the tip of the robot. The coordinate systems can be seen in Fig. 4.2. The mapping

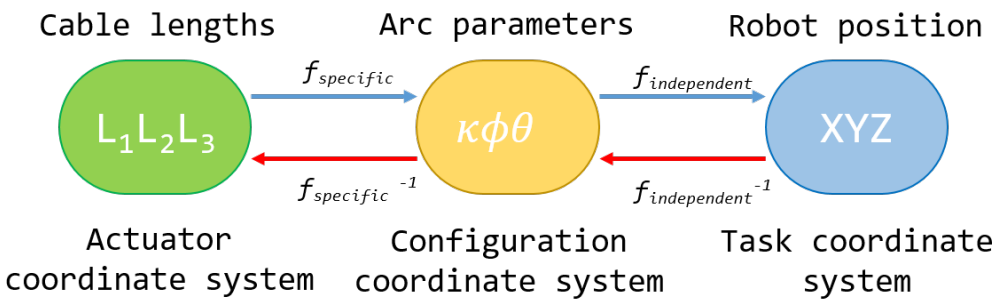


Fig. 4.2 Mapping

can be done in two directions. In one direction, the mapping is done between the actuator coordinate system and the configuration coordinate system, called $f_{specific}$, and between the configuration and task coordinate system, called $f_{independent}$. Or, in the opposite direction, where the mapping is labelled with the index -1. The labelling "specific" and "independent" implies that the mapping between the actuator and configuration coordinate system is specific to the particular robot that is being modelled and the mapping between the configuration and task coordinate system is not dependent on specific design or actuation system of the modelled robot.

The mapping in task coordinate system is therefore defining the robots' pose in the Cartesian coordinate system. It is defined by the point **p**, which is at the tip of the robot, and the backbone of the robot is viewed as an arc with radius r. The coordinates of point **p** are defined as $\mathbf{p} = [r(1-\cos\theta) r\sin\theta]^T$ in the $x-z$ plane, and angle θ represents the rotation about the y axis. This representation only accounts for the movement in the 2D space and, therefore, to account for the movement in the 3D space, the plane in which the robot bends is described by the angle ϕ.

$$\mathbf{T} = \begin{bmatrix} \mathbf{R_z}(\phi) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R_z}(\theta) & \mathbf{p} \\ 0 & 1 \end{bmatrix} \tag{4.1}$$

The angle ϕ is representing the rotation about the z axis. The equation 4.1 is known as the transformation matrix. The arc parameters are then used to further develop the matrix.

$$\mathbf{T} = \begin{bmatrix} \cos\phi\cos\theta & -\sin\phi & \cos\phi\sin\theta & \frac{\cos\phi(1-\cos\theta)}{\kappa} \\ \sin\phi\cos\theta & \cos\phi & \sin\phi\sin\theta & \frac{\sin\phi(1-\cos\theta)}{\kappa} \\ -\sin\theta & 0 & \cos\theta & \frac{\sin\theta}{\kappa} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.2}$$

The radius of the arc that is representing the robot backbone, which is defined as $r = 1/\kappa$ and the aforementioned angle $\theta = \kappa s$, where s is a point between 0 and l. In equation 4.2, the developed matrix can be seen. This transformation matrix describes the relationship between the arc parameters and X, Y, Z coordinates, which is labelled in Fig. 4.2 as $f_{independent}$. Another mapping that can be seen in this figure is called $f_{specific}$. This mapping is describing the relations between the lengths of tendons that are actuation the robot and the arc parameters that are describing the robot. This step consists of angle ϕ, θ and curvature κ, which are defined in equations 4.3,4.4,4.5.

$$\phi = \tan^{-1} \frac{\sqrt{3}(l_2 + l_3 - 2l_1)}{3(l_2 - l_3)} \tag{4.3}$$

$$\kappa = \frac{2\sqrt{l_1^2 + l_2^2 + l_3^2 - l_1 l_2 - l_1 l_3 - l_2 l_3}}{r(l_1 + l_2 + l_3)} \tag{4.4}$$

$$\theta = \frac{2\sqrt{l_1^2 + l_2^2 + l_3^2 - l_1 l_2 - l_1 l_3 - l_2 l_3}}{3r} \tag{4.5}$$

The equation 4.3 is the function of angle ϕ. This angle corresponds to the rotation of the plane in which the robot bends. Therefore, it is responsible for the direction in which the robot bends in free space. Equation 4.4 describes the curvature of the arc that corresponds to the robot backbone with radius r. This parameter can be further developed as $\kappa = 1/r$. In the following equation 4.5 the bending angle θ of the robot backbone can be seen. This parameter is responsible for the angle of the robot in the bending plane. It is the angle of the aforementioned backbone in the plane that is rotated by the angle ϕ. The workspace generated by this model can be seen in Fig. 4.3.

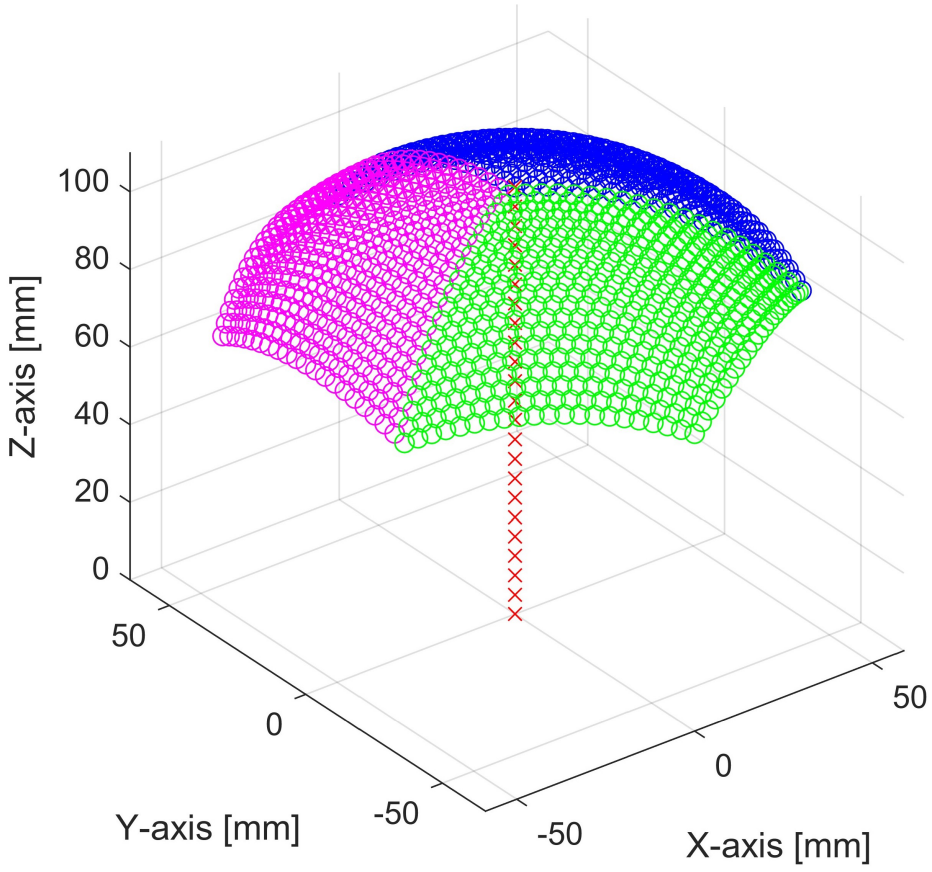


Fig. 4.3 Workspace of the continuum robot

4.2 Reinforcement learning based control algorithm

The main objective of this thesis is to research and develop a control algorithm for continuum robot's positioning in static 3D space without any obstacles. The use of numerical and analytical methods seem as viable option, however the learning methods have shown great capabilities and space for further research, it was opted to use machine learning for this task. Even though the learning process can be slower than traditional methods, the learning methods are able to learn more complex and complicated models to control the robots, and once the models are learned the deployment is fast. Machine learning offers a wide variety of approaches and methods that could be used for control tasks. The nature of continuum robot kinematics and overall design of continuum robots can be complicated and has many variable parameters, therefore a learning interaction between the robot or its model and control algorithm seems as a natural option. These types of methods are generally referred to as reinforcement learning methods.

All of the reinforcement learning methods can be described by the Markov Decision Process(MDP). MDP is a mathematical framework that is used to describe the decision making processes that are ongoing between the agent and the environment[77]. The MDP can be represented as a tuple $< S,A,P,R,\gamma >$, denoting the state, action, transition model, reward and discount factor γ.

The state S is determined by the mathematical model that was introduced in 4.1.1 and is part of the environment. Every state $s \in S$ is determined by $\delta_x, \delta_y, \delta_z$. These deltas can also be seen as $(x_{tar} - x_{act}), (y_{tar} - y_{act}), (z_{tar} - z_{act})$. The tar represents the coordinates of the target point, and act represents the coordinates of the actual position achieved. The state therefore represents the distance of the actual state to the target position. The robot is attached on a horizontal plane and its bending angle is limited to the 90° angle.

The actions are taken based on the agent's instructions. There are defined 12 possible actions that the agent can choose from. The first 6 defines all combinations of shortening and extending the tendons by 0.1 mm and the following 6 are the same combinations but with 0.2 mm. These combinations of actions are in reality signals for the motors to either shorten 1 tendon and extend 2 tendons or shorten 2 and extend 1. The action space A is then defined as $A = \{a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}\}$ and every action $a_N \in A$ where $N = 0, ..., 11$. The actions are then defined as $a_0 = \{+0.1, +0.1, -0.1\}$, $a_1 = \{+0.1, -0.1, +0.1\}$, $a_2 = \{-0.1, +0.1, +0.1\}$, $a_3 = \{-0.1, -0.1, +0.1\}$, $a_4 = \{-0.1, +0.1, -0.1\}$, $a_5 = \{+0.1, -0.1, -0.1\}$, $a_6 = \{+0.2, -0.2, -0.2\}$, $a_7 = \{+0.2, -0.2, -0.2\}$, $a_8 = \{+0.2, -0.2, -0.2\}$, $a_9 = \{+0.2, -0.2, -0.2\}$, $a_{10} = \{+0.2, -0.2, -0.2\}$, $a_{11} = \{+0.2, -0.2, -0.2\}$. Every action a_N contains 3 values, each value is assigned to one of the tree motors and represents the amount that the tendon connected to that motor

has to be shortened or extended(+/- is assigned to extending/shortening action of the motor respectively). At every step, agent takes an action a_N, based on the current reward. By using these steps, the agent should be able to cover the movement of the robot in the entire workspace within one episode. Using smaller actions as steps would require more steps within an episode to achieve the target point and more time.

After an agent takes an action, the numerical feedback signal called reward R is sent back to the agent. The reward function is defined as follows:

$$r_t = -\delta_{norm} + \mu_t + \begin{cases} -1 & \text{if the robot is in the starting point} \\ -1 & \text{if episode end, desired point } \pm 0.5mm \text{ not reached} \\ +1 & \text{if reached the desired point } \pm 0.5mm\text{, episode done} \\ +1 & \text{if the robot has reached the desired point } \pm 0.6mm \end{cases} \tag{4.6}$$

$$\delta_{norm} = (\delta_t / \delta_{max}),\ \delta_{norm} \in < 0, 1 > \tag{4.7}$$

$$\mu_t = \begin{cases} +1 & \text{if } \delta_t - \delta_{t-1} < 0 \\ -1 & \text{if } \delta_t - \delta_{t-1} > 0 \end{cases} \tag{4.8}$$

The reward signal denoted r_t consists of three values. One of the values is δ_{norm}, which is the euclidean distance between the achieved point and the desired point, denoted as δ_t, however, this distance is then normalized between $< 0, 1 >$. The parameter μ_t rewards the agent in case the distance between the actual point and the desired point has decreased and punishes when the distance has increased based on the current and previous Euclidean distance between the actual point and the desired point of the robot. The last parameter is punishing or rewarding the agent with the ± 1 depending on the conditions. The parameter is set to -1 in two cases. In case number one the parameter is set to -1, when the robot is stuck in the initial position an is used as punishment for the agent. In the second case, the parameter is set to -1 for the case when the number of steps is 400, which is the maximal number of steps per episode and the robot has not reached the desired point within the $\pm 0.5mm$ perimeter. The other two cases of this parameter serve as a reward for the agent. In the first case, the agent is rewarded with $+1$, if the robot reaches the desired point within the $\pm 0.5mm$ perimeter (the episode is ended in case as well). In the second case, the agent is rewarded with $+1$ if the robot is in the perimeter of $\pm 0.6mm$ of the desired point, but the episode is not yet finished.

The main idea of the control algorithm is that an agent is going to interact with the environment and based on the feedback from the environment, the agent learns the proper policy to control the robot. The proposed control algorithm is based on deep Q-learning introduced in previous chapter 2.2.2. The algorithm consists of three main components. One

of them is the environment. The environment consists of mathematical model of continuum robot, described in section 4.1.1, which is controlled by three motors. The environment receives the actions from agent for the motors and the output of the model is the difference between the coordinates of the achieved position and target position defined as $\delta_x, \delta_y, \delta_z$. The deltas are then sent to the replay buffer as *states* of the environment. The agent's job

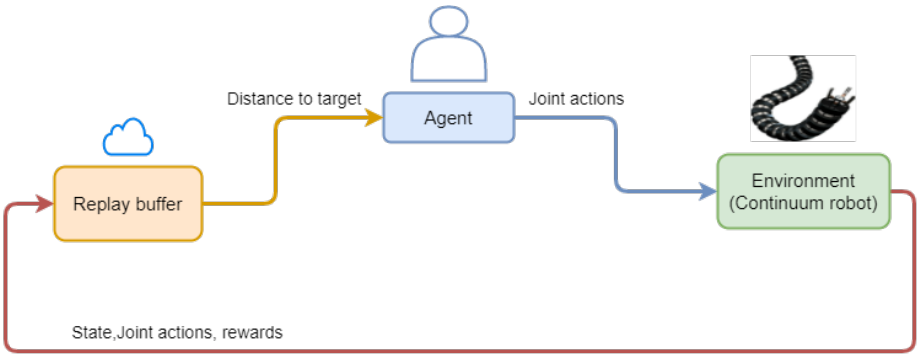


Fig. 4.4 Control scheme

is then to decide what action to take based on the state and reward that has been received from the environment. The single agent learns from the environment. The agent contains two networks, one main and the other one called target network. The main network is being trained, and its weights are used to update the weights of the target network based on the update interval. The main network is used to approximate the action-value function $Q(s,a,\theta)$, θ being a vector that contains the weights of the main network. The output of the target network $Q^*(s_{j+1}, a_{j+1}, \theta^-)$, is used to calculate the target y_j.

$$y_j = \mathbb{E}_{s_{j+1}\sim\kappa}[r_j + \gamma max_a Q^*(s_{j+1}, a_{j+1}, \theta^-)|s_j, a_j] \tag{4.9}$$

The target y_j is then used to calculate the loss function, which is minimized during training. The loss function can be seen in the equation 4.10:

$$L_j(\theta_j, \theta_j^-) = \mathbb{E}_{s,a\sim\rho}[(y_j - Q(s_j, a_j; \theta^-))^2] \tag{4.10}$$

The policy used for training was $\varepsilon - greedy\ policy$. This policy uses the parameter ε that determines how often the agent is likely to take an exploratory action, $\varepsilon \in < 0, 1 >$, the higher the epsilon, the more likely is that the agent takes a random action to explore. At the beginning of the learning algorithm, that can be seen in algorithm 4, the parameter ε is set to 1 to explore the space, and with the time the parameter decays so that random actions are less frequent. Another method used in this algorithm is called the experience replay method. This method was also described in 2.2.2. The replay memory consists of maximal

Set algorithm parameters $\alpha, \gamma, \varepsilon$, mini-batch size, update interval
Set target point $(X_{tar}, Y_{tar}, Z_{tar})$
Initialize replay memory D to capacity N
Initialize action-value function Q with random weights
for *episode = 1,M* **do**
  Initialize episode reward, step counter
  Reset state
  for *t = 1, T* **do**
    With probability ε select random action a_t
    otherwise select $a_t = \max_a Q^*(s_t, a_t; \theta)$
    Execute action a_t and observe reward r_{t+1} and state s_{t+1}
    Store transition $(s_t, a_t, r_t, s_{t+1}, done)$ in D
    Sample random minibatch of transitions $(s_j, a_j, r_{j+1}, s_{j+1}, done)$ from D
    Set $y_j = \begin{cases} r_{j+1} & \text{for terminals} s_{j+1} \\ r_{j+1} + \gamma \max_{a_{j+1}} Q(s_{j+1}, a_{j+1}; \theta) & \text{for non-terminals} s_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(s_j, a_j; \theta))^2$ according to equation 2.12
    Update weights from Main network to Target network according to update interval
  end
  Epsilon decay
end

Algorithm 4: Deep Q-Learning control algorithm

capacity of 10 000 experiences, where each experience consists of (state, action, reward, new state, whether the episode is done or not). During learning, the experiences were randomly selected from replay memory in the so-called minibatches of 100 experiences that are used for training the network. The networks are fully connected neural networks, they have 3 input neurons for the $\delta_x, \delta_y, \delta_z$ as input, 12 output neurons(each for every possible action) and 3 hidden layers, each 340 neurons, followed by ReLU activation functions and Adam optimizer. The network was trained using an off-policy training, and the robots' task was to achieve a desired position that was set at the beginning of the training. It was trained for 1500 episodes and was set to have a maximum of 400 steps per episode. After each episode, the robot was reset to its initial position and started again. At every episode, the robot started from its initial straight position and was trying to get closer to the target with every step. The episode ends either when the agent reaches the maximum number of steps or gets to the target within $\pm 0.5mm$. The experiment with this algorithm is presented in chapter 5, where the capabilities of this controller will be tested. The result of this experiment will be discussed in chapter 6.

4.3 Supervised learning

Another approach that is proposed as a learning-based controller is using supervised learning. The supervised learning is using data to learn how to actuate the robot in order to get it in to the desired positions. In this case the supervised learning model is going to predict real values of cable displacements, therefore it will be a regression type of model. The regression model relies on the data that were collected from the continuum robot prototype described in Chapter 3 Fig. 3.5. For data collection, it was opted to generate 30 000 different random actions for the three motors controlling the cable of the robot in order to cover as much of the work space of the robot as possible. Before every action, the robot started from the initial position, which was a straight position parallel to the ground and every cable was at the centre of its range of elongation or shortening. The robot received actions, positioned itself according to the actions and the tip position was measured. These XYZ coordinates were saved together with the actions for the robot. This process could be called labelling, pairing the inputs with the outputs of the system. Part of this process is captured in Fig. 4.5.

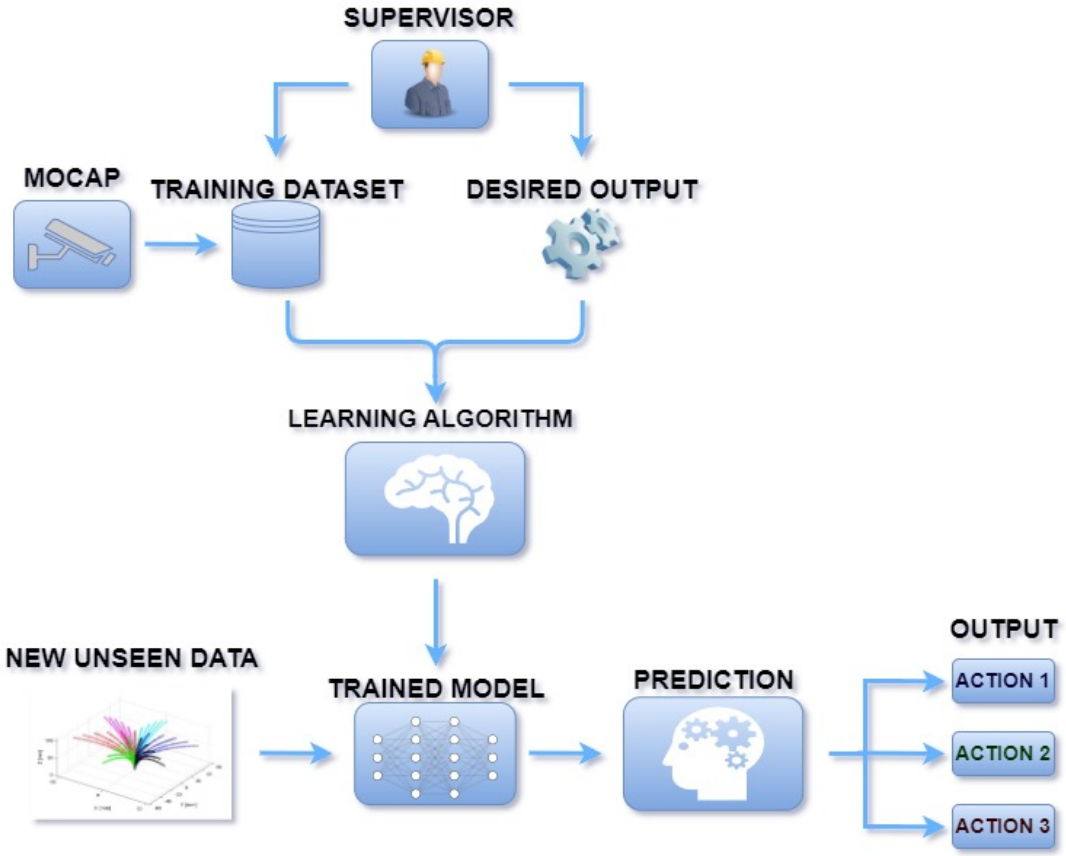


Fig. 4.5 Supervised learning scheme

In this case, the supervisor's role is automated, and data labelling is done as the positions are measured. The dataset has to be divided into training data and testing data. The training data are used for training the model. It will be 29600 data points and the rest will be used for testing. The testing data are data points, that the model has never seen during the training and are new to the model. The proposed model is made of feedforward neural networks containing three hidden layers and an input layer with three neurons and an output layer with three neurons as well. The algorithm used for training the supervised learning model

```
Function main():
    Initialize model and data parameters;
    Set MODEL_NAME to '3x3', input_dim to 3, timesteps to 1, output_dim to 3;
    Load and shuffle input data from 'dataset';
    Split data into training and validation sets;
    // Define model building function
    Function build_model(hp):
        Build a Keras Sequential model with hyperparameter tuning;
        Compile model with Adam optimizer and MSE loss;
        return Compiled model;
    // Configure and run hyperparameter tuner
    Create RandomSearch tuner to optimize model hyperparameters;
    Set tuner objective to minimize loss, max trials to 10;
    Perform hyperparameter search using training and validation data;
    Get best hyperparameters found by the tuner;
    // Build and train final model
    Build model using best hyperparameters;
    Train model on training data for 2000 epochs;
    // Evaluate and save model
    Evaluate model on validation data;
    Display validation loss and accuracy;
    Output model summary;
    Save trained model;
main();
```

Algorithm 5: Supervised learning algorithm

can be seen in algorithm 5. The initial data from the dataset are shuffled to randomise the data. These are then split to training data and validation data, one used only for training and one only for the validation of the model. It can be seen that to to properly tune the hyperparameters of the model, the keras tuner library was used. This library iterates over multiple hyperparameters to find the best one. These are then used to train the model.

4.4 Chapter summary

In this chapter two different machine learning approaches to the control of the continuum robot were presented. The first approach is using reinforcement learning method called Deep Q learning to train a single agent to position the robot in the desired position. It is trained on constant curvature approximation kinematic model of continuum robot. During the training of an agent the agent has a set number of actions that the agent can take and it has to choose the correct action to get the robot to the desired positon. Based on the the current distance to that target, the reward, which is a feedback signal for the agent, is calculated. The reward is also influenced by the distance in the previous step, whether the new step moves the robot closer to the target or not. The training is performed for 1500 episodes and each episode has at most 400 steps. In the second approach a supervised learning approach was proposed to train a regression type of model, predicting actions to achieve the desired target position with the robot. The model is based on feed forward neural networks and it is using data captured from real robot. The captured data consists of 30 000 generated actions for the motors and the achieved position of the tip of the robot after applying the action. These actions and positions are matched in the dataset and used for training the model. The model is trained by matching the input(actions) to the output(XYZ coordinates) and learning from this dataset it is able to predict an action based on the XYZ coordinate.

Chapter 5

Experiments

In this chapter, we will focus on setting up the experiments carried out in this thesis. We will explain the methodology of each experiment and reasons why these experiments are important. An important part of the experiments will be the measurement system, that was used to measure the positional data of the robot.

5.1 Experimental setup

Except the testing bench that is described in section 3, measurement of the positional data is necessary to perform the experiments. For positional data, a motion capture system(mocap) was used to obtain the positional information about the robot that is needed. For the

Fig. 5.1 Motion capture system used for experiments

experiments six mocap cameras were available with Motive software managing the cameras.

The measurement with a mocap system is based on the IR light that is produced by the cameras and reflected back by the reflective markers to the camera sensors. By using multiple cameras, the system is able to triangulate the positions of markers and determine the XYZ coordinates. Before an experiment a calibration process is performed to determine the coordinate system and [0,0,0] point with respect to the robot. In Fig. 5.2, the markers

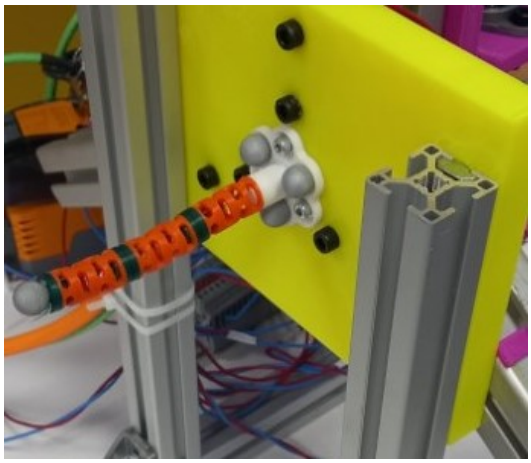


Fig. 5.2 Reflective markers attached to the robot and testing bench

mounted on the robot can be seen. The three markers at the base of the robot were used to determine the position of the base in the coordinate system. The marker at the tip of the robot is then used to measure the position of the tip after an action is applied to the robot and the tip has reached its final position. The accuracy of such mocap system is sub mm depending on the calibration before the measurement. The average accuracy after calibration is in the range between 0.1 mm to 0.5 mm. The calibration process is a manufacturer predefined process that requires data collection and calculations of the accuracy is performed solely by the given software and cannot be altered by the user. To get to the positional data of the markers, it is possible to stream the data to Matlab or ROS. Here, the data can be further used or saved. Since the serial communication of the PC with the robot was already setup in Matlab, the data during the experiment were also streamed there and used or saved to the text file for further use.

5.2 Experimental identifications of repeatability of continuum robot structures movement experiment

To determine which kinematic structure of the three presented ones is the most reliable, an investigation of various movements was performed and the structures were compared. One

of the key properties of continuum robot structure is the repeatability of the movement. The design of a kinematic structure can greatly influence this property. The repeatability is an important factor when applying control, since the same input would have different results on the position of the robot in the workspace. Depending on the various design features like

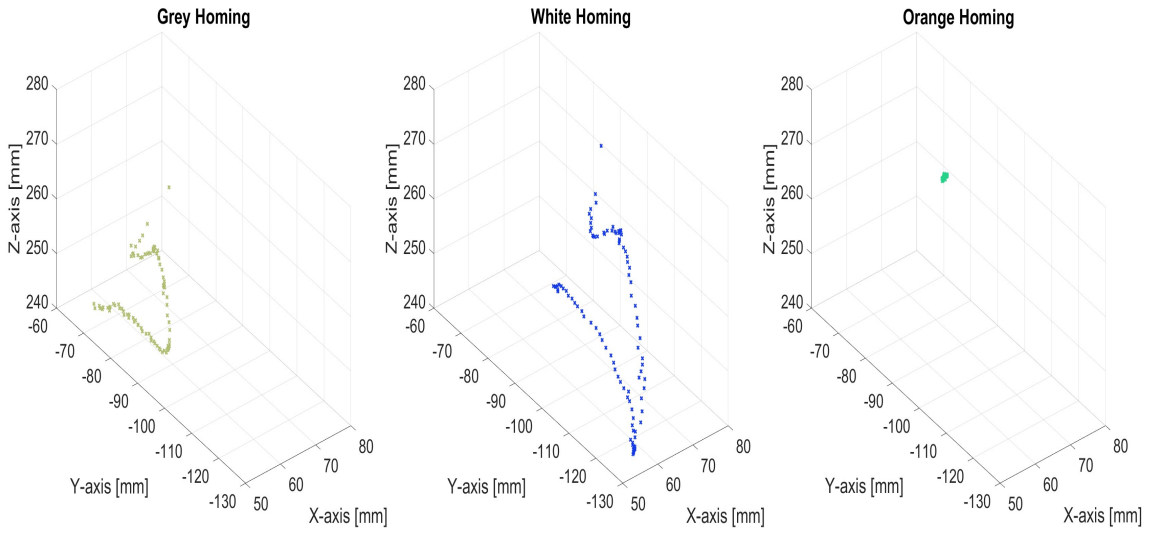


Fig. 5.3 Homing movement

number of segments, number of cables, type of connections between the segment and the rigidity of the segments the actuation can become more difficult and the movement can be difficult to predict and describe. Therefore, this first experiment focuses on measuring the achieved home positions after performing 100 different actions. In this case, the robot started in the straight home position, which is parallel to the ground and moved to a new position. After the robot has moved to a new position, it was going back to its initial position. This initial position was then measured, what can be seen in Fig. 5.3. The graph on the very left of the figure represents the home positions of the second structure from chapter 3, the centre graph represents the position of the third structure and the graph on the right represent the first structure from the chapter 3. All of these graphs show only the tip of the robot in the measured home position after performing an action. In the second part of this experiment, the six markers were placed along the length of the robot, evenly spaced to track the shape of the kinematic structure when being in the home position. The number of markers was determined by trial and error, since more markers would have to be closer to each other. In the cases when the markers were closer than the six markers that were evenly spaced, the mocap system was unable to distinguish the markers from each other. Due to these problems, reflective tape was used instead to mimic the markers and reflect the IR light. The application

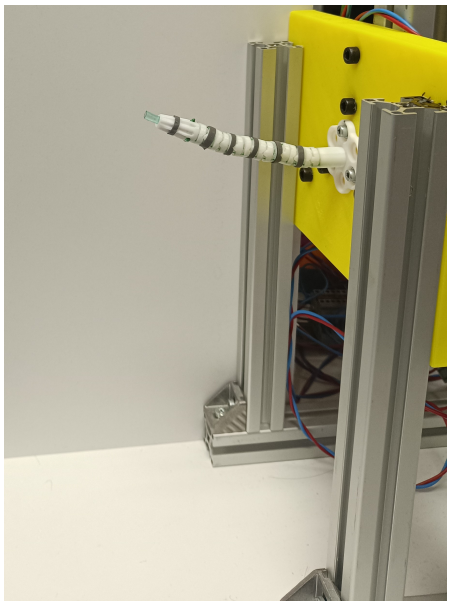


Fig. 5.4 Second part of the experiment using evenly spaced reflective markers

of the reflective tape can be seen in Fig. 5.4. The tape allowed to place more reflective points on the kinematic structure and better copying the shape of the robot. In this part of the experiment, each structure was commanded to go in four opposite directions: up, down, left, right. After performing these actions, the robot moved back to the initial home position and the shape was measured in this home position. The results of these experiments will be presented in the following chapter 6.

5.3 Dataset collection experiment

Another experiment necessary for the supervised learning algorithm is the data collection from the real hardware. The use of such data has many advantages compared to artificial data generated from the model. Such a model is always an approximation of some parts of the real hardware to achieve lower computational costs. These approximations are therefore neglected and are not taken into account when learning-based algorithms are using these artificial data. The data from real hardware can therefore better capture the real state of the system and take into account all the imperfections, material properties and forces acting during the movement of the robot. For this experiment the first continuum robot structure from chapter 3, was selected. Based on the results of experiment from section 5.2 that can be seen in chapter 6. The selected kinematic structure was mounted on the testing bench and a marker was placed at the tip of the continuum robot. The task of this experiment is to generate actions that would be able to cover as much of the robot's workspace as possible.

For this experiment both Matlab script and mocap software is used. The main script is the

```
Calculate all possible actions that the robot can take
Initialise communication with mocap software
Initialise serial communication with the PLC
for i = 1,Number of all actions do
    Cable1 = Action(i, 1)
    Cable2 = Action(i, 2)
    Cable3 = Action(i, 3)
    Robot is home
    if the robot is home then
        The robot is not home
        Send actions to the robot
        if The robot has arrived to the position then
            Trigger the mocap system
            Measure XYZ coordinates
            Save to the dataset [Cable1, Cable2, Cable3, X, Y, Z]
            Move the robot to the home position
        end
    end
end
```

Algorithm 6: Data collection algorithm in Matlab

one in the Matlab. It can be seen in algorithm 6. The Matlab script is connected with the PLC of the robot as well as with the mocap software. Initially in this script the calculation of all possible actions is performed. Here, all the combinations of shortenings of the cables are calculated. Next, the communication with the mocap software is initialised and the serial communication with the robot's PLC is initialised as well. If the communications are configured, the algorithm iterates through all the actions one by one. Initially, the action for each cable is loaded and the initial home position is set. If the robot is in the home position the script sends the action to the PLC and waits until the robot arrives to its final position. When the robot reaches its final position, the mocap software is triggered and the XYZ coordinates are measured. These coordinates of the tip of the robot are streamed to the Matlab script. The coordinates together with the actions are then saved and the robot is commanded to go to its home position. The whole process is repeated for all the actions once the robot is homed again. The results of this experiment are discussed in the following chapter.

5.4 Reinforcement learning algorithm experiment

To test the reinforcement learning algorithm for the control of continuum robot that was described in chapter 4.2, the experiment of following the predefined trajectory is suitable for such case. It was decided to generate a circular trajectory within the robots' workspace that the robot will follow. Since the algorithm tested was trained on the mathematical model of continuum robot, it was opted to generate the trajectory using this model. For the experiment the actuation of the cables was designed such that either one or two cables at the same time are being shortened and the other ones are kept stationary. By using this actuation method, the stationary cable or cables would have higher tension while actuating the other cable or cables. Then this tension on the way back to the initial position would create a force,

```
Initialise variables
Max displacement= 10
Step= 0.5
CirclePlane=8
for i = 1,Step, Max displacement do
    for u = 1,Step, Max displacement do
        for p = 1,Step, Max displacement do
            WorkspaceCoordinates = MatModel(i,u,p)
            if The the distance of the circle plane from the top == CirclePlane then
                Save the last point
            end

        end
    end
end
Calculate the centre point C0 of the saved points
Calculate the radius R of circle with centre C0 going through the saved points
Calculate 100 points creating circular trajectory with centre C0 and radius R
Save the circular trajectory
```

Algorithm 7: Circular trajectory generation for RL algorithm

which helps the kinematic structure get into the initial position. In the following Matlab algorithm, the trajectory generation for the experiment is described. First, the variables used in this script are initialised and maximal displacement, step and circle plane is defined. The maximal displacement is the maximum displacement of the cables of the robot. These were set to be 10 mm at the limits of its range. The step is the change of the displacement, which is set to be at 0.5 mm increments. Next, the combinations of the displacement are generated, using three for loops. The displacement are sent to the mathematical model presented in the

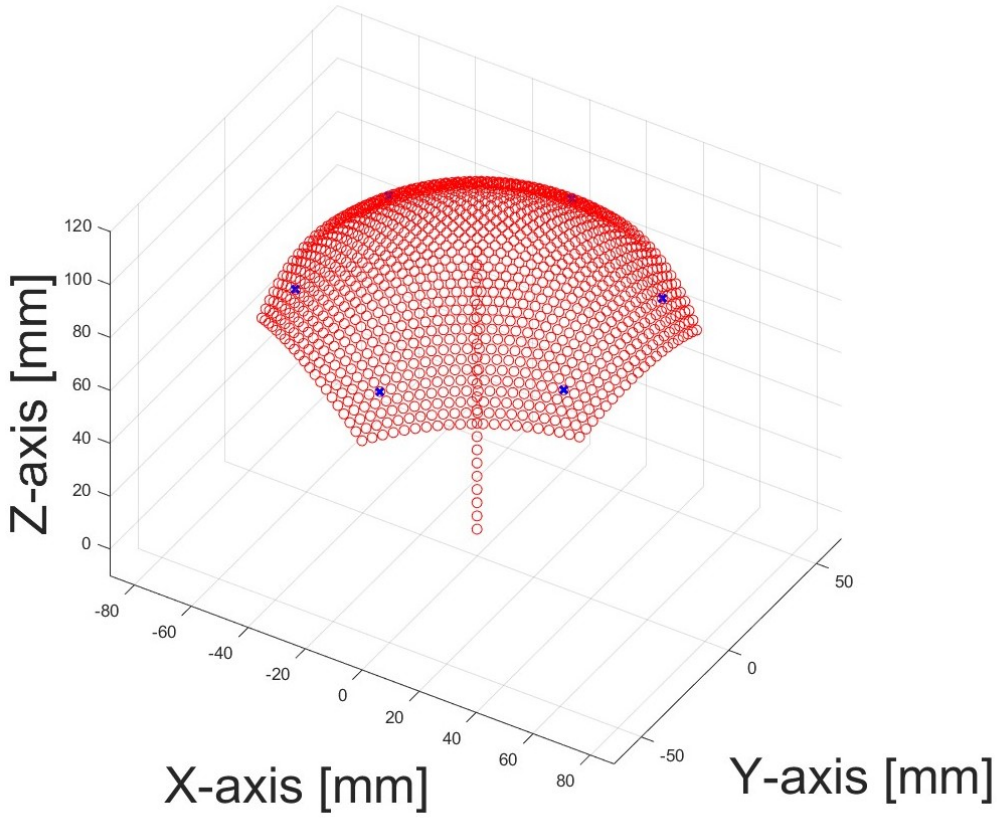


Fig. 5.5 Trajectory generated within the mathematical model of continuum robot

chapter 4, section 4.1.1, and the coordinates are received from the model and saved. Out of these coordinates, the last point at the 8 mm mark displacement is saved. This is done for every combination of cable displacements, which creates a set of points in one plane. These points are place on the borders of six segments of the workspace, which represent six combinations of cable displacement used for this experiment. This can be seen in Fig. 5.5. Using these points, a centre point is calculated as well as the radius of the circular trajectory. Using the centre point and the radius, the circular trajectory is calculated and the trajectory is sampled into 100 points. For each point, the XYZ coordinates are saved for further use. The task of this experiment is for the robot to follow this circular trajectory. For that, these 100 points are sent to the RL controller, and the actions of the controller are used for actuation. In this experiment the action generated from the RL algorithm is going to be tested on the mathematical model in simulation to test the achieved points during the navigation of the algorithm. These points should closely follow the circular trajectory. The algorithm used for the experiment can be seen in algorithm 8. The circle trajectory is loaded, and the algorithm

Initialise the communication with python
Load the circular trajectory coordinates
for *i = 1,Number of points in the trajectory* **do**
    Robot is home
    Action = RLModel($X_c(i)mY_c(i), Z_c(i)$)
    Cable1 = Action($i, 1$)
    Cable2 = Action($i, 2$)
    Cable3 = Action($i, 3$)
    [X, Y, Z] = MathematicalModel(Cable1, Cable2, Cable3)
    Save to the dataset [$Cable1, Cable2, Cable3, X, Y, Z, X_h, Y_h, Z_h$]
    Move the robot to the home position
end

Algorithm 8: Validation of RL control algorithm in Matlab

iterates through all points in the trajectory. These are sent one by one to the RL algorithm, and actions for the cables are received. These are applied to the mathematical model, and the XYZ coordinates are saved. The results of these experiments are discussed in chapter 6.

5.5 Supervised learning algorithm experiment

The last experiment described in this chapter is the experiment that tests the supervised learning algorithm from chapter 4. The experiment in this section focuses on testing the capability of the supervised learning algorithm to follow a predefined trajectory. The supervise learning algorithm, as mentioned in chapter 4, is trained on gathered data from a real robot. To define a trajectory within the workspace of this robot, the gathered data have to be investigated and used to generate the trajectory. The trajectory that the controller is going to follow is also going to be a circular trajectory, since this trajectory fits to the workspace quite well and can test the controller by placing the robot in different segments or parts of its workspace. In Fig. 5.6, the workspace made of real measured points can be seen. In this workspace, the circular trajectory is defined using a similar approach as in the experiment 5.4. First, the workspace is generated and a height of the plane in which the circle is defined. In this height, four points with maximal and minimal Z and X coordinates are identified. These point are then used to calculate the centre point among them, which defines the centre of the circular trajectory. Then, the radius is calculated and the circle is defined intersecting the four identified points earlier. The final circular trajectory is again split to 100 points that define the trajectory and saved for the experiment with the robot. This saved trajectory is then tested on the real robot using the supervised learning algorithm. The experiment algorithm can be seen in the algorithm 9. This algorithm coltrols the whole experiment. First of all

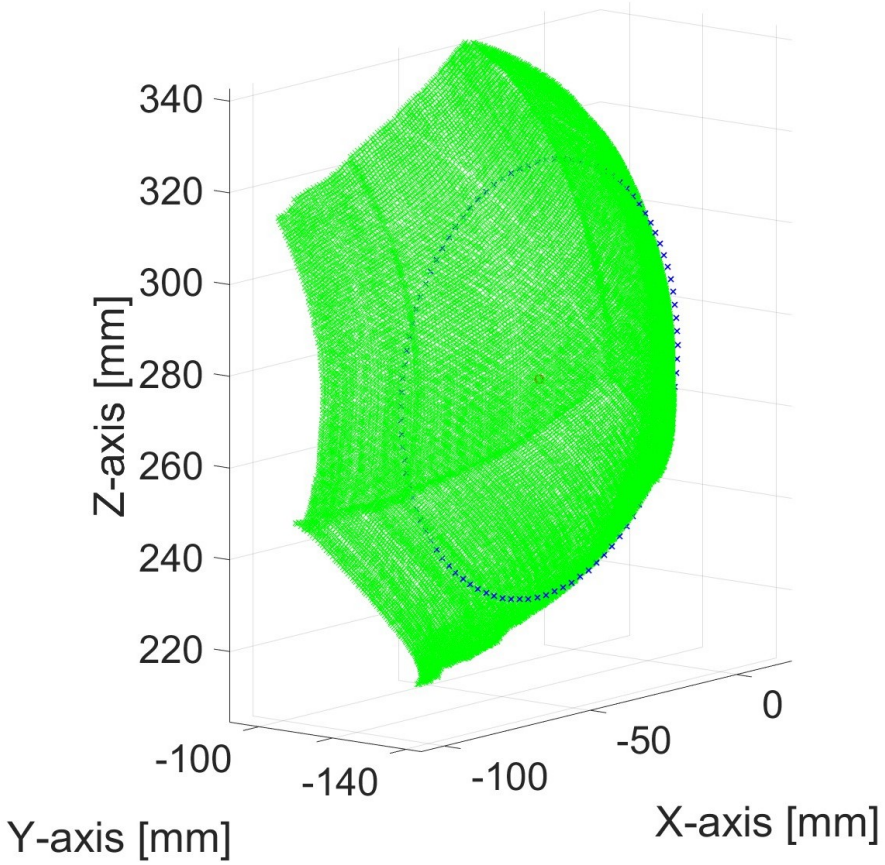


Fig. 5.6 Trajectory generated within the real workspace of the continuum robot

the communication is initialised with the mocap system, PLC and python script which is using the supervised learning model for predicting the actions. Next, the circular trajectory is loaded and the algorithm iterates through all the points in the circular trajectory. At every iteration the python script is called and a XYZ coordinate from the circular trajectory is sent to the model. The model then predicts the action and sends it to the Matlab script. Here, the action is sent to the robot's PLC, but first the robot is in its initial homing position and this is also measured with the mocap system. After the measurement of the home position, the actions are performed by the robot and the tip is measured in its target position. This is saved along with the actions and the home position to the data set. After this is done the robot goes back to its home position and the process starts again with another trajectory point until all 100 points are used and the positional data are saved. The result of this experiment will be further discussed in chapter 6.

```
Initialise the communication with python
Initialise communication with mocap software
Initialise serial communication with the PLC
Load the circular trajectory coordinates
for i = 1,Number of points in the trajectory do
    Action = SupLModel(X_c(i)mY_c(i), Z_c(i))
    Cable1 = Action(i, 1)
    Cable2 = Action(i, 2)
    Cable3 = Action(i, 3)
    Robot is home
    if the robot is home then
        Trigger the mocap system
        Measure XYZ home coordinates
        The robot is not home
        Send actions to the robot
        if The robot has arrived to the position then
            Trigger the mocap system
            Measure [XYZ] coordinates
            Save to the dataset [Cable1, Cable2, Cable3, X, Y, Z, X_h, Y_h, Z_h]
            Move the robot to the home position
        end
    end
end
```

Algorithm 9: Circular trajectory data collection algorithm in Matlab

5.6 Chapter summary

The focus of this chapter is in the experiment performed with the testing bench and the proposed algorithms for controlling the continuum robot. For the positional measurement of the movement of the continuum robot the mocap system using six cameras. This system is using the IR light, which is reflected from the reflective markers, to determine the position of these markers. The system is able to determine the position of the markers with sub-mm accuracy. By attaching these markers to the robot it is possible to track the robot's movements and positions in the 3D space. This was precisely used for the experimentation with the kinematic structures and proposed control algorithms. In this chapter four experiments were described. In the first experiment the repeatability of the proposed kinematic structures is studied. Each of the structures was positioned into 100 different position and was commanded to go back to its initial, home position. The home position was the first initial straightened

position of the robot and it was taken as the ground truth during this experiment. In addition, the shape of the robot was also measured when coming back to the home position from four different directions up, down, left and right. In the second experiment, the focus was on the data collection of the whole workspace. For that the orange flexible kinematic structure was selected, which seemed to be the most promising one. Then, all possible combinations of actions were calculated with a displacement increment of 0.1 mm. All together more then 30000 combination were found and sent to the robot one by one. For the each achieved position the tip of the robot was measured and saved. In the third and fourth experiment, the reinforcement learning algorithm and supervised learning algorithm from chapter 4, was tested by following a circular trajectory. In each of the sections of this chapter, it is explained, how the trajectory was generated and then tested. Result from all the described experiments are discussed in the following chapter 6.

Chapter 6

Results

In the previous chapter, four different experiments are explained and the methods used in these experiments were presented. In this chapter, the results of these experiments are presented and explained. In the sections below, the measured results can be seen. For the positional measurements the mocap system was used and calibrated before every experiment. Each of the experiments is performed using the testing bench from Chapter 3 and one or all kinematic structures.

6.1 Experiment 1 - Repeability and homing of kinematic structures

The task of the first experiment was to test the repeatability of the three kinematic structures that were shown in the chapter 3. For the first part of this experiment, each of the structures was tasked to navigate to 100 different positions and then travel back to the home position, which is a straight position parallel to the ground. This home position was then measured with the mocap system and the measured points were validated and compared to the initial home position. Before this experiment, the calibration process was done according to the mocap software. The achieved calibration error calculated by the software was 0.245 mm.

In each of the structures, the first initial home position was measured and considered as ground truth compared to all the other home positions. Based on this, the results of homing first kinematic structure can be seen in Fig. 6.1. The structure used is the first orange flexible structure. On left side of this figure the graph of actual measured homing points can be seen. There, it can be observed that the spread of the points is quite small and all of the points are contained in a cube roughly with the dimensions of 1x1x1 mm. In the second part of this figure, a graph of errors can be seen. The errors in this case are the Euclidean distances of

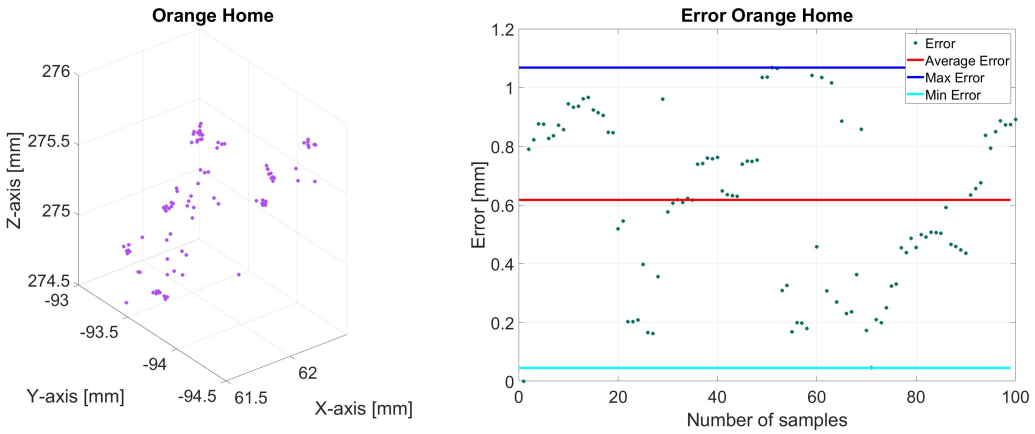


Fig. 6.1 Homing of structure 1

the homing points to the initial homing point considered as ground truth. The results show that the average error is 0.62 mm considering all 100 points. The maximal measured error is 1.1 mm and the minimal error is 0.05 mm.

Using the second kinematic structure from chapter 3, the same experiment was repeated. Unlike the previous kinematic structure, this one is composed of fully rigid sub-sections and the only flexible part is the central backbone made of a flexible tube. The results of this experiment are shown in Fig. 6.2. Again on the left side of this figure the actual measured

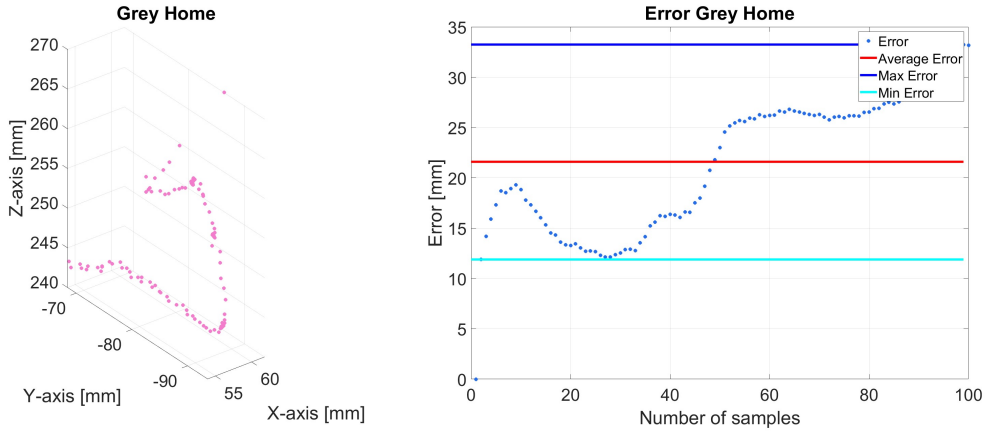


Fig. 6.2 Homing of structure 2

homing positions can be seen. From this graph, it is apparent that the spread of the homing points is quite wide and inconsistent. Unlike in the previous case, here the points cannot be contained in the cube 1x1x1 mm but much larger roughly 8x25x30 mm. On the right side the measured errors or Euclidean distances from the ground truth are shown. The average error

for this structure is 21.6 mm. The maximal measured error is 33.2 mm and the minimal error is 11.9 mm. It is therefore apparent that this structure did much worse than the first one.

In the third measurement, the experiment was repeated with the third structure from chapter 3. This kinematic structure is also rigid, but unlike the previous one, this one has interlocking grooves in between the sub-segments that create this robot. Same as before, on

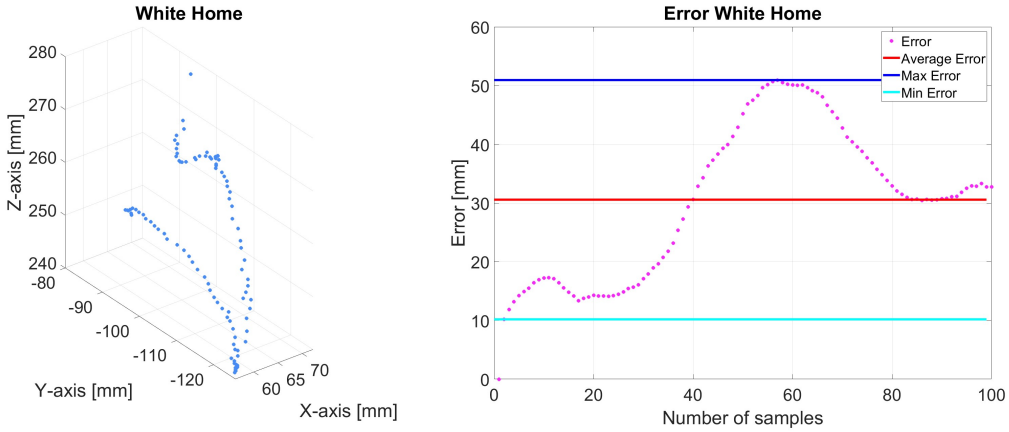


Fig. 6.3 Homing of structure 3

the left side of the Fig. 6.3, the actual measured homing positions are displayed. In this case, it can be deduced from this figure that this structure has even worse results. It is apparent that the homing points in this graph are even more spread than in the figure before. The approximate cube in which these points would fit is 15x50x40 mm, which is again much worse. On the right side of this figure, the graph with the errors is shown. The average error is 30.57 mm, and the maximal distance of the point to the ground truth is 50.94 mm and minimal error 10.19 mm. Based on these results it can be determined that the in interlocking grooves might actually worsen the performance of this kinematic structure compared to the one before, without these grooves.

The second part of this experiment is also tracking the homing position, however, this time the robot was commanded to go to four positions: up, down, left, right. When the robot returned to the home position, not only was the tip of the structure measured, but also six points along the length of the structure. In this experiment, the investigation of the shape of the kinematic structure in the home position is investigated. In Fig. 6.4, the shapes of the kinematic structures can be seen in the home position after coming back from doing one of the four movements. The red color denotes the shape of the first orange kinematic structure, the green color denotes the second grey kinematic structure and the blue color denotes the white structure with interlocking mechanism. The black line in the centre is a straight line considered as a ground truth in comparison with the kinematic structure. It can be observed

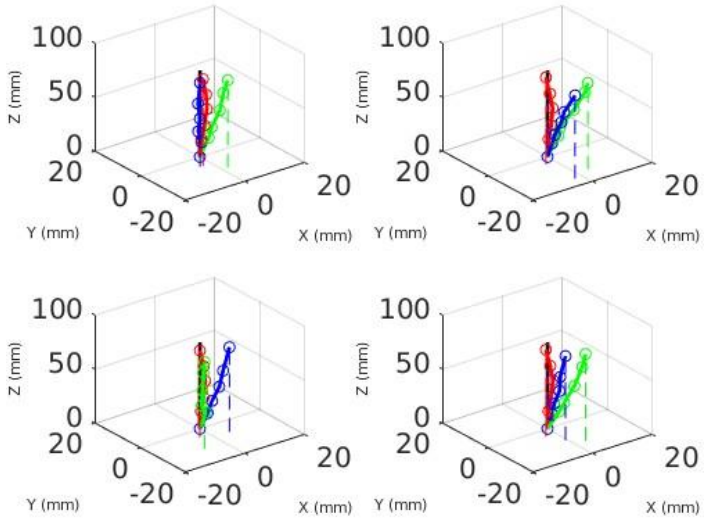


Fig. 6.4 Shapes in the home position

that the red line is the closest one to the ground truth in all of the measurements, just as in the previous part of this experiment. On average, the red line is 5.3 mm away from the ground truth, while other two are 12.3 and 10.0 mm away. These results can be seen in table 6.1, showing the error after coming to the home position after doing one of the movements.

Table 6.1 Shape measurements during homing

Motion	CR1(Orange) [mm]	CR2(Grey) [mm]	CR3(White) [mm]
Up	5.8915	10.1958	6.7348
Down	4.8787	13.4241	16.0505
Left	5.5428	12.8526	8.2937
Right	5.0262	12.7513	9.0044
Average	5.3348	12.3060	10.0209

In the first experiment, the homing of the three different kinematic structures of continuum robots was tested. When the results of the three kinematic structures, it can be said that the repeatability of the first, flexible structure achieved the best results. It is apparent that this structure has the lowest error and is closest to the initial home position. These results were also confirmed in the second part of the experiment, where the shape was measured and was also closest to the initial position. Using these results, the following experiments and testing would be done using the first flexible kinematic structure. By using this structure, the results should be more repeatable and stable.

6.2 Experiment 2 - Dataset collection experiment results

In the second experiment, the collection of tip positions was done to investigate the actual workspace of the real robot. To collect the data, the first kinematic structure was chosen, due to the best results in the first experiment out of the three structures. For this experiment only

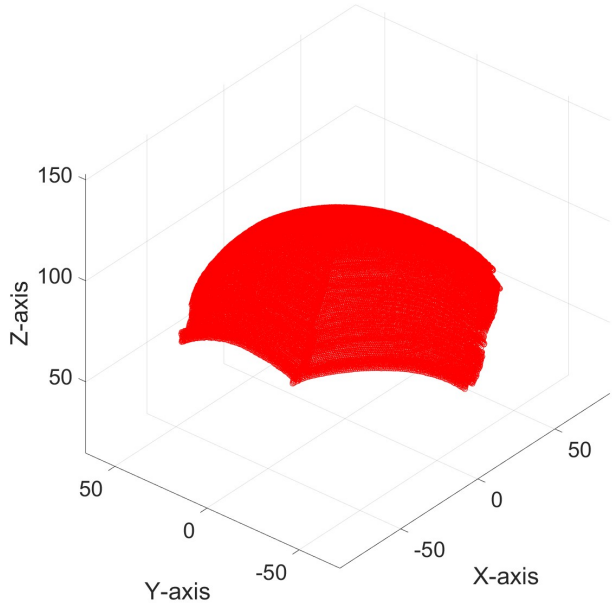


Fig. 6.5 Collected sample of the achieved positions with the tip of the robot.

one marker at the tip of the robot's structure was placed and the achieved positions were measured with the mocap system. The algorithm presented in Section 5.3 shows how the data were collected. In the Fig. 6.5, the collected data can be seen. For collecting this 30 000 position the same amount of combinations of cable displacement were generated and sent to the robot. The cable displacement ranges from 0 to 10 mm for each cable. The step for each change was set to be 0.1 mm. From Fig. 6.5 it can be observed that these steps created a mesh of the tip points that is quite dense. The workspace spans in a square roughly 100 by 100 mm and is comparable in size with the mathematical model from section 4.1. This can be seen in Fig. 6.6, where the red color denotes the measured points and green color denotes the mathematical model using the same amount of samples. To align the coordinate systems, the highest points in the Z axis were found in both sets of samples and aligned so that the results could be compared. To compare the measured points and the mathematical model, we calculated the Euclidean distances of the two closest points between the model and the measured points. Based on this, the graph in the following Fig. 6.7 was created. It

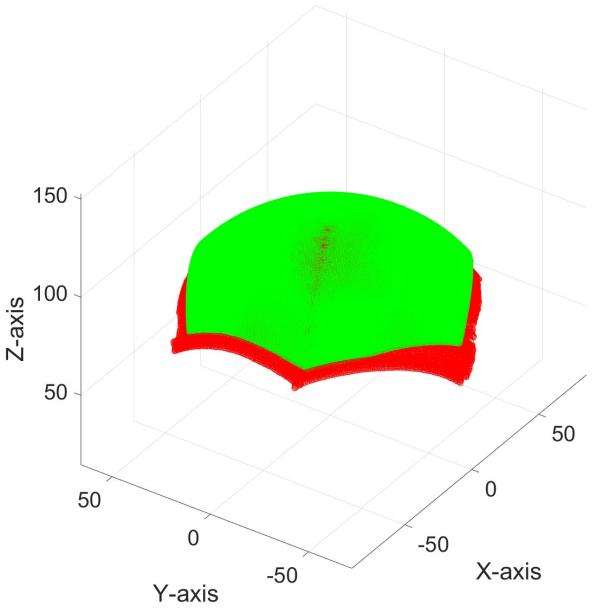


Fig. 6.6 Collected sample vs mathematical model with same amount of samples.

can be observed that the average error was 2.93 mm and the maximal 7.47 mm. Using this knowledge, one can deduce how close/far the model is from the actual robot. The average error of 2.93 mm is a good result, so that the robot can be tested with the proposed control algorithms.

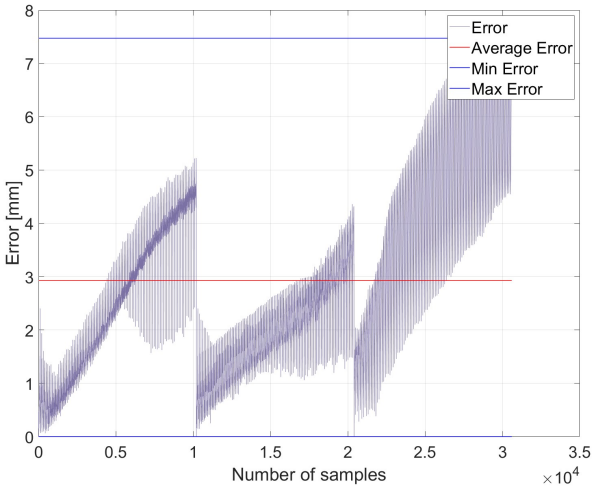


Fig. 6.7 Difference between the collected samples vs mathematical model.

6.3 Experiment 3 - Reinforcement learning algorithm trajectory following

In the previous experiment it was found that the mathematical model is relatively close to the real robot, by comparing their workspaces. In this experiment, the focus was on the testing and verification of DQN based control algorithm. This algorithm was trained on the mathematical model and therefore the first verification and this experiment will be done on this model as well. The controller was tasked to follow the circular trajectory and predict the

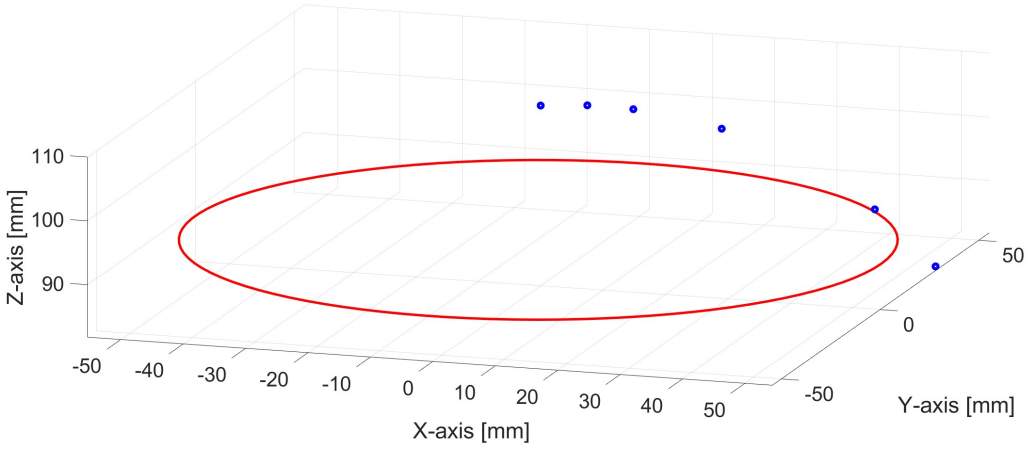


Fig. 6.8 Test of DQN based controller from chapter 4.

actions necessary for the robot's tip to get to the coordinates of the circular trajectory. The way in which the trajectory was sampled is described in section 5.4, where it was described that the trajectory was sampled into 100 points. These points were then sent to the control algorithm, and the algorithm predicted the necessary actions. The achieved positions are shown in the graph in Fig. 6.8. In this figure, it is visible that the controller is not working correctly. For all 100 points, it was only able to predict 6 different positions multiple times. These positions seems to be gradually getting closer to the circular trajectory, however, only in one direction. This can also be observed in Fig. 6.9, where graph with the errors is shown. The calculation of the error was done in a way where achived point is compared with the corresponding point on the circular trajectory, where the robot was supposed to be. It can be seen here that the average error was calculated to be 54.58 mm, the maximal error 88.50 mm, and the minimal error 30.87 mm. It is therefore evident that the trained DQN controller was unable to learn properly, even though during training the accuracy and distance to the target

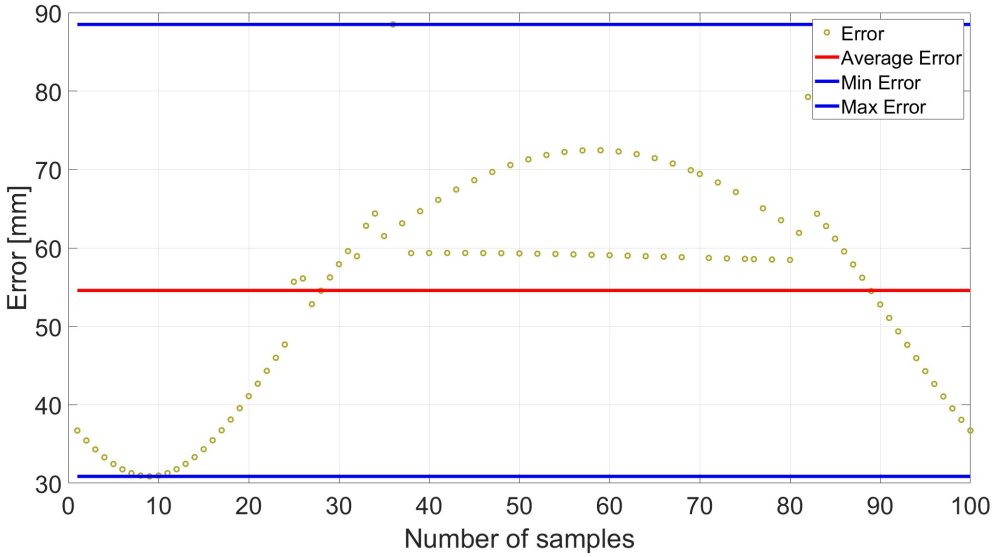


Fig. 6.9 Calculated error between the achieved trajectory and target trajectory.

were satisfactory. This could be due to various factors that influence how the robot learns the policy. One of the factors in this case might be the number of episodes used during training. By increasing the number of episodes, the agent might be able to improve the policy further. Another, even more important factor is the reward function, which acts as a feedback to the agent how the selected action performed and influences the learned policy. Currently, the reward function gives information about the current distance to the target point and whether the current distance is smaller or larger than the previous one. By adding information on the direction in which the target is placed, the training performance could be improved. Since the performance of the control algorithm is already unsatisfactory, there is no need to test it on the real robot, where the results would not show any improvements or any other new findings.

6.4 Experiment 4 - Supervised learning algorithm trajectory following

In the last experiment of this chapter, the testing of the supervised learning algorithm will follow the same objectives as in the experiment before. The basics of the experiment will remain the same and the task will be to follow a circular trajectory in the workspace of the robot. Since the supervised learning algorithm was trained on actual data from the real robot's movements, this experiment will test the supervised learning algorithm on the real

robot. The trajectory generation was described in the section 4.3, where the circular trajectory is generated and divided into 100 sample points. These data points were fed to the supervised algorithm and the algorithm predicted the appropriate actions for the robot to take in order to get to the desired point. These were sent to the robot and the achieved positions of the tip of the robot were measured with the mocap cameras. Even though the collected positional data for training of the supervised learning algorithm were only collected from the first structure, the algorithm will be tested on all three structures to see how will other structures respond using this supervised learning model. The experiment started with the first, flexible kinematic structure. The experiment went according to the algorithm 9, where the sample points are

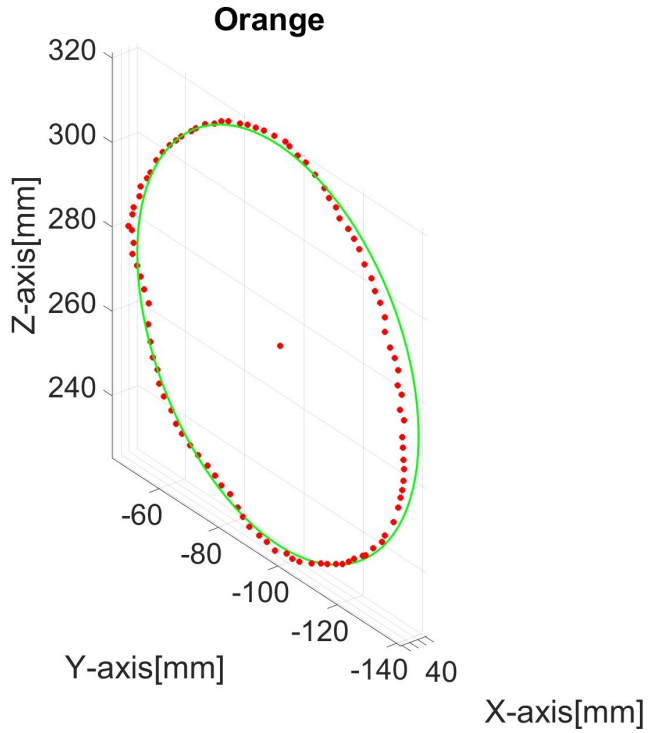


Fig. 6.10 Trajectory followed by first kinematic structure using supervise learning algorithm.

sent to the supervised leaning model, which predicts the actions. These are then sent to the robot's PLC and after the robot achieved the final position, the mocap cameras measured the tip's position. The results of testing the algorithm on the first structure can be seen in Fig. 6.10. Here, the measured points follow the circular trajectory quite closely and full circle trajectory was achieved with this kinematic structure. Observing the trajectory, except for a few parts where the measured points deviated from the circle, the results show that the algorithm was able to predict the actions closely to the target. In the following graph in Fig. 6.11 the error of the actual trajectory from the target trajectory can be seen. The

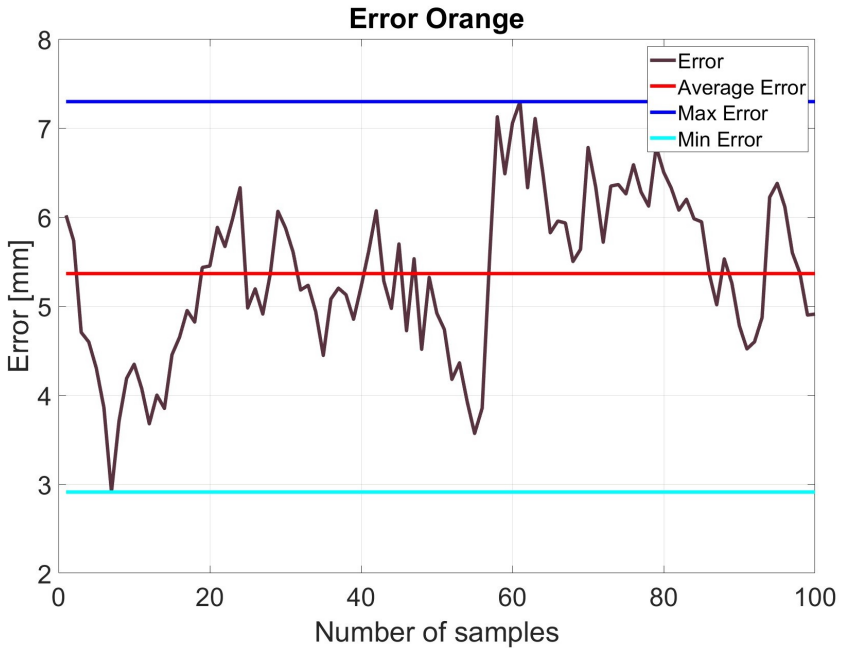


Fig. 6.11 Error of the trajectory followed by first kinematic structure using supervise learning algorithm.

error was calculated as an Euclidean distance between the measured point and the point on the trajectory. The maximal error of 7.3 mm, an average error of 5.37 mm and a minimal error of 2.91 mm were measured. This graph therefore confirms that the deviation from the actual trajectory is not that substantial. If Fig. 6.10 is closely observed, most of the deviation is found on the X axis, compared to the deviation in the Y and Z axes. This can be due to multiple reasons. One being the nature of the workspace that has a semi-spherical shape or some other constraint that influences the movement of the robot. However, the control algorithm using the supervised learning model worked well with this kinematic structure.

The supervised learning algorithm was then tested on the second kinematic structure of chapter 3. Unlike the previous structure, this one is composed of fully rigid sub-segments connected with a flexible tube in the centre. Although the algorithm tested was trained on the data from the flexible kinematic structure, the aim of this testing was to investigate how different kinematic structures react to this control algorithm. The experiment with the second kinematic structure followed the same algorithm as with the first structure. First, the circular trajectory sample points are loaded and one by one sent to the supervised learning algorithm. The algorithm then predicts the actions and sends them to the robot's PLC. When the robot reaches the final position, the coordinates of the tip are measured and saved with the actions. The results of the experiments can be seen in Fig. 6.12. The measured positions can be seen

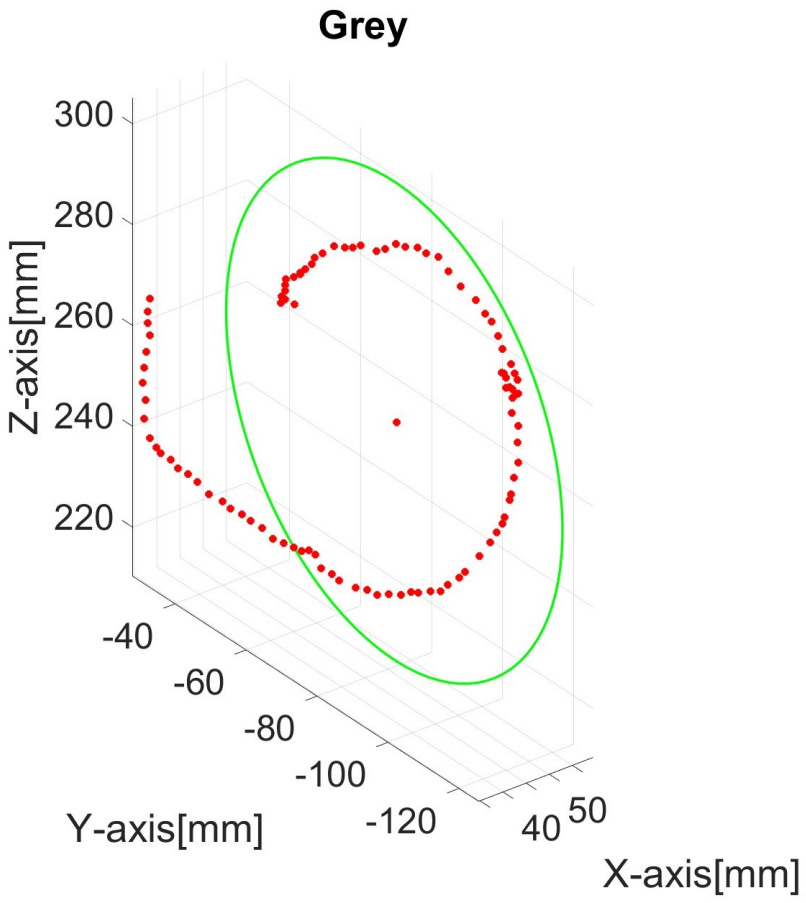


Fig. 6.12 Trajectory followed by second kinematic structure using supervise learning algorithm.

in this graph. It can be observed that the measured trajectory is very loosely following the predefined circular trajectory. It can be seen that the final measured trajectory is not even finishing the full circle and does not reach the initial point that it started with. The significant differences between the target trajectory and the final measured trajectory can be seen this time in all axes. In comparison with the first kinematic structures, this one has done much worse. The errors in this part of the experiment are shown in Fig. 6.13. In this graph, the measured errors are significantly worse than in the first part of this experiment. The average error between the trajectories is 20.15 mm, the minimal error is 14.35 mm, and the maximal error 27.02 mm. From the graph, we can deduce that the trend of the error is increasing towards the end of the measurements. This could be due to multiple reasons, one of them can be the caused by the mechanical design of the structure. Since, the structure is composed of multiple smaller segments on top of each other. These segments move with respect to each other during the movement of the robot. These movements between the sub-segments

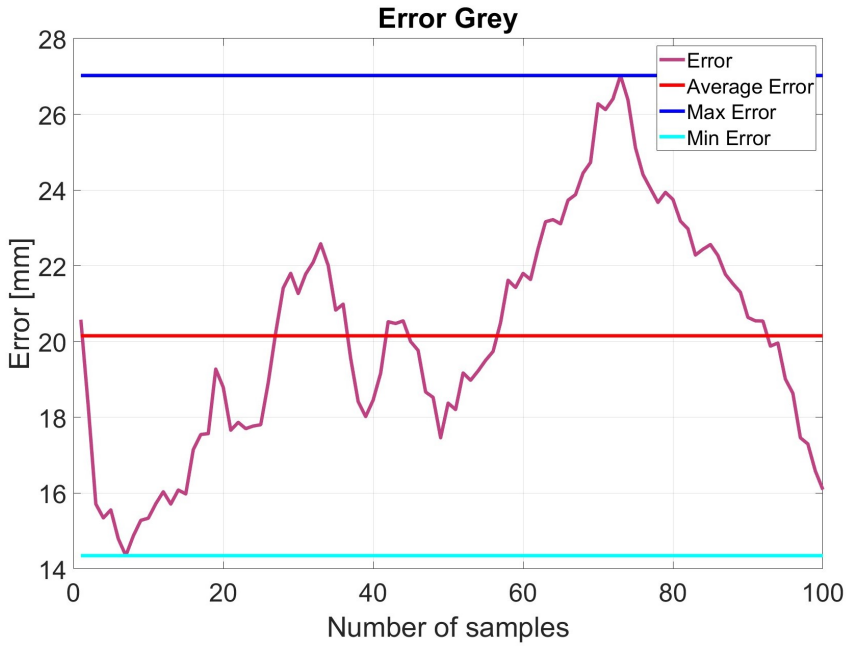


Fig. 6.13 Error of the trajectory followed by second kinematic structure using supervise learning algorithm.

can cause the structure to have different shapes during the movement. This can cause that the more the robot moves, the more error is introduced into the movement of the robot. The results therefore show that the control algorithm is not suitable to be used with this kinematic structure due to large errors in the actual trajectory and target trajectory.

In the last part of this experiment, the supervised learning algorithm was tested on the third kinematic structure from Chapter 3. Similarly as in the previous part of the experiment, this structure is also composed of multiple rigid sub-segments. However, in addition, these sub-segments have an interlocking mechanism in the form of the grooves. These grooves are perpendicular to each other so that the structure can move to all the directions, but have higher rigidity. The experiment task was again to follow the circular trajectory using the supervise learning algorithm trained on the collected dataset from the first kinematic structure. Using the same algorithm as before, The robot starts with the first of 100 sample points of the trajectory, predicts and action using the supervised learning algorithm. The predicted actions are sent to the robot and when the robot reaches it's final position, the tip of the robot is measured with the mocap cameras. The XYZ coordinates are saved with the actions and the algorithm iterates over the rest of the 100 sample points and saves the measured coordinates. The results of the measured trajectory can be seen in Fig. 6.14. The measured trajectory in this graph follows the target circular trajectory better than the second kinematic structure, which was tested, however, worse than the first one. The deviations can be seen in all axes,

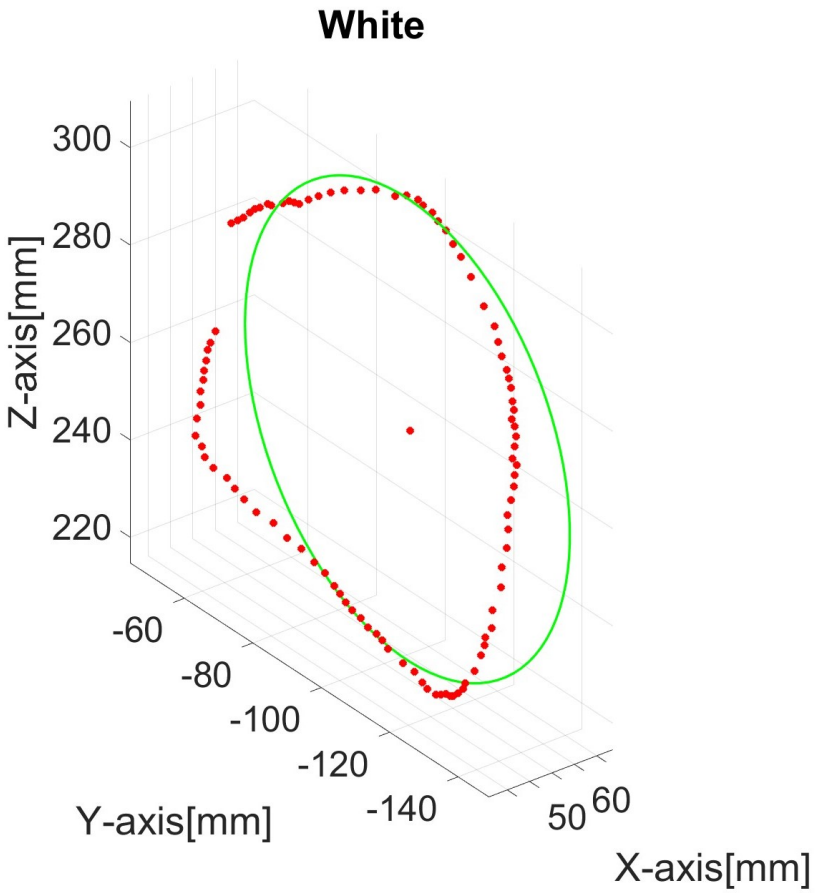


Fig. 6.14 Trajectory followed by third kinematic structure using supervise learning algorithm.

however the most dominant ones can again be seen in the X axis. The actual trajectory again deviated from its original shape and have not finished the full circular shape. In comparison, it is, however, closer to the trajectory than the second tested kinematic structure. The graph in Fig. 6.15 also supports the findings from Fig. 6.14. The graphs show how the error is evolving through out the movement of the robot in the circular trajectory. The data show that the average error is 13.74 mm maximal error 18.96 and minimal error 7.72. The error data show that the third structure performed better using the supervised learning algorithm than the second structure. However, the error is still high, but this can be expected, since the supervised learning model is still trained on the data from the first structure. The first structure is more flexible and therefore has different properties.

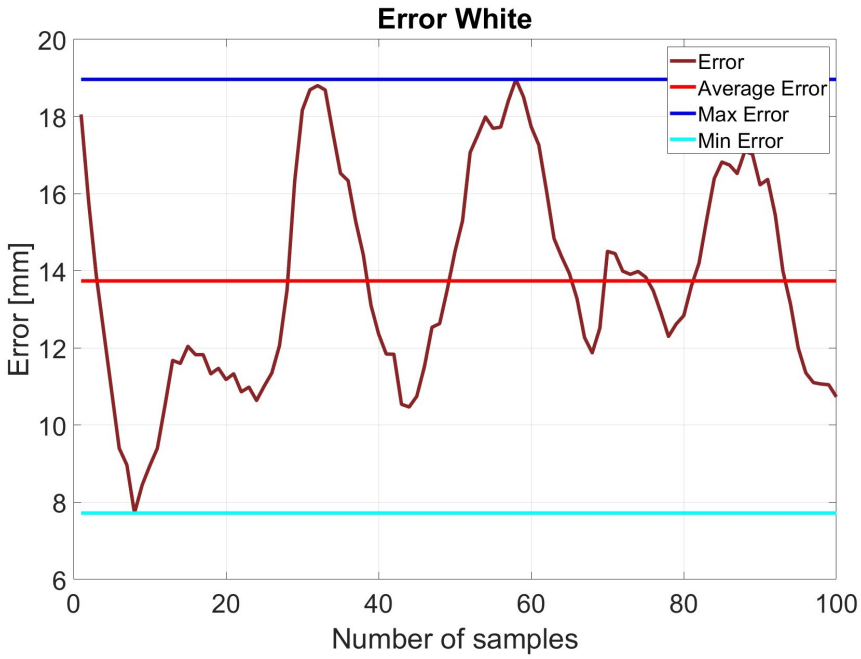


Fig. 6.15 Error of the trajectory followed by third kinematic structure using supervise learning algorithm.

6.5 Chapter summary

The chapter 6 focusses on the presentation of the results of the experiments explained in the chapter 5. For the measurement the mocap system was used to measure the marker's coordinates, which was placed at the tip of the robot. Using this method, it was possible to determine the tip position with sub-millimetre accuracy and directly gather the data of the robots positions. To test the kinematic structures and the proposed control algorithms, four different experiments were carried out. In the first experiment, the repeatability of three kinematic structures from chapter 3 was tested. Each of the structures was commanded to travel to 100 different positions and then return to its initial home position. The ground truth home position was the one in which the robot started with, before the first movement. Based on the results of this experiment it is possible to say that the first kinematic structure showed the best repeability of the three tested. The average error of the first kinematic structure was only 0.62 mm. Compared to the average error of 21.6 mm and 30.57 mm of the other two structures, the first structure is significantly better at coming back to the home position. This also suggests that the repeatability of the movement in general should be much better. In the second part of the first experiment the shape was also watched when returning home after conducting four movements up, down, left and right. These result show similar findings, since the first structure performed the best. In the second experiment , the task was to collect

positional data of the tip positions after performing more than 30 000 movements. To collect these data the best performing kinematic structure from experiment 1 was selected. These data show a good representation of the real workspace of the robot. Comparison of the measured data with the mathematical model from chapter 4, shows the similarity of the two workspaces. When two closest points from each of the workspaces were compared, an average error of 2.93 mm was calculated by comparing all the points. In the third experiment, the testing of reinforcement learning algorithm based on DQN, from chapter 4 was shown. The task of this experiment was to follow the circular trajectory sampled into 100 points, using the DQN controller. The controller was trained using the mathematical model and therefore it was also tested on this model. The performance of this controller during this experiment has shown poor results. The trained agent was only able to predict six different sets of actions, of which most were not even close to the circular trajectory. Due to these reasons the average error was measured to be 54.58 mm. In the last fourth experiment, the task was to test the second proposed control algorithm based on supervised learning from chapter 4. Using the same method for testing as before, the circular trajectory was followed. In this experiment, the testing was done on all three kinematic structures of the robot and tip positions were measured with mocap cameras. As expected the second and third kinematic structure performed much worse. Their average deviation from the trajectory was 20.15 mm and 13.74 mm respectively. This is understandable, since the supervised learning was trained using the data from the first kinematic structure. The first structure, performed much better and the average error was measured to be 5.37 mm. Here it was also observed that the most of the deviation is in X-axis.

Contributions and Conclusion

The aim of this thesis is to enhance the understanding and application of cable-driven continuum robots and their control using methods of machine learning. In this thesis, we designed and used a testing bench for cable-driven continuum robots, with a focus on easy adaptation for a wide variety of cable driven continuum robots. The number of controlled cables can be easily increased or decreased, and the kinematic structures can be quickly changed, as was also shown. The purpose of the test bench was to test different kinematic structures and control algorithms. Two control algorithms using machine learning were proposed. One algorithm was based on the reinforcement learning approach using a deep Q-learning agent to learn from the constant curvature model of the cable-driven continuum robot. The agent's task was to learn how to manage the lengths of the cables controlling the robot based on the actual distance to the target position. A different approach was based on the supervised learning approach for controlling a continuum robot, focussing on predicting cable displacements through regression modeling. Data were collected by generating 30,000 combinations of actions to extensively cover the robot's workspace, starting each action from a standardised initial position. The robot's movements and resulting positions were recorded using a mocap system, creating a labelled dataset. This was then used to train the FFNN model to predict the cable displacements when the target point is the input to the model.

The thesis outlines the study and development of continuum robots and their control algorithms over seven chapters. It begins with an introduction to continuum robots, including their classification, sensor types, and mathematical modelling. Subsequent chapters discuss classification of machine learning techniques, the hardware specifics of continuum robots developed by the research group, and the proposed control algorithms and their implementation. The experimental setup and results are detailed in later chapters, culminating in a summary of findings and performance evaluations of the experiments carried out.

Based on the result obtained from the experiments, multiple areas for future work were identified and these are as follows:

- Reinforcement learning approach needs further improvement to have better results. There are multiple aspects during the learning procces that could improve the performance of the controller. One of the aspects that could be improved is the reward shaping. By improving the reward, the agent should learn more efficiently and have better results. This could be improved by adding information about the target point direction or position in the workspace with respect to the robot.
- Based on the results from comparison of dataset and mathematical model, it could be said that these workspaces are not far from each other. This could be used for further research of learning-based controllers, where the model would act as sort of prior knowledge and less data would be required for further training of the controller.
- Another point that can be further studied is the introduction of feedback from the robot to the learning-based controller for even better performance.
- Based on the results of section 6.3, where the algorithm trained on the dataset of one structure was applied to multiple structures, we can see that the performance was much worse on the other structures, however, the trajectory followed at some points was close to the predefined trajectory. Using this knowledge, we think that by characterising some common features between the structures and introducing them to the learning-based controllers could result in more general models. These could be then used for various kinematic structures.

Summary of the scientific and technological contributions are as follows:

- **The main scientific contribution** of this thesis is the development of two machine learning based controllers. One controller is based on reinforcement learning approach using a DQN agent trained on the mathematical model of continuum robot. The agent is trained to manage displacements of cables controlling the continuum robot when a target position is defined. The training of the agent was mainly influenced by the reward function thaht takes into an account the actual distance to the target and improvement of the new action compared to the old one. Based on the experiment carried out it was shown that the performance of the agent needs further improvements.

  In the second control algorithm, a data-based approach was chosen using supervised learning and real positional data gathered from the robot. The dataset containing 30

000 positional data was collected and labelled. The dataset was then used to train a FFNN for predicting appropriate actions when the target position is known. This approach was tested on real hardware and has shown good performance. It was tested by following a predefined circular trajectory using the test bench designed in this thesis. The best average deviation from the target trajectory was found to be 5.37 mm which is a promising result for such an open-loop controller.

- **The second scientific contribution** of this thesis is in the creation of an open-source dataset using the presented test bench and kinematic structure. This dataset can be found online and available for the community for further research of control algorithms for cable-driven continuum robots. Along with the dataset the CAD files to the test bench and source codes to the presented algorithms can be found.

- **The first technological contribution** is in the development of an open-source test bench for cable driven continuum robots. The design accounts for different designs of continuum robots and number of their actuated cables. The actuation modules can be easily manufactured and added to the bench as needed. The test bench also allows for quick and easy replacement of various kinematic structure that used cables for their actuation.

- **The second technological contribution** of this thesis was in the testing and validation of multiple kinematic structures on the testing bench. These results can be seen in this work and validate the test bench. In addition to validation of the kinematic structures, this thesis showed the validation of various control algorithms using the test bench. The results show, that the first kinematic structure, with flexible features performed the best in terms of repeatability and precision. Together with the dataset and test bench will be available online, for further research and experiments for the community.

The presented algorithms, datasets and CAD files are published at https://github.com/ARM-Lab.

References

[1] Abah, C., Orekhov, A. L., Johnston, G. L., Yin, P., Choset, H., and Simaan, N. (2019). A Multi-modal Sensor Array for Safe Human-Robot Interaction and Mapping. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 3768–3774. ISSN: 2577-087X.

[2] Abidi, H., Gerboni, G., Brancadoro, M., Fras, J., Diodato, A., Cianchetti, M., Wurdemann, H., Althoefer, K., and Menciassi, A. (2018). Highly dexterous 2-module soft robot for intra-organ navigation in minimally invasive surgery. *The international journal of medical robotics + computer assisted surgery: MRCAS*, 14(1).

[3] Ashwin, K. P. and Ghosal, A. (2021). Forward Kinematics of Cable-Driven Continuum Robot Using Optimization Method. In Sen, D., Mohan, S., and Ananthasuresh, G. K., editors, *Mechanism and Machine Science*, Lecture Notes in Mechanical Engineering, pages 391–403, Singapore. Springer.

[4] Baaij, T., Holkenborg, M. K., Stölzle, M., Tuin, D. v. d., Naaktgeboren, J., Babuška, R., and Santina, C. D. (2022). Learning 3D shape proprioception for continuum soft robots with multiple magnetic sensors. *Soft Matter*, 19(1):44–56.

[5] Back, J., Dasgupta, P., Seneviratne, L., Althoefer, K., and Liu, H. (2015). Feasibility study- novel optical soft tactile array sensing for minimally invasive surgery. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1528–1533.

[6] Bruns, T. L., Remirez, A. A., Emerson, M. A., Lathrop, R. A., Mahoney, A. W., Gilbert, H. B., Liu, C. L., Russell, P. T., Labadie, R. F., Weaver, K. D., and Webster, R. J. (2021). A modular, multi-arm concentric tube robot system with application to transnasal surgery for orbital tumors. *The International Journal of Robotics Research*, 40(2-3):521–533.

[7] Burgner-Kahrs, J., Rucker, D. C., and Choset, H. (2015). Continuum Robots for Medical Applications: A Survey. *IEEE Transactions on Robotics*, 31(6):1261–1280.

[8] Centurelli, A., Arleo, L., Rizzo, A., Tolu, S., Laschi, C., and Falotico, E. (2022). Closed-Loop Dynamic Control of a Soft Manipulator Using Deep Reinforcement Learning. *IEEE Robotics and Automation Letters*, 7:1–1.

[9] Chen, Y., Oliveira, J. M., and Hunter, I. W. (2013). Two-axis bend sensor design, kinematics and control for a continuum robotic endoscope. In *2013 IEEE International Conference on Robotics and Automation*, pages 704–710. ISSN: 1050-4729.

[10] Chikhaoui, M. T. and Burgner-Kahrs, J. (2018). Control of Continuum Robots for Medical Applications: State of the Art. In *ACTUATOR 2018; 16th International Conference on New Actuators*, pages 1–11.

[11] Chollet, F. (2017). *Deep Learning with Python*. Manning Publications, second edition.

[12] Cios, K. J., Swiniarski, R. W., Pedrycz, W., and Kurgan, L. A. (2007). Unsupervised Learning: Association Rules. pages 289–306. Springer US, Boston, MA.

[13] Costa, C. F. R. and Reis, J. C. P. (2023). End-Point Position Estimation of a Soft Continuum Manipulator Using Embedded Linear Magnetic Encoders. *Sensors*, 23(3):1647.

[14] Cunningham, P., Cord, M., and Delany, S. J. (2008). Supervised Learning. pages 21–49. Springer, Berlin, Heidelberg.

[15] del Real Torres, A., Andreiana, D. S., Ojeda Roldan, A., Hernandez Bustos, A., and Acevedo Galicia, L. E. (2022). A Review of Deep Reinforcement Learning Approaches for Smart Manufacturing in Industry 4.0 and 5.0 Framework. *Applied Sciences*, 12(23):12377.

[16] Disse, L. (2020). *System Identification for an Elephant-Trunk Like Robotic Arm Spring Term 2018 Declaration of Originality*. PhD thesis.

[17] Dong, X., Axinte, D., Palmer, D., Cobos, S., Raffles, M., Rabani, A., and Kell, J. (2017). Development of a slender continuum robotic system for on-wing inspection/repair of gas turbine engines. *Robotics and Computer-Integrated Manufacturing*, 44:218–229.

[18] Dong, X., Palmer, D., Axinte, D., and Kell, J. (2019). In-situ repair/maintenance with a continuum robotic machine tool in confined space. *Journal of Manufacturing Processes*, 38:313–318.

[19] Dupont, P. E., Simaan, N., Choset, H., and Rucker, C. (2022). Continuum Robots for Medical Interventions. *Proceedings of the IEEE. Institute of Electrical and Electronics Engineers*, 110(7):847–870.

[20] El-Atab, N., Mishra, R. B., Al-Modaf, F., Joharji, L., Alsharif, A. A., Alamoudi, H., Diaz, M., Qaiser, N., and Hussain, M. M. (2020). Soft Actuators for Soft Robotic Applications: A Review. *Advanced Intelligent Systems*, 2(10):2000128.

[21] Falkenhahn, V., Hildebrandt, A., Neumann, R., and Sawodny, O. (2017). Dynamic Control of the Bionic Handling Assistant. *IEEE/ASME Transactions on Mechatronics*, 22(1):6–17.

[22] Fellmann, C. and Burgner-Kahrs, J. (2015). Implications of trajectory generation strategies for tubular continuum robots. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 202–208.

[23] Fras, J., Glowka, J., and Althoefer, K. (2020). Instant soft robot: A simple recipe for quick and easy manufacturing. In *2020 3rd IEEE International Conference on Soft Robotics (RoboSoft)*, pages 482–488.

[24] Galloway, K. C., Chen, Y., Templeton, E., Rife, B., Godage, I. S., and Barth, E. J. (2019). Fiber Optic Shape Sensing for Soft Robotics. *Soft Robotics*, 6(5):671–684.

[25] George Thuruthel, T., Falotico, E., Renda, F., and Laschi, C. (2018). Model-Based Reinforcement Learning for Closed-Loop Dynamic Control of Soft Robotic Manipulators. *IEEE Transactions on Robotics*, PP:1–11.

[26] Gerboni, G., Diodato, A., Ciuti, G., Cianchetti, M., and Menciassi, A. (2017). Feedback Control of Soft Robot Actuators via Commercial Flex Bend Sensors. *IEEE/ASME Transactions on Mechatronics*, 22(4):1881–1888.

[27] Giorelli, M., Renda, F., Ferri, G., and Laschi, C. (2013). A feed-forward neural network learning the inverse kinetics of a soft cable-driven manipulator moving in three-dimensional space. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5033–5039.

[28] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. Google-Books-ID: omivDQAAQBAJ.

[29] Grassmann, R. M., Rao, P., Peyron, Q., and Burgner-Kahrs, J. (2022). Fas—a fully actuated segment for tendon-driven continuum robots. *Frontiers in Robotics and AI*, 9.

[30] Grondman, I., Busoniu, L., Lopes, G. A. D., and Babuska, R. (2012). A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(6):1291–1307.

[31] Guo, H., Ju, F., Cao, Y., Qi, F., Bai, D., Wang, Y., and Chen, B. (2019). Continuum robot shape estimation using permanent magnets and magnetic sensors. *Sensors and Actuators A: Physical*, 285:519–530.

[32] Hainsworth, T., Smith, L., Alexander, S., and MacCurdy, R. (2020). A Fabrication Free, 3D Printed, Multi-Material, Self-Sensing Soft Actuator. *IEEE Robotics and Automation Letters*, 5(3):4118–4125.

[33] Hannan, M. W. and Walker, I. D. (2003). Kinematics and the implementation of an elephant's trunk manipulator and other continuum style robots. *Journal of Robotic Systems*, 20:45–63.

[34] Ikuta, K., Ichikawa, H., Suzuki, K., and Yajima, D. (2006). Multi-degree of freedom hydraulic pressure driven safety active catheter. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 4161–4166. ISSN: 1050-4729.

[35] Jang, B., Kim, M., Harerimana, G., and Kim, J. (2019). Q-Learning Algorithms: A Comprehensive Classification and Applications. *IEEE Access*, PP:1–1.

[36] Jeon, J. and Kim, C. (2021). Shape Sensor Using Magnetic Induction with Frequency Sweeping for Medical Catheters. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7137–7143. ISSN: 2577-087X.

[37] Ji, G., Yan, J., Du, J., Yan, W., Chen, J., Lu, Y., Rojas, J., and Cheng, S. S. (2021). Towards Safe Control of Continuum Manipulator Using Shielded Multiagent Reinforcement Learning. *arXiv:2106.07892 [cs]*. arXiv: 2106.07892.

[38] Jo, T. (2021). *Machine Learning Foundations*. Springer.

[39] Jones, B. and Walker, I. (2006). Kinematics for multisection continuum robots. *IEEE Transactions on Robotics*, 22(1):43–55.

[40] Kato, T., Okumura, I., Song, S.-E., Golby, A. J., and Hata, N. (2015). Tendon-Driven Continuum Robot for Endoscopic Surgery: Preclinical Development and Validation of a Tension Propagation Model. *IEEE/ASME Transactions on Mechatronics*, 20(5):2252–2263.

[41] Katzschmann, R., Della Santina, C., Toshimitsu, Y., Bicchi, A., and Rus, D. (2019). *Dynamic Motion Control of Multi-Segment Soft Robots Using Piecewise Constant Curvature Matched with an Augmented Rigid Body Model*.

[42] Kim, T., Yoon, S. J., and Park, Y.-L. (2018). Soft Inflatable Sensing Modules for Safe and Interactive Robots. *IEEE Robotics and Automation Letters*, 3(4):3216–3223.

[43] Kolachalama, S. and Lakshmanan, S. (2020). Continuum Robots for Manipulation Applications: A Survey. *Journal of Robotics*, 2020:e4187048.

[44] Kubat, M. (2021). *An Introduction to Machine Learning*. Springer Nature. Google-Books-ID: cshEEAAAQBAJ.

[45] Li, J., Zhang, F., Yang, Z., Jiang, Z., Wang, Z., and Liu, H. (2022). Shape Sensing for Continuum Robots by Capturing Passive Tendon Displacements With Image Sensors. *IEEE Robotics and Automation Letters*, 7(2):3130–3137.

[46] Liang, K., Zhang, G., Guo, J., and Li, W. (2023). An Actor-Critic Hierarchical Reinforcement Learning Model for Course Recommendation. *Electronics*, 12(24):4939.

[47] Liu, H., Farvardin, A., Pedram, S. A., Iordachita, I., Taylor, R. H., and Armand, M. (2015). Large deflection shape sensing of a continuum manipulator for minimally-invasive surgery. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 201–206. ISSN: 1050-4729.

[48] Liu, J., Shou, J., Fu, Z., Zhou, H., Xie, R., Zhang, J., Fei, J., and Zhao, Y. (2020). Efficient reinforcement learning control for continuum robots based on Inexplicit Prior Knowledge. *arXiv:2002.11573 [cs]*. arXiv: 2002.11573 version: 2.

[49] Liu, Q. and Wu, Y. (2012). Supervised Learning.

[50] Lu, Y., Chen, W., Chen, Z., Zhou, J., and Liu, Y. (2022). FBG-Based Variable-Length Estimation for Shape Sensing of Extensible Soft Robotic Manipulators. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 01–08. ISSN: 2153-0866.

[51] Luo, K. (2023). Modeling of continuum robots: A review. *Journal of Physics: Conference Series*, 2634(1):012029.

[52] Mehryar Mohri, Afshin Rostamizadeh, A. T. (2018). *Foundations of Machine Learning*. MIT Press.

[53] Mitros, Z. (2017). [2018] Towards Modelling Multi-Arm Robots: Eccentric Arrangement of Concentric Tubes.

[54] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

[55] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533.

[56] Morimoto, R., Nishikawa, S., Niiyama, R., and Kuniyoshi, Y. (2021). Model-Free Reinforcement Learning with Ensemble for a Soft Continuum Robot Arm. In *2021 IEEE 4th International Conference on Soft Robotics (RoboSoft)*, pages 141–148.

[57] Nguyen, T.-D. and Burgner-Kahrs, J. (2015). A tendon-driven continuum robot with extensible sections. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2130–2135.

[58] Ouyang, B., Liu, Y., and Sun, D. (2016). Design of a three-segment continuum robot for minimally invasive surgery. *Robotics and Biomimetics*, 3(1):2.

[59] Park, M., Ohm, Y., Kim, D., and Park, Y.-L. (2019). Multi-Material Soft Strain Sensors with High Gauge Factors for Proprioceptive Sensing of Soft Bending Actuators. In *2019 2nd IEEE International Conference on Soft Robotics (RoboSoft)*, pages 384–390.

[60] Prasai, A., Jaiprakash, A., Pandey, A., Crawford, R., Roberts, J., and Wu, L. (2016). *Design and Fabrication of a Disposable Micro End Effector for Concentric Tube Robots*.

[61] Rao, P., Peyron, Q., Lilge, S., and Burgner-Kahrs, J. (2021). How to Model Tendon-Driven Continuum Robots and Benchmark Modelling Performance. *Frontiers in Robotics and AI*, 7:223.

[62] Rao, P., Pogue, C., Peyron, Q., Diller, E., and Burgner-Kahrs, J. (2023). Modeling and Analysis of Tendon-Driven Continuum Robots for Rod-Based Locking. *IEEE Robotics and Automation Letters*, 8(6):3126–3133.

[63] Rebala, G., Ravi, A., and Churiwala, S. (2019). *An Introduction to Machine Learning*. Springer. Google-Books-ID: u8OWDwAAQBAJ.

[64] Robinson, G. and Davies, J. (1999). Continuum robots - a state of the art. In *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, volume 4, pages 2849–2854, Detroit, MI, USA. IEEE.

[65] Roesthuis, R. J. and Misra, S. (2016). Steering of multisegment continuum manipulators using rigid-link modeling and fbg-based shape sensing. *IEEE Transactions on Robotics*, 32(2):372–382.

[66] Rummery, G. A. and Niranjan, M. (1994). On-line q-learning using connectionist systems.

[67] Russell, S. and Norvig, P. (2009). *Artificial Intelligence: A Modern Approach*. Pearson, Upper Saddle River, 3rd edition edition.

[68] Sahu, S. K., Tamadon, I., Rosa, B., Renaud, P., and Menciassi, A. (2022). A Spring-Based Inductive Sensor for Soft and Flexible Robots. *IEEE Sensors Journal*, 22(20):19931–19940.

[69] Satheeshbabu, S., Uppalapati, N. K., Chowdhary, G., and Krishnan, G. (2019). Open Loop Position Control of Soft Continuum Arm Using Deep Reinforcement Learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 5133–5139. ISSN: 2577-087X.

[70] Schulman, J., Levine, S., Moritz, P., Jordan, M. I., and Abbeel, P. (2017). Trust Region Policy Optimization. Technical report. arXiv:1502.05477 [cs] type: article.

[71] Searle, T. C., Althoefer, K., Seneviratne, L., and Liu, H. (2013). An optical curvature sensor for flexible manipulators. In *2013 IEEE International Conference on Robotics and Automation*, pages 4415–4420. ISSN: 1050-4729.

[72] Sears, P. and Dupont, P. (2006). *A Steerable Needle Technology Using Curved Concentric Tubes*.

[73] Seleem, I. A., El-Hussieny, H., and Ishii, H. (2023). Recent Developments of Actuation Mechanisms for Continuum Robots: A Review. *International Journal of Control, Automation and Systems*, 21(5):1592–1609.

[74] Shen, H. (2016). Meet the soft, cuddly robots of the future. *Nature*, 530(7588):24–26.

[75] Sincak, P. J., Prada, E., Mikova, L., Mykhailyshyn, R., Varga, M., Merva, T., and Virgala, I. (2024). Sensing of Continuum Robots: A Review. *Sensors*, 24(4):1311.

[76] Su, H.-J. (2009). A pseudorigid-body 3r model for detmining large deflection of cantilever beams subject to tip loads. *Journal of Mechanisms and Robotics*, 1.

[77] Sutton, R. S. and Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. MIT Press.

[78] Szepesvári, C. (2010). *Algorithms for Reinforcement Learning*, volume 4.

[79] Tian, Y., Luan, M., Gao, X., Wang, W., and Li, L. (2016). Kinematic Analysis of Continuum Robot Consisted of Driven Flexible Rods. *Mathematical Problems in Engineering*, 2016:e6984194.

[80] Till, J., Aloi, V., and Rucker, D. (2019). Real-Time Dynamics of Soft and Continuum Robots based on Cosserat-Rod Models. *The International Journal of Robotics Research*, 38:723–746.

[81] Torres, L. G., Kuntz, A., Gilbert, H. B., Swaney, P. J., Hendrick, R. J., Webster, R. J., and Alterovitz, R. (2015). A motion planning approach to automatic obstacle avoidance during concentric tube robot teleoperation. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2361–2367. ISSN: 1050-4729.

[82] Trivedi, D., Lotfi, A., and Rahn, C. D. (2007). Geometrically exact dynamic models for soft robotic manipulators. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1497–1502.

[83] Veiga, T. d., Chandler, J. H., Lloyd, P., Pittiglio, G., Wilkinson, N. J., Hoshiar, A. K., Harris, R. A., and Valdastri, P. (2020). Challenges of continuum robots in clinical context: a review. *Progress in Biomedical Engineering*, 2(3):032003.

[84] Walker, I. D. and University, C. (2017). Use of continuum robots for remote inspection operations. In *2017 Computing Conference*, pages 1382–1385.

[85] Wang, X., Wang, S., Liang, X., Zhao, D., Huang, J., Xu, X., Dai, B., and Miao, Q. (2022). Deep Reinforcement Learning: A Survey. *IEEE Transactions on Neural Networks and Learning Systems*, PP:1–15.

[86] Wang, Y., Ju, F., Cao, Y., Yun, Y., Wang, Y., Bai, D., and Chen, B. (2019). An aero-engine inspection continuum robot with tactile sensor based on EIT for exploration and navigation in unknown environment. In *2019 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 1157–1162. ISSN: 2159-6255.

[87] Watkins, C. J. C. H. and Dayan, P. (1992). Q-learning. *Machine Learning*, 8(3):279–292.

[88] Webster, R. J., Okamura, A. M., and Cowan, N. J. (2006). Toward Active Cannulas: Miniature Snake-Like Surgical Robots. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2857–2863. ISSN: 2153-0866.

[89] WebsterIII, R. J. and Jones, B. A. (2010). Design and Kinematic Modeling of Constant Curvature Continuum Robots: A Review. *I. J. Robotic Res.*, 29:1661–1683.

[90] Wooten, M., Frazelle, C., Walker, I. D., Kapadia, A., and Lee, J. H. (2018). Exploration and Inspection with Vine-Inspired Continuum Robots. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5526–5533. ISSN: 2577-087X.

[91] Wu, L. (2015). *Motion planning of continuum tubular robots based on features extracted from statistical atlas*.

[92] Wu, L., Song, S., Wu, K., Lim, C. M., and Ren, H. (2017). Development of a compact continuum tubular robotic system for nasopharyngeal biopsy. *Medical & Biological Engineering & Computing*, 55(3):403–417.

[93] Wurdemann, H., Sareh, S., Shafti, A., Noh, Y., Faragasso, A., Liu, H., Althoefer, K., Chathuranga, D., and Hirai, S. (2015a). Integrated soft bending sensor for soft robotic manipulators. *Joint Workshop on Computer/Robot Assisted Surgery*.

[94] Wurdemann, H. A., Sareh, S., Shafti, A., Noh, Y., Faragasso, A., Chathuranga, D. S., Liu, H., Hirai, S., and Althoefer, K. (2015b). Embedded electro-conductive yarn for shape sensing of soft robotic manipulators. In *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 8026–8029. ISSN: 1558-4615.

[95] Xie, Z., Yuan, F., Liu, J., Tian, L., Chen, B., Fu, Z., Mao, S., Jin, T., Wang, Y., He, X., Wang, G., Mo, Y., Ding, X., Zhang, Y., Laschi, C., and Wen, L. (2023). Octopus-inspired sensorized soft arm for environmental interaction. *Science Robotics*, 8(84):eadh7852.

[96] Xu, K. and Pérez-Arancibia, N. O. (2020). Electronics-Free Logic Circuits for Localized Feedback Control of Multi-Actuator Soft Robots. *IEEE Robotics and Automation Letters*, 5(3):3990–3997.

[97] Yaming, W., Ju, F., Cao, Y., Yun, Y., Wang, Y., Bai, D., and Chen, B. (2019a). An aero-engine inspection continuum robot with tactile sensor based on EIT for exploration and navigation in unknown environment. In *2019 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 1157–1162. ISSN: 2159-6255.

[98] Yaming, W., Ju, F., Yun, Y., Yao, J., Wang, Y., Hao, G., and Bai, C. (2019b). An inspection continuum robot with tactile sensor based on electrical impedance tomography for exploration and navigation in unknown environment. *Industrial Robot: the international journal of robotics research and application*, ahead-of-print.

[99] Yan, H., Wang, Y., Shen, W., Li, F., Gao, G., Zheng, T., Xu, Z., Qian, S., Chen, C.-y., Zhang, C., Yang, G., and Chen, T. (2022). Cable-Driven Continuum Robot Perception Using Skin-Like Hydrogel Sensors. *Advanced Functional Materials*, 32(34):2203241.

[100] Yang, Z., Yang, H., Cao, Y., Cui, Y., and Zhang, L. (2023). Magnetically Actuated Continuum Medical Robots: A Review. *Advanced Intelligent Systems*, 5(6):2200416.

[101] Yoshikawa, D., Shimizu, M., and Umedachi, T. (2023). A single motor-driven continuum robot that can be designed to deform into a complex shape with curvature distribution. *ROBOMECH Journal*, 10(1):18.

[102] You, X., Zhang, Y., Chen, X., Liu, X., Wang, Z., Jiang, H., and Chen, X. (2017). Model-free control for soft manipulators based on reinforcement learning. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2909–2915. ISSN: 2153-0866.

[103] Zhao, W., Zhang, Y., and Wang, N. (2021). Soft Robotics: Research, Challenges, and Prospects. *Journal of Robotics and Mechatronics*, 33(1):45–68.

Appendix A

Appendix A : CD