Towards Learning-Based Control for a Tendon-Driven Parallel Continuum Robot

Spencer Teetaert

Supervisor: Professor Jessica Burgner-Kahrs

Graduate Student Supervisors: Reinhard Grassmann & Sven Lilge

April 2023

B.A.Sc. Thesis

Division of Engineering Science
UNIVERSITY OF TORONTO

Abstract

Continuum robots prove difficult in modelling and control because of their extremely non-linear behaviour and complex modelling requirements. Traditional beam modelling methods are either computationally inefficient or are incapable of accurately controlling non-simulated robots, even ones that are well constructed. The added complexity of trying to model inaccuracies in robot construction makes controlling continuum robots challenging. Parallel continuum robots add the opportunity of higher applied forces to the compliant robots, but they only add to the complexity of the control problem. With the rise of machine learning many deep neural networks have been demonstrated effective in modelling complex relationships. In this work we take steps towards learning-based control of a planar parallel tendon-driven continuum robot. This work introduces a robot prototype designed to be a standard research platform for learning based control design for PCRs, a dataset collected with the intent of use in both state estimation and control tasks, two baseline controllers to use as a reference point for future learning based work, and several negative learning-based control experiment results to inform future work. The contributions in this thesis lay the foundations for learning-based state estimation and control for a parallel continuum robot. All project code is available freely at: `https://github.com/spencerteetaert/pcr_control`.

Keywords: parallel continuum robots, differential control, machine learning, learning-based control, benchmarks

Acknowledgements

I would first like to thank Professor Burgner-Kahrs for the opportunity to pursue this research at CRL. This experience will be the foundation for my future academic work. Thank you to Reinhard Grassmann and Sven Lilge for guiding me through this project. They both provided a tremendous amount of help throughout the project both contributing to the work and guiding my goals in academia moving forward. Thank you to the one UTM shuttle bus driver who does not accept my ticket because "this school takes enough of your money". He really made my day during the weeks where it felt like nothing was working. Thank you to Callie Moore and Aidan Grenville, without whom I would have never made it through these last five years. Lastly, thank you to my dad. Though we disagree on many things, he has always been the one who encouraged me to be curious, to question why, and to figure things out for my own.

Contents

List of Symbols

Abbreviations

CR = Continuum Robot

$TDCR$ = Tendon-Driven Continuum Robot

PCR = Parallel Continuum Robot

CC = Constant Curvature (modelling assumption)

PID = Position, Integral, and Derivative controller

$RMSE$ = Root Mean Squared Error

OL = Open Loop (in controls)

CL = Closed Loop (in controls)

DOF = Degree Of Freedom

Math Symbols

O_{ee} = End effector position [$\in \mathbb{R}^2$]

O_i = Arm base position [$\in \mathbb{R}^2$]

O_{goal} = Goal position [$\in \mathbb{R}^2$]

$\ell_{ee,i}$ = Distance between base point and end effector point

$\ell_{goal,i}$ = Distance between base point and goal point

l_t = Length of a tendon

κ_i = Arm curvature

α_i = Arm angle

φ_i = Arm subtended angle

K_p = Controller proportional gain

K_d = Controller derivative gain

K_i = Controller integral gain

u = Control signal. Typically a position or velocity

q = Joint variable. Typically a motor rotation or angular velocity

Robot Physical Constants

ℓ_i = 80 cm, Arm length

ℓ_t = 11 mm, Distance between the beam and the tendon

G = 64, Motor gear ratio

r_m = 11 mm, Tendon spindle radius

List of Figures

List of Tables

1 Introduction

Continuum Robots (CRs) are a current topic of research because of their inherent compliance, ability to be manufactured on sub-millimeter scales, and ability to have non-linear shape deformations [1], [2]. This makes them ideal robots to be used in surgical and inspection applications [3], [4]. Because of a CR's compliance, it is limited in the force it can exert on the environment through a given end effector. Parallel Continuum Robots (PCRs) see multiple CRs attached at a common end effector, maintaining system compliance while enabling the robot to apply higher forces to the environment [2], [5].

Forward and inverse kinematic modelling of a robot can be used in model-based control to move a robot with specified motions. Static modelling allows for the consideration of forces while dynamic modelling considers temporal effects on the system. Current models of CRs are unable to account for a number of complex factors in the system, such as internal robot friction, surface friction, and external loads, without significant computational overhead [6]. To achieve computation speeds that enable real-time control, often assumptions are made to simplify the robot's static model to a simple kinematic one [7] or to represent the robot's state with geometric approximations [6]–[8]. These simplifications greatly improve computation time while sacrificing model predictive accuracy. It is desirable to have the means to model these systems in real-time without making accuracy-sacrificing assumptions that seldom hold true in real-world applications.

Machine learning approaches have promised faster results and more accurate approximations of complex systems [9]. Several works have demonstrated learning-based systems and their value to both the control and state estimation tasks [10]–[14]. It is therefor a natural next step for PCRs to take advantage of the advantages of learning-based solution to help address the shortcomings of existing approaches. While some work has been done to use learning in CR control [15]–[17], nothing has been proposed for PCRs.

1.1 Contributions

To achieve learning-based control for this system while ensuring future works have an accurate comparison point, several foundational steps must be taken. This thesis takes several of those steps to advance the field of CRs towards learning-based solutions. The contributions of this thesis are:

1. Re-design and validation of a prototype for a two DOF planar parallel tendon-driven continuum robot. The robot is designed as an easy to implement research platform by using off-the-shelf components and 3D printed parts and being built for modularity and rapid reconfiguration

2. Establishes two real-time baseline controllers utilizing PID control and the constant curvature assumption that satisfy a 1000 Hz update rate and have zero closed-loop asymptotic tracking error in task space

3. Establishes a data set for both control and state estimation tasks in addition to providing a learning-based pipeline for these tasks

4. Presents several negative learning-based control results to inform future work

1.2 Document Structure

In Section 2 a background of concepts is provided alongside a review of current state-of-the-art modelling methods. Section 3 describes the robot prototype, baseline controller derivations, and dataset generation methodology. Section 4 will present the results of the four aspects of this project. Section 5 will explore the significance of the experiment results and highlight the limitations of this project. Lastly, Section 6 will explore potential future work that could follow the proposed work.

2 Background

In this section we consider a portion of available literature on both CRs and control to motivate the use of learning-based control for the PCR considered in this work. As learning-based control for a PCR configuration has not yet been studied in the available literature, the scope of this section is only to provide a background of relevant topics and not a comprehensive review of all available control and modelling methods.

2.1 Types of Continuum Robots

CRs can be defined as "an actuatable structure whose constitutive material forms curves with continuous tangent vectors." [3] CRs are robots with an infinite number of joints and degrees of freedom. By using flexible components, CRs are inherently compliant, making them useful in applications where interactions with stiff joints may cause damage to the environment. They are also able to be manufactured on sub-millimeter scales. As such, these robots have seen use in medical and inspection applications [1], [3], [4]. Continuum robots come in a variety of configurations, materials, and actuation methods [18]. Modelling methods for continuum robots are discussed in Section 2.2. The subset of continuum robots pertaining to this thesis is considered.

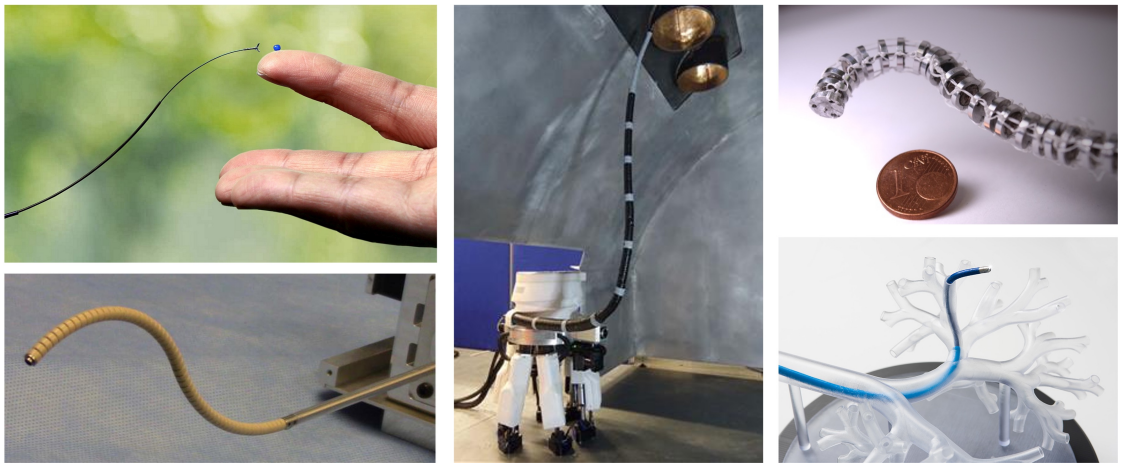


Figure 1: Continuum robot examples, from [19]

2.1.1 Tendon-Driven Continuum Robots

Tendon-Driven Continuum Robots (TDCR) are a class of CRs that are actuated by contracting a tendon that is attached to a point along the robot's backbone. The backbone is typically a highly elastic rod or beam that provides stiffness to the robot while remaining compliant. The tendon

is constrained to be a fixed distance from the backbone by a series of spacer disks. Contracting the tendon results in the backbone bending towards the side the tendon is offset on [6], [20]. An "arm" of a TDCR refers to a single beam that can be actuated independently of other arms in the robot.

2.1.2 Parallel Continuum Robots

PCRs use multiple arms joined together to increase the stiffness and maximum applied force of a CR while maintaining the inherent compliance that CRs have to offer [2].

2.1.3 Robot Configurations

PCRs can take on a wide variety of configurations. In [5], a six-beam PCR is proposed and modelled based on Cosserat rod theory. Flexible rods are translationally pushed and pulled to achieve actuation. A tracking accuracy of 2.89% of the length of one arm of the robot was achieved. [8] introduces a three-beam parallel TDCR with six degrees of freedom at the end effector. The authors model each beam using a geometric kinematic model based on the constant curvature assumption. A planar configuration reduces the task space to a 2D plane. These robots have at most three degrees of freedom at the end effector; two spatial components and a rotational one. In [7], a planar PCR is proposed along with a geometric model controller based on the constant curvature assumption. It achieves an accuracy of 1% of the length of one arm of the robot.

2.2 Modelling Continuum Robots

It is desirable to construct a forward and inverse kinematic model when controlling a robot. A forward kinematic model provides an expression for the end effector position given a robot's joint variables. It is used to estimate the end effector position of a robot during operation when direct measurements are not available. An inverse kinematic model solves for a robot's joint variables given a desired end effector position. It is used in control tasks to move the robot into a desired configuration. In both cases, the accuracy of the model is determined by a number of factors including the simplifying assumptions made in exchange for increased computational efficiency [6]. Figure 2 demonstrates the run time benefit from using models that make simplifying assumptions and the sacrifice that is made in terms of accuracy.

There are three major types of robot models that we consider: (1) kinematic models, (2) static models, and (3) dynamic models. Kinematic models model the shape of the robot without consideration of forces. Static models consider the robot at points of equilibrium, where the

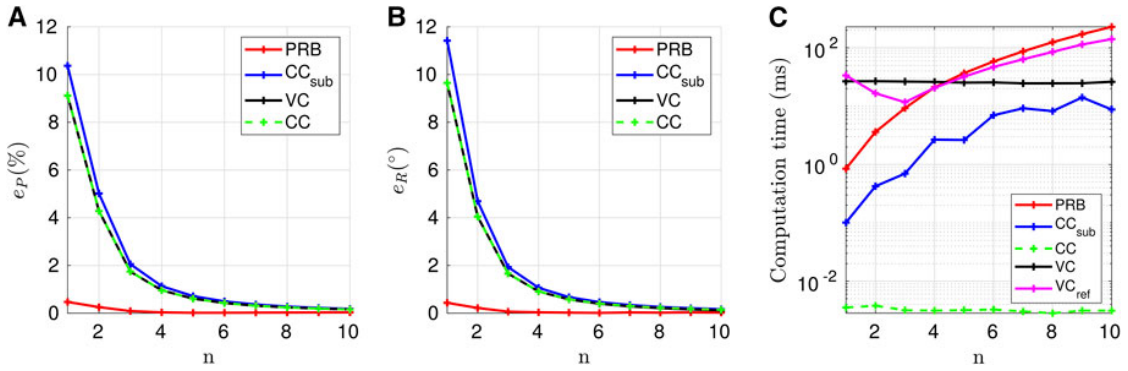


Figure 2: Timing analysis of pseudo-rigid body, constant curvature, and variable curvature modelling methods in a simulated environment, from [6]

robot is not moving and has zero net force. Dynamic models model the robot as it's moving, accounting for transient effects (such as spring forces, friction, air resistance, etc.).

In rigid robotics, simple linear models exist that accurately model the robot's motion. CRs do not have such a luxury due to their non-linear deformation. As such, more system assumptions or more complex models must be used for even the simplest of CRs [6]. As this is an emerging research field, the accuracy of models for CRs is not yet comparable to conventional industrial robots.

2.2.1 Constant Curvature Assumption

The most common and convenient kinematic modelling approach for continuum robots uses the constant curvature (CC) assumption [21]. It assumes that the curvature throughout the entire length of an arm is constant. This assumption greatly reduces the complexity of the forward and inverse kinematic models of a continuum robot.

Euler Beam Theory Euler beam theory can be used to create a static model assuming that the effects of shear and twist are negligible [6]. The CC assumption allows for computationally efficient implementations of the inverse kinematics problem. It does however result in a number of inaccuracies as beams under high bending do not have CC. Modelling a continuum robot using the CC assumption and Euler beam theory has been shown to result in poorer accuracy compared to using a variable curvature model [6].

2.2.2 Variable Curvature Representation

Without losing generality, one can model the curvature of a continuum robot using a variable curvature representation. This representation makes no assumptions about the backbone shape

as it can model any point along it with six degrees of freedom [6].

Cosserat Rod Theory The Cosserat theory of elastic rods extends the variable curvature kinematic representation into a static model. Modelling in this way accounts for shear deformations, something Euler beam theory did not. While the Cosserat rod theory approach has been shown to produce excellent accuracy in continuum robot applications [6], [22]–[24], it does so at great computation costs, resulting in the inability to be used for real-time control [6], [23]. A number of implementations for the cosserat theory model have been realized with accuracies ranging from 12%[24] to 4×10^{-10} m - 8×10^{-10} m Root Mean Squared Error (RMSE) for a robot of length 30 cm[23].

2.3 Machine Learning

Machine learning has emerged as a useful tool for estimating complex functions using data collected from a system. In robotics, learning has been used to develop controllers for complex systems that are difficult to model. Deep learning approaches have shown promise in modelling complex dynamic behaviour for controller design in robotics [9]. Because data-driven approaches can estimate arbitrarily complex functions [25] they are thought to be useful in continuum robots. Learning-based applications in continuum robotics have primarily focused on learning the inverse statics and kinematics of a robot [15]. With each of the results reviewed here, one must be weary of directly comparing results directly as large variations in performance can arise from changes in robot configuration, actuation type, and construction. In each of the works reviewed, the proposed learning-based model outperformed baseline approaches on the same robot.

2.3.1 In Continuum Robotics

Data-driven approaches In [26], the authors use a model-free approach using an adaptive Kalman Filter. This uses sensor data to update the model's uncertainty estimations during operation, resulting in a more controller that had an experimental tracking RMSE of 1.52 mm (with a 20 cm - 25 cm long pneumatic continuum robot). In [27], the authors implement three regression models to learn the inverse kinematics of a flexible surgical robot, achieving a top RMSE of 2.1275 mm with a KNN regression. The authors do not report the total robot length.

Deep learning approaches For deep learning approaches to work, a large amount of data or an accurate simulation environment must be available. As the dynamic effects of a continuum robot are what is difficult to model, simulation environments are not readily available. In [16], a dataset is released for concentric tube continuum robots. The authors propose a baseline learning

model that achieves tip tracking errors of 0.74 mm (or 0.4% of the robot's length). Using deep neural networks for approximating the inverse statics equation is the approach used in [17]. The authors are able to achieve errors of around 7 mm[1] for an underwater soft continuum robot (or about 2.5% of the robot's length). For both of these implementations, a large amount of data is required that maps the robot's end effector position to its joint configuration, marking a large drawback to deep learning-based models.

2.3.2 In Other Robotics

Although learning-based solutions are limited in their applications to CRs, they have been deeply explored in other robotics in areas that may be applicable to CRs such as sensing [12], [28], state estimation [10]–[12], and control [13], [14]. Each of these aspects have had results that demonstrate learning-based solutions and their ability to replace more conventional methods. Often however, hybrid approaches that make use of conventional modelling methods that are augmented using learning are shown to have the best results.

In [10], the authors highlight several elements in deep learning that prove useful in state estimation including the use of long short-term memory (LSTM) layers to enable long-term correspondence between input data, the use of Bayesian modelling to help account for complex system noise, and applying attention mechanisms to learn what is important and unimportant in the input data at each time step. The survey goes on to highlight the value in hybrid approaches that integrate learned systems with model-based approaches. The authors in [11] take this one step further by demonstrating the hybrid approach on a manipulation task with a deformable object. The authors show that they can accurately estimate a rope's full state using a self-supervised deep learning network trained on image data.

Robotic systems rely on their sensor data as the basis for state estimation. This data is generally corrupted by some amount of noise. [12] demonstrates the use of deep learning to explicitly account for system noise in attitude estimation of a UAV. The authors in [28] propose a pipeline for explicit handling of uncertainty caused from noisy inputs. They demonstrate their approach's success in robotic applications by estimating model uncertainty by propagating noise through the model and quantifying the effect.

For learning-based control, [13] highlights how neural networks can improve results in model predictive control (MPC). It looks at three different applications including using learning to model system dynamics and using learning to enhance objective function design. In [14], deep learning is used to solve the inverse kinematic problem for a rigid parallel industrial robot. The

[1]Number, as reported, is an estimation by the thesis author. See reference paper for complete results breakdown.

work shows that deep learning based inverse kinematic solutions have applications in parallel robotics, boasting a 99.8% accuracy compared to the analytical model.

2.4 Summary

PCR design is still an open area of research. Work has focused on the benefits of these robots in their relevant application spaces. Research modelling and control for CRs have largely focused on single-beam CRs. A research gap exists for modelling and control of PCRs that is unaddressed by current classical modelling methods. Significant modelling assumptions that ignore dynamic effects of the system or make inaccurate claims about the kinematic description of the system allow for fast but inaccurate control. When these assumptions are not made, controllers have infeasible computation speeds for real-time control. Data-driven approaches may be able to solve this problem but are underdeveloped for parallel continuum robots. This thesis aims to address this gap by taking steps towards learning-based control for a planar parallel TDCR.

3 Methods

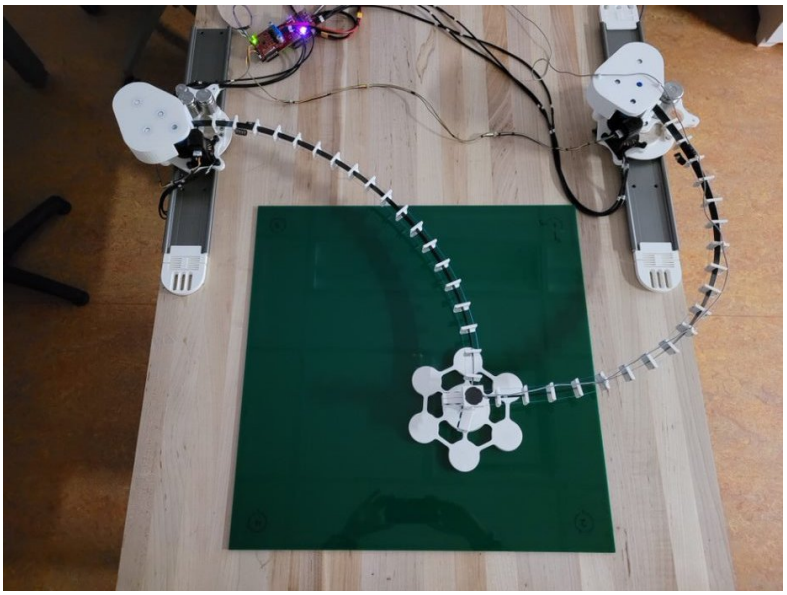


Figure 3: Prototype setup

3.1 Prototype Hardware

The robot is designed with the Open Continuum Robotics project in mind. It uses easily accessible, off-the-shelf components, and 3D printed parts so that other groups can reconstruct the same robot. A full breakdown of the robot's physical design can be found in Appendix **??**.

3.1.1 Physical Description of Prototype

The arms are each 80 cm long and consist of a 0.0200" thick, 0.75" wide beam made from 1095 spring steel, with 19 evenly spaced 3D printed tendon spacers. The spacers hold one tendon on each side of the beam 11 mm away from the steel beam. The 1095 spring steel beam is used as the central beam as it provides rigidity in the direction of motion off the robot's planar surface, while remaining flexible along the plane. The high elasticity of the material allows the system to undergo large curvatures without permanently deforming the beam. The tendons used are 30 lbs braided fishing line (Super8Slick V2, Power Pro, CA, USA). Each tendon has four 0.75" nuts on it that act to maintain tension during operational periods where it is being extended.

The end effector features a wide base to prevent the end effector from twisting off the plane. Revolute joints link the ends of each arm to the motor bases and the end effector, allowing both ends to rotate freely about the z-axis. The two arms share the same end effector. The bases

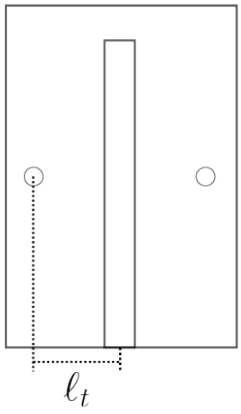


Figure 4: Cross section of the beam spacers. $\ell_t = 11\,\text{mm}$

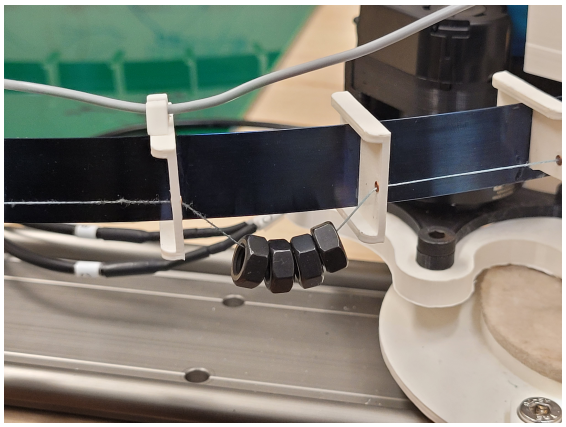
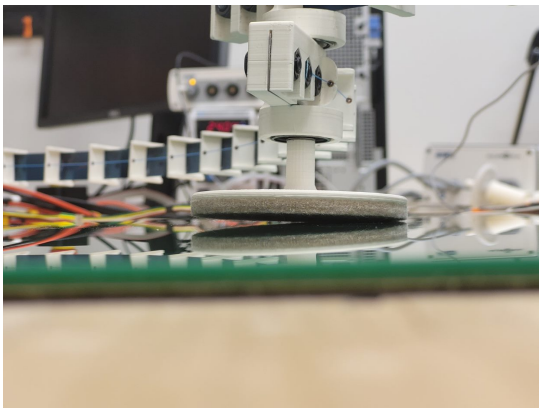


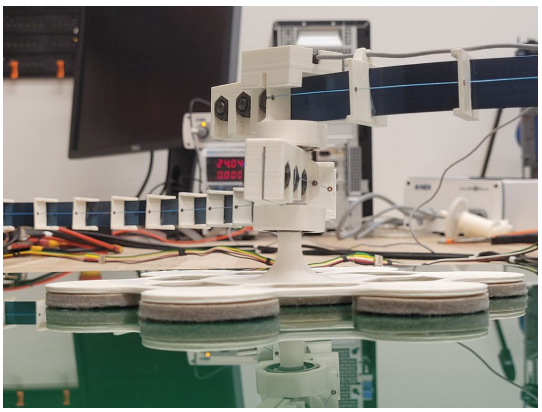
Figure 5: Four 0.75" nuts placed on a tendon to maintain tendon tension throughout operation

for each arm are positioned 60 cm apart.Each arm having its own independent degree of freedom grants the end effector in this configuration two degrees of freedom laying on the table plane. At this distance, the reachable workspace of the end effector covers a majority of the AURORA tracker workspace. During operation, a smaller end effector base does not provide enough area to balance the moments being applied by the two arms actuating the piece from different heights. Figure 6 demonstrates this effect in action. Felt pads are attached to the bottom of the end effector to reduce friction between the end effector and table surface.

Gearboxes are actuated directly by motors for each arm. The gear boxes are 3D printed but use purchase gears with a gear ratio of 16:1, enabling significantly higher applied torques than the motors are natively capable of. A 4:1 gear ratio is achieved by the motor unit. This results in a total gear ratio of 64:1 for the actuation unit. Each box contains two 10 mm gears (2662N313,



(a) Smaller base causing the end effector to twist off the planar surface



(b) Larger base ensuring end effector remains on the plane

Figure 6: Comparison of end effector sizes

McMaster-Carr Supply Company, IL, USA) and three 40 mm gears (2662N321, McMaster-Carr Supply Company, IL, USA) to achieve this rotation. The gearbox actuates a 3D printed spindle with a radius of 11 mm. Two tendons are spooled around the spindle in opposite directions, so when it rotates, one tendon is extended while the other is contracted. Both tendons are firmly attached at the end effector side of each arm. This equal and opposite actuation of the tendons is what causes the arm to bend.

The workspace additionally has an acrylic sheet used to reduce friction. Beneath the table surface, an AURORA electromagnetic tracking system (20-20 planar, Northern Digital Inc., ON, Canada) is positioned. The tracker is positioned to ensure maximal coverage of the robot's task space. The tracker is limited to an effective tracking area of 50 cm x 50 cm on the workspace plane. This unit tracks a wire coil which is fastened on the robot's end effector, directly above the revolute joint. This is where the end effector position is defined to be as a point in space.

3.1.2 Electronic Components

Each arm is driven by a single Antigravity drone motor (MN4004 KV300, T-MOTOR, JX, P.R. China) which are both controlled by an off-the-shelf microprocessor (LAUNCHXL-F28069M, Texas Instruments, Dallas, USA) fitted with two booster cards (BOOSTXL-DRV8305EVM, Texas Instruments, Dallas, USA). The motors are part of an actuation unit shown in Figure 7 and contain a 4:1 gear ratio and an Avago optical encoder (AEDM 5810, Broadcom Inc., CA, USA) that is used to provide system feedback at a rate of 1000 hz to the workstation. The actuation setup comes from the Open Continuum Robot project [29], [30]. A single micro-controller

communicates with a workstation through the CAN bus protocol. The robot is powered using a 240 W power supply operating at 24 V. An emergency stop is installed within reach of the operator for safety.

The workstation used is a Dell OptiPlex 7090. It boasts an Intel Core i7-10700 CPU with 16GB of memory. It runs the RT-Preempt real time Linux kernel. The station communicates with the TI micro-controllers through a CAN-to-PCI Express interface card (IPEH-003027, PEAK system, Hessen, Germany). The station interfaces with the Aurora tracker through USB. Because the code for this thesis is written in Python, use of the real time kernel is not strictly required as Python does not support real time operations. For this project Python was deemed an appropriate language as it enables faster development times, and the shortcoming in terms of runtime is not seen to cause issues on the system.

3.1.3 Maximum Curvature Test

Before the 16:1 gearbox was designed and installed, the actuation unit had a 4:1 gear ratio. An experiment was conducted to determine the arm length that enabled the largest range in the end effector position. Knowing the maximum and minimum distance that can be reached from an arm's base is required to set workspace bounds for the robot controller. The maximum distance is trivially given by the length of the arm. To find the minimum distance, a maximum curvature is applied by sending the highest current command a motor can support. The curvature that equates the bending forces to the maximum motor torque is taken as the maximum curvature. In this position, the distance between the end effector and arm base was measured. The results are displayed in Table 1 and Figure 8. The arms were left at 80 cm length as no notable gain in performance was observed at either test length.

Table 1: Arm distances during the maximum curvature test

Extended Distance	Fully Contracted Distance	Contraction Distance
80 cm	(59 ± 1) cm	(21 ± 1) cm
55 cm	(33 ± 1) cm	(22 ± 1) cm

This experiment showed the torque output of the motors resulted in a significantly smaller range of motion than what was initially expected. In response, a new gearbox was designed and manufactured to increase the maximum torque output of the robot. With the new gearbox, the robot is able to contract its arm completely with 0 cm between the end effector and base.

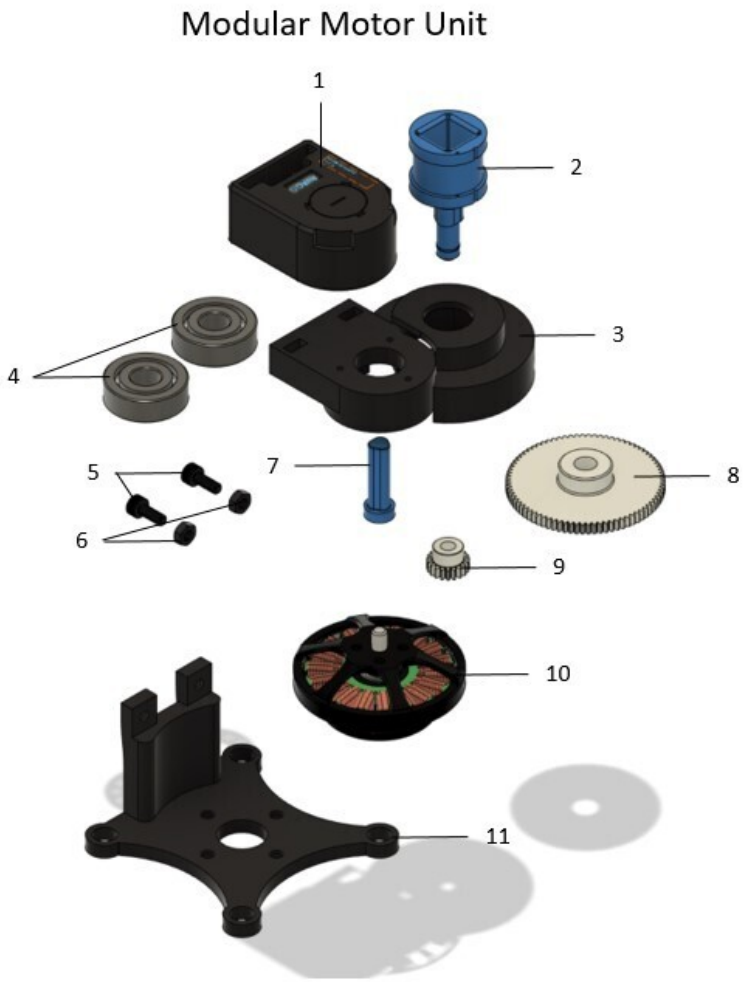
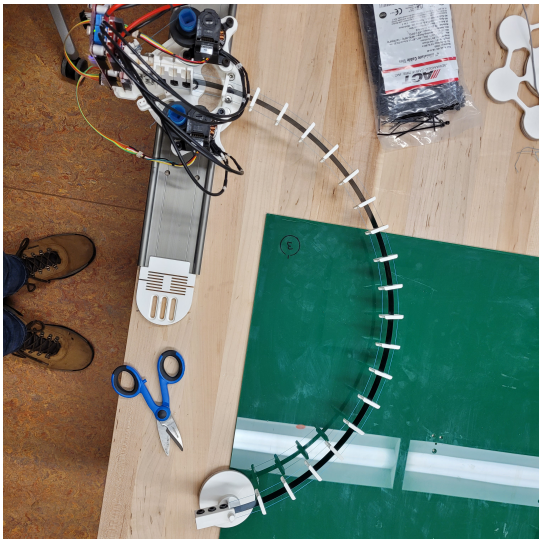
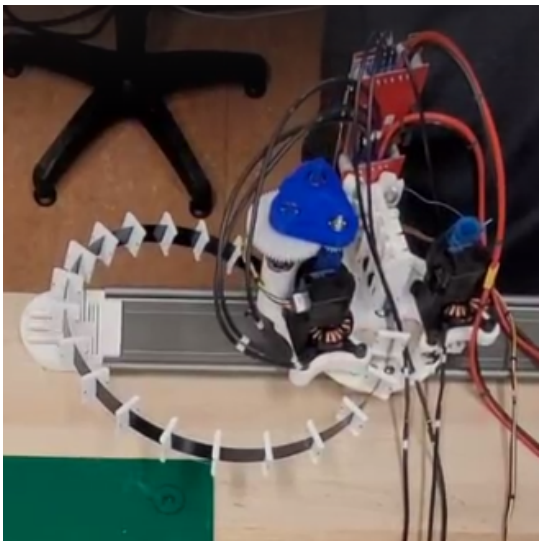


Figure 7: Complete schematic of the motor actuation unit, from [31]



(a) Without gearbox



(b) With gearbox

Figure 8: Maximum curvature tests

3.2 Robot System Controller

The robot is controlled using the schematic shown in Figure 9, referred to as the "system controller". This should not be confused with the "robot controller" which refers to the controller that maps target motions to joint values.

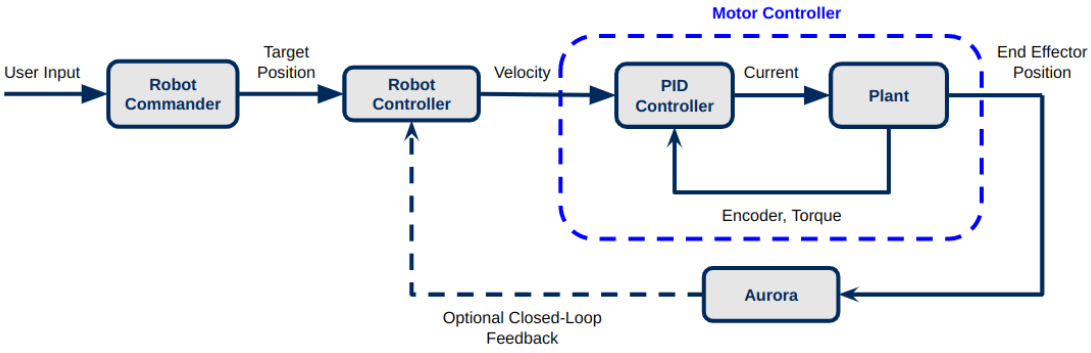


Figure 9: Robot system controller schematic

3.2.1 Controller Description

A high level commander module maintains the robot's state at all times. This module interfaces with the robot user, and determines which operation modes the robot is in. It handles logging, recovery behaviour, and manages a watchdog thread for each component of the system

that will raise an error if any process dies. All robot states are described in Table 3. During states that are engaged in point tracking, a termination criteria of the end effector being within 2 cm of the goal point is used. During trajectory tracking, no such condition is required as the goal point is progressed in time using a predefined trajectory. This module is referred to as the robot commander.

A closed loop cascade controller controls the motion of the robot. The inner loop uses a PID controller in conjunction with motor encoder feedback and a reference signal from the robot controller to determine the current to be sent to each motor. The gains used in this controller are shown in Table 2. This loop is referred to as the motor controller.

Table 2: Motor controller gains

Controller	K_p	K_d	K_i
Position	20	0.2	10
Position (Recovery)	0.05	0.7	0.03
Velocity	3	0	60

The robot controller determine the joint displacements or joint rates required to achieve a desired end effector motion. With this setup, the controller can either operate in a closed-loop scenario using measurements from the Aurora tracking system, or in an open-loop scenario, where an external state-estimation pipeline must be implemented. Explicit state estimation of this robot is outside of the scope of this thesis, so each proposed baseline controller will operate in the closed-loop configuration.

3.2.2 Lost AURORA Recovery

In the case of lost AURORA measurements, a recovery mode is entered where the robot returns to its home position using only motor encoder feedback. The home position is where both arms are fully extended and the motor encoders read a rotation of 0 rad. This recovery behaviour is the only motion that can be executed in a closed-loop configuration without AURORA measurements.

3.3 Baseline Models

As no other research provides a baseline that can be compared to for this system, I propose two baseline controllers: a differential controller based on the constant curvature assumption and a PID controller based on a prismatic joint assumption. The selection of the constant curvature

Table 3: Controller state description

State	**Description**	**Exit Condition**
ERROR	The robot has experienced a critical error	Manual reset of the software and hardware
STOPPED	The robot is blocked from tracking reference signals. Used when the robot should remain stationary while code runs	State moves to READY after code is completed
RANDOM TRACKING	The robot is tracking random target points. Used during data collection	User terminated
REFERENCE TRACKING	The robot is tracking a pre-loaded reference trajectory. Used during model testing	Returns to READY when reference trajectory is complete
REFERENCE STARTING	The robot is moving towards the start of a provided reference trajectory	Moves to HOLDING once the robot reaches the starting point to a reference trajectory
MANUAL TRACKING JOINT	The robot is tracking a point determined through the user control API in joint space	User terminated
MANUAL TRACKING TASK	The robot is tracking a point determined through the user control API in task space	User terminated
RECOVER	The robot is recovering from a lost Aurora signal. It is returning to the home position	Moves to READY when home position is reached
READY	The robot is stationary and able to take new commands	User terminated
HOLDING	The robot is intentionally waiting at a point before progressing to the next command. Used during testing to provide time to set up video feeds	Moves to REFERENCE TRACKING after a set period of time

model is motivated by its ease of derivation and implementation. Other models, such as those based on a variable curvature backbone representation may provide more accurate results than our selected baseline at the expense of increased computation [6]. These models also have their own challenges representing the dynamic effects of continuum robots. Future works may look at the results of these models on this system but it is deemed as beyond the scope of this thesis. The PID controller was chosen as it is a fast model both in runtime and implementation time, and can be tuned to work reasonably well in a closed-loop scenario.

3.3.1 Robot Model

First, a robot model is developed that maps an arm's curvature value to a tendon displacement. Equation (1) solves for the change of tendon length required to achieve a given constant curvature. Two solutions are provided, one for each side of the beam. Equation (1) is used to find the tendon length given a beam's curvature, assuming a fully constrained tendon path [6] along the length of the beam.

$$\Delta l_{t,i} = \pm \ell_i \ell_t \kappa_i \tag{1}$$

Equation (1) can be extended to consider the prototype motor's gear ratio and spindle radius which the tendons are wrapped around. This gives an expression for the motor rotation required to achieve the desired tendon length displacement. The motor rotation q found in Equation (2) is the joint variable for the proposed PCR. I take the positive solution as both tendons are actuated in their respective directions by the same motor.

$$q = \kappa G \ell_i \frac{\ell_t}{r_m} \tag{2}$$

3.3.2 Constant Curvature Controller

A model using the constant curvature assumption [21] is used as a baseline model for this parallel continuum robot. This model is used to solve the inverse kinematic problem. This is used to determine control signals for the robot during the data generation, enabling the ability to control the position of the end effector (with some error). Equation (3) gives a general expression relating the curvature of a constant curvature beam and a beam's base and end position [8].

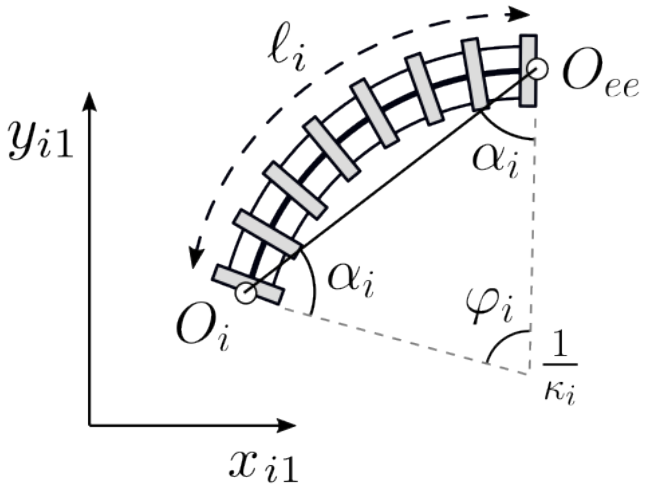


Figure 10: Constant curvature model diagram

$$
\begin{aligned}
||O_{ee} - O_i|| &= 2\frac{1}{\kappa_i}\cos(\alpha_i) \\
&= 2\frac{1}{\kappa_i}\sin\left(\frac{\varphi_i}{2}\right) \\
h(\kappa_i, O_{ee}) = h_i = 0 &= \frac{2}{\ell_i\kappa_i}\sin\left(\frac{\ell_i\kappa_i}{2}\right) - \frac{||O_{ee} - O_i||}{\ell_i}
\end{aligned} \tag{3}
$$

The solution for κ is found by finding the roots of Equation (3). For the parallel continuum case, one solves Equation (3) for each of the robot's arms. One extra check is required that the target point is within reach of each arm to guarantee a solution. The solution is found numerically using scipy.optim.fsolve[2]. As this model is strictly a position controller, it does not have the ability to reduce the robot's steady state error between its end effector and a goal point to zero. As such, I do not explore this controller in depth on the system, but rather use it as a starting point for its differential counterpart.

3.3.3 Differential Constant Curvature Controller

By taking the derivative of Equation (3) with respect to time, we can derive a differential version of the constant curvature position controller. This enables us to control the robot using velocity control in the motor controller.

$$
\frac{\partial h}{\partial \kappa}\frac{\partial \kappa}{\partial t} + \frac{\partial h}{\partial O_{ee}}\frac{\partial O_{ee}}{\partial t} = 0
$$

[2] `https://github.com/scipy/scipy/blob/v1.10.0/scipy/optimize/_minpack_py.py\#L48-L181`

$$A = \begin{bmatrix} \frac{\partial h_1}{\partial \kappa_1} & 0 \\ 0 & \frac{\partial h_2}{\partial \kappa_2} \end{bmatrix}, B = \begin{bmatrix} \frac{\partial h_1}{\partial O_{ee,x}} & \frac{\partial h_2}{\partial O_{ee,y}} \\ \frac{\partial h_1}{\partial O_{ee,x}} & \frac{\partial h_2}{\partial O_{ee,y}} \end{bmatrix}$$

$$A\dot{\kappa} + B\dot{O}_{ee} = 0$$

While matrices A and B are invertible, they contain the κ_i terms which must still be found numerically. A closed form solution to this model does not exist.

$$\dot{\kappa} = (-B_i^{-1}A_i)^{-1}\dot{O_{ee}} \tag{4}$$

In this particular model, point tracking is achieved by defining an error term between the end effector position and the goal point, and using a simple P controller to determine the desired end effector velocity. A value of $K_p = 0.01$ is used. This is shown in Equation (5) and can be combined with Equation (4) to retrieve the full controller model for the differential constant curvature controller shown in Equation (6).

$$\dot{O}_{ee} = K_p||O_{goal} - O_{ee}|| \tag{5}$$

$$\dot{\kappa} = K_p(-B_i^{-1}A_i)^{-1}||O_{goal} - O_{ee}|| \tag{6}$$

Using the robot model from Equation (2) yields the complete differential constant curvature controller shown in Equation (7). $\dot{q}$ can be commanded directly to the motor controller as a velocity set point.

$$\dot{q} = K_p G\ell_i \frac{\ell_t}{r_m}(-B_i^{-1}A_i)^{-1}||O_{goal} - O_{ee}|| \tag{7}$$

3.3.4 Prismatic Assumption PID Controller

A second controller is proposed by modelling each continuum arm as a prismatic joint. The idea is that the arm is capable of contracting and extending, which simply actuates the distance between the end effector and an arm's base point. Because the robot's workspace is clear of obstacles, no concern is required for maintaining an estimate of the shape of the robot. For every goal point, each arm has a specific distance it needs to extend or contract to which can be solved for independently of the other arms.

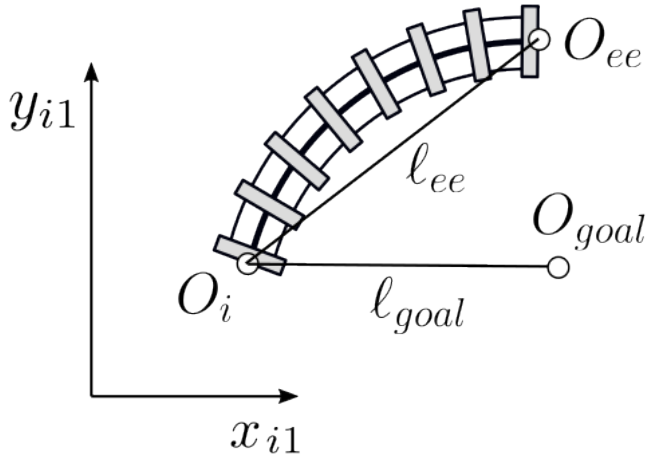


Figure 11: Prismatic PID model diagram

An error term (Equation (8)) is defined by subtracting the distance between the goal point and an arm's base, and the end effector position and an arm's base. This error is used in a PID controller to yield the joint velocity command in Equation (9). The gains found to result in desirable behaviour are $K_p = 4, K_d = 0, K_i = 0$.

$$\begin{aligned} e_i(t) &= ||O_{goal} - O_i|| - ||O_{ee} - O_i|| \\ &= \ell_{goal} - \ell_{ee} \end{aligned} \tag{8}$$

$$\dot{q}_i = K_p e_i(t) + K_d \frac{de_i}{dt} + \int e_i(t)dt \tag{9}$$

This controller has a closed form update, suggesting a computational advantage over the differential constant curvature model which must be solved iteratively. It also is significantly easier to derive and implement in software, making it a fast and easy option for controlling this robot. Because the model does not solve for a curvature value, the robot model is not required. The conversion to motor velocities is handled in the controller gains.

3.4 Data Collection

3.4.1 Workspace Coordinate Frame Definition

The workplace has a coordinate frame defined by three point markings on the green acrylic sheet at three corners (shown in Figure 12). During regular operation of the robot, all readings from the AURORA system are transformed to be represented in the workspace coordinate frame

(sometimes referred to as the "robot frame"). A method is used to calibrate the reference frame between each session on the robot. A total of 30 AURORA measurements are taken at each of the three points. The average position of the points is found by a simple mean over the 30 readings. The origin of the workspace is taken to be the location of the first point, O_1. The workspace axes are defined as follows:

$$\hat{x} = \frac{O_3 - O_1}{||O_3 - O_1||}$$
$$\tilde{y} = \frac{O_2 - O_1}{||O_3 - O_1||}$$
$$\hat{z} = \hat{x} \times \tilde{y}$$
$$\hat{y} = \hat{z} \times \hat{x}$$

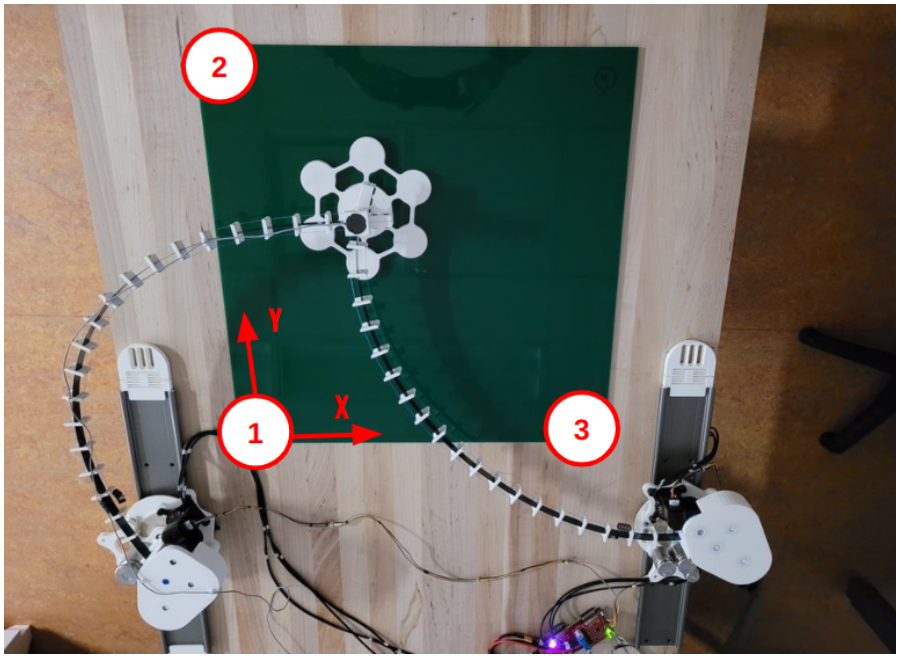


Figure 12: Robot workspace coordinate frame diagram

3.4.2 Data Schema

Data is collected from all parts of the system at various rates as described in Table 4. All data during operation is saved to log files with unique names that can later be loaded into a dataloader object.

Table 4: Robot data collection schema

Data Type	**Unit**	**Frequency in Hz**
Motor Controller		
Current command	A	100
Current draw	A	100
Motor position set point	rad	100
Motor position	rad	100
Motor angular velocity set point	rad/s	100
Motor angular velocity	rad/s	100
Controller type	N/A	100
Recording timestamp	s	100
AURORA System		
End effector position (AURORA frame)	m	5
End effector orientation (AURORA frame)	m	5
End effector position (robot frame)	m	5
Recording timestamp	s	5
Robot Commander		
Sample termination cause	From table 5	On termination
Commander state	From table 3	5
Recording timestamp	s	5

3.4.3 Collection Method

The robot controller used for data collection is the prismatic assumption PID baseline controller from Equation (9). It was selected because of its good relative performance on a number of metrics discussed in Section 4.2. Data was only collected using this single baseline due to project timing limitations. As such, the data collected only contains robot states that are accessible using this controller, which has the potential to instill bias into the dataset.

The sampling method used for deciding which paths to move the robot over is a random sampling over a uniform distribution. Sample goal points are drawn inside a 15 cm radius of the end effector position at the start of the sample and outside a 5 cm radius of that same position. An upper bound of 15 cm was selected as much above that would result in harsh robot movements that caused damage to the robot. A lower bound of 5 cm was selected as much below that would result in motor set points that result in an applied motor torque that is too low to overcome the

static friction of the end effector in some positions in the workspace.

A more complex path sampler was explored using randomized path generation while ensuring joint actuation smoothness, but was ultimately removed as it did not satisfy the workstation's real time runtime constraints. More information on this method is presented in Appendix B.

3.4.4 Termination Cases

Non-critical failures occur when for some reason the robot is not able to find a way to withing a 2 cm radius of the sampled goal point. A number of reasons can cause this, including lost AURORA measurements or a stalled end effector. The robot can recovery without user intervention from non-critical failure cases and the data from them is complete and usable.

Critical failures occur when the robot failed in a way that could result in corrupted data. This happens when a portion of the code crashes or the robot loses power, typically by the user hitting the emergency stop. In critical failure cases the user must reset the system manually and delete the affected data logs.

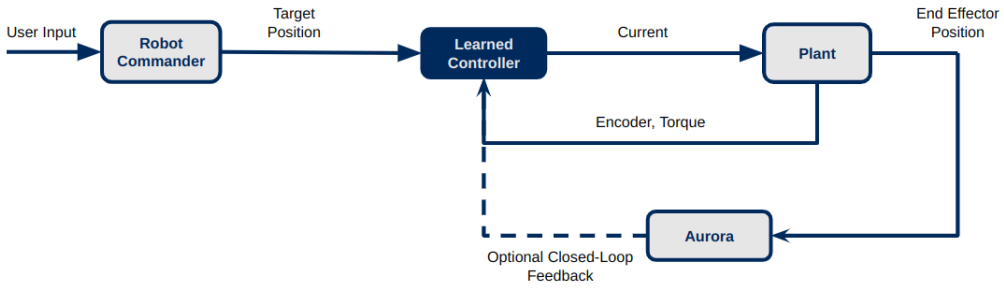
Successful samples occur when the robot successfully arrives at the sampled goal point without triggering any of the failure cases. A full description of all termination causes is provided in Table 5.

3.5 Learning Pipeline

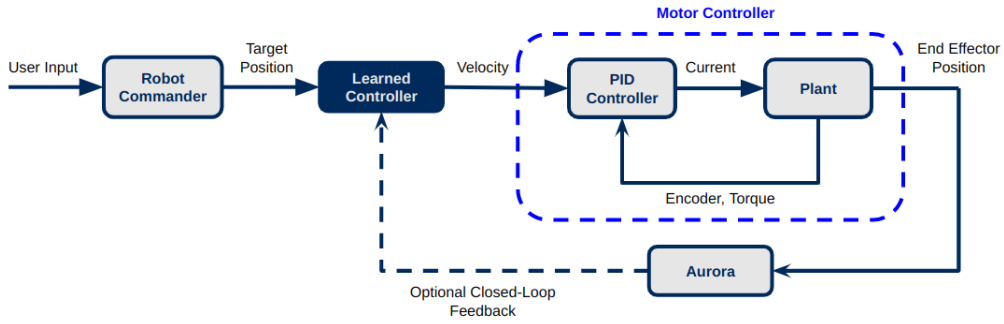
Two primary learning-based control pipelines are supported by this project. The first has trained models replacing both the robot controller and motor controller in the robot system. These models take as inputs a target position with system feedback and directly outputs motor current commands. The second has a trained model that acts as the robot controller, leaving the motor control loop as is.

Table 5: Robot sample termination types

Termination Type	**Description**
Success	The robot successfully reached the target point
Stalled	The robot's end effector velocity was below 0.001 m/s for more than 5 s
Aurora lost	At least 3 consecutive AURORA measurements were errors, typically caused by the end effector leaving the workspace
Recovery	The robot underwent recovery behaviour where AURORA readings are not necessarily available
Mode switch	The robot operator manually switched the robot into a different mode
User terminated	The user shut down the system using the control interface
Tracking finished	A reference trajectory was completed
Error	An error is raised during the trial. This results from a crashed thread, or a failed recovery attempt



(a) Learning-based controller implementation replacing motor controller



(b) Learning-based controller implementation as just the robot controller

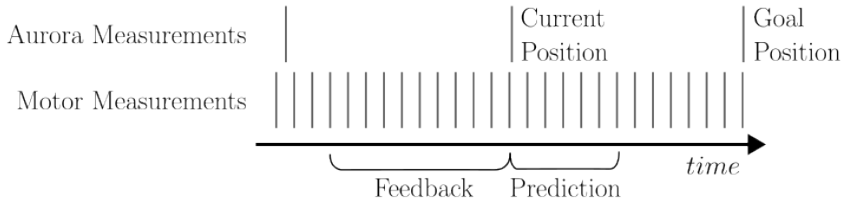
Figure 13: Learning-based controller implementation pipeline options

Option 1 In the first option (Figure 13a), the learned model is responsible for more aspects of control. This gives the model more flexibility in its approach to control as it has access to a lower-level control signal. However, elements of control that the PID motor controller accounts for such as limiting the change in command signals or reducing accumulated error over time are also required to be learned to achieve comparable performance.

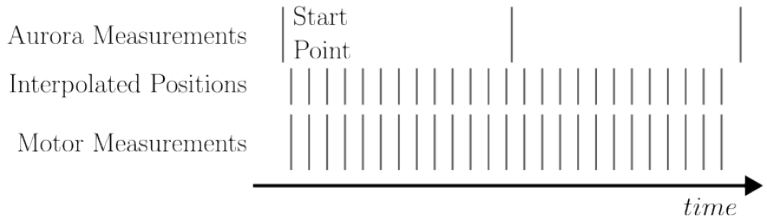
Option 2 In the second option (Figure 13b), the motor controller remains in the loop. This approach has the learned controller replace one of the two baseline controllers presented in Section 3.3. This approach may be limited in accuracy as the inclusion of a PID controller restricts the system's output.

3.5.1 Dataloaders

Two dataloaders were developed in order to provide an interface between the robot's data logs and the Pytorch deep learning library[3], one designed for learning control and the other for state estimation.



(a) Control dataloader data representation



(b) State estimation dataloader data representation

Figure 14: Representation of data loader with dataloader

The control dataloader provides as input the robot's current ground truth position, a horizon of feedback from the motor controller, and the robot's next ground truth position. Each of these inputs is available to the robot in real time during closed-loop operation. For open-loop control, the current position input should either not be used or replaced with the output of a state

[3]https://pytorch.org/

estimation module. The labels provided are the next N motor commands. Figure 14a shows the timing breakdown of each of the provided pieces of data for this dataloader.

The state estimation dataloader provides as input the robot's starting position, and a series of feedback from the motor controller. The labels provided are an interpolation of the robot's ground truth state as measured at each motor controller feedback tick. The dataloader linearly interpolates the AURORA data to provide a state vector for every motor feedback command available. Figure 14b shows the timing breakdown of each of the provided pieces of data for this dataloader.

3.6 Learning Experiments

3.6.1 Model Architecture

A base model architecture is used as a starting point for all learning experiments and is shown in Figure 15. It consists of a head, a backbone, and a state estimator. The head consists of a configurable set of linear layers using the ReLU activation function. The backbone similarly is a configurable set of linear layers using the ReLU activation function that take as input the starting and goal position, along with any configuration parameters for the given dataset. These configuration parameters are tunable values that are indexed based on which robot setup was used to collect that data. A new configuration parameter is used for each time the robot has its tendons replaced. The state estimator is a configurable set of LSTM cells that take as input a horizon of motor feedback data. Specifically, experiments use current draw, encoder position, and encoder velocity for each motor. The output of the head is passed to one last linear layer without an activation layer. The output is then reshaped to be an $N \times 2$ vector where N is the prediction horizon of the model.

3.6.2 Training Parameters

RMSE loss is used for both training and comparing validation performance on a holdout dataset for each experiment (Equation (10)).

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{N} ||y_{pred} - y_{gt}||^2}{N}} \tag{10}$$

where:

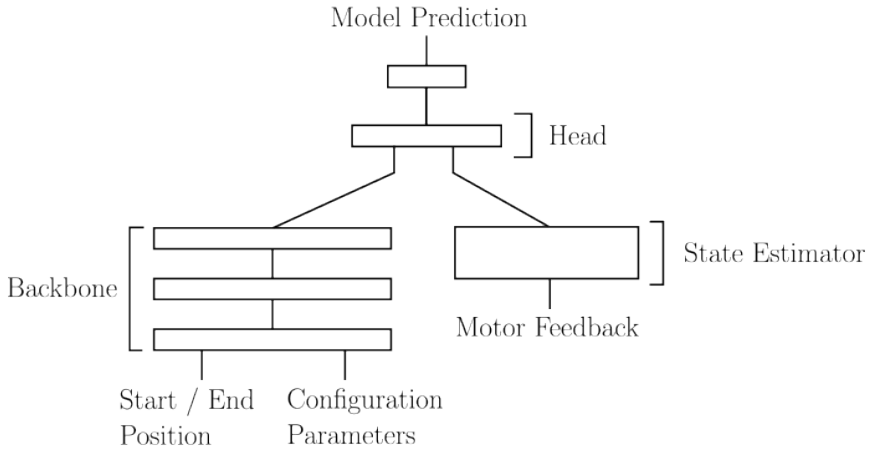


Figure 15: Base model architecture for learning-experiments

y_{pred} = Predicted label from learned model
y_{gt} = Dataset label
N = Number of items predicted

Constant training parameters described in Table 6 are set and remained constant between all experiments. A runtime rate of 1000 Hz for individual model calls is enforced for each trial to ensure that any model trained is capable of running in real time on the robot.

Variable training parameters described in Table 6 are varied through a series of experiments to find the optimal values for reducing validation loss. In each of these experiments (omitting the model architecture search trials), a baseline configuration is used as a comparison.

Table 6: Training parameters for learning experiments

Parameter	**Value**
Static Parameters	
Optimizer	Adam (default parameters)
Learning rate	0.001
Batch size	512
Epochs	1000
Early stopping	10
Variable Parameters	
Feedback horizon	1 - 300
Backbone layers	1 - 3
Backbone layer size	10 - 30
Head layers	0 - 3
Head layer size	0 - 50
LSTM layer size	0 - 40
Configuration parameter size	0 - 20
Data weights	Multiple functions

4 Results

4.1 Robot Prototype

During data collection, several kinematic limits (shown in Table 7) to the prototype became obvious. Maximum tested speeds under contraction reached 0.2 m/s while maximum speeds under extension reached 1.0 m/s.

The primary failure modes of this prototype are tendons breaking and the tendon spindle snapping. During data collection, the robot had four critical failures where a piece on the physical system broke. Twice the tendons snapped and twice the spindle sheared (Figure 16).

4.2 Baseline Controllers

Both the differential CC and PID baseline controllers were tested on three tasks: following a path, following a trajectory, and repeated the same motion. The results for each are presented

Table 7: Prototype physical limits

Limit	**Value**
Top contraction speed	$0.2\,\mathrm{m/s}$
Top extension speed	$1.0\,\mathrm{m/s}$
Maximum curvature	$7.39\,\mathrm{m}^{-1}$
Failure rates	$\mu = 83.2\,\mathrm{min}$
	$\sigma = 48.9\,\mathrm{min}$

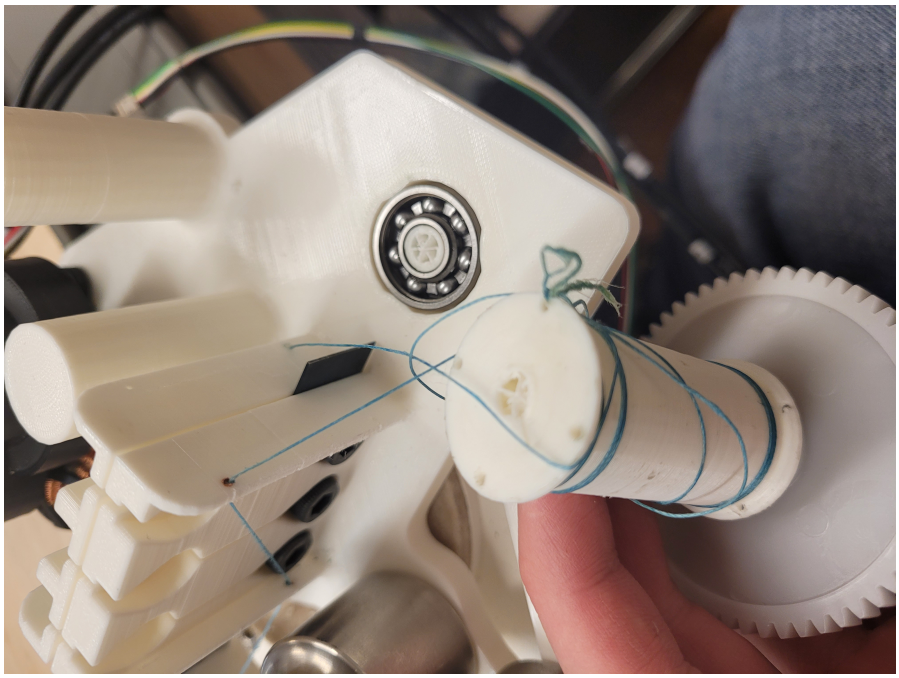


Figure 16: Tendon spindle shear failure

in the following sections. The runtime of each controller is presented as well.

4.2.1 Path Tracking

Both baseline controllers were asked to follow five test paths. Each path is discretized into 50 way points. The controllers are required to measure an end effector position within 2 cm of each way point before progressing to the next. The paths traced by each controller is shown in Figures 17 and 18. Table 8 highlights key metrics from each run including the total distance travelled by the robot and the time it took to complete each path. This test was repeated three times and the average results across all three trials are displayed.

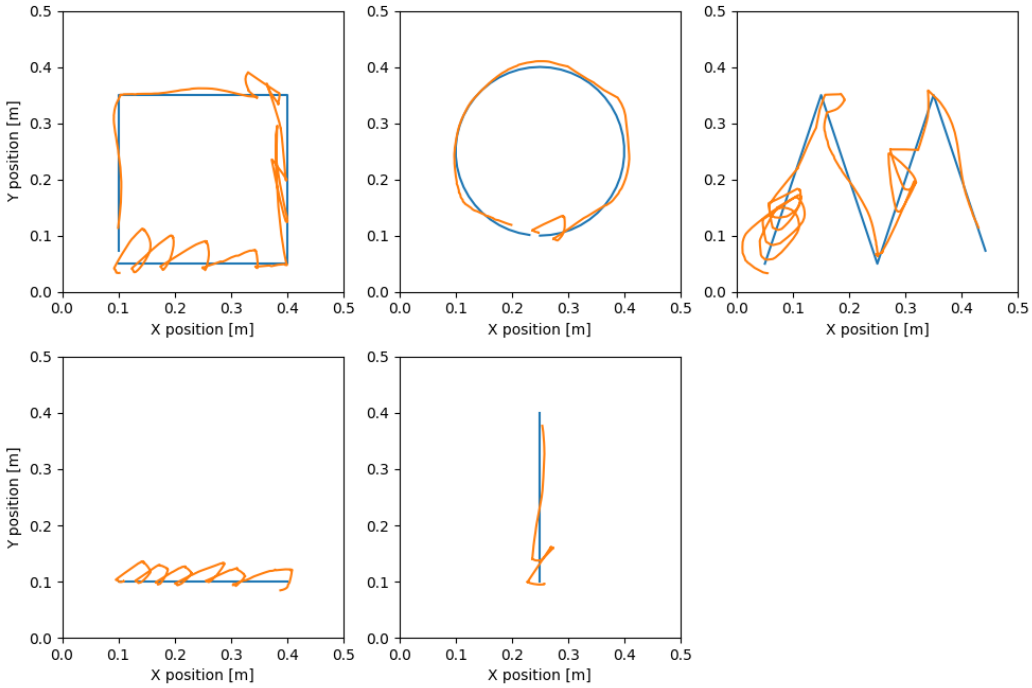


Figure 17: Differential CC controller point tracking path (orange) with reference path (blue)

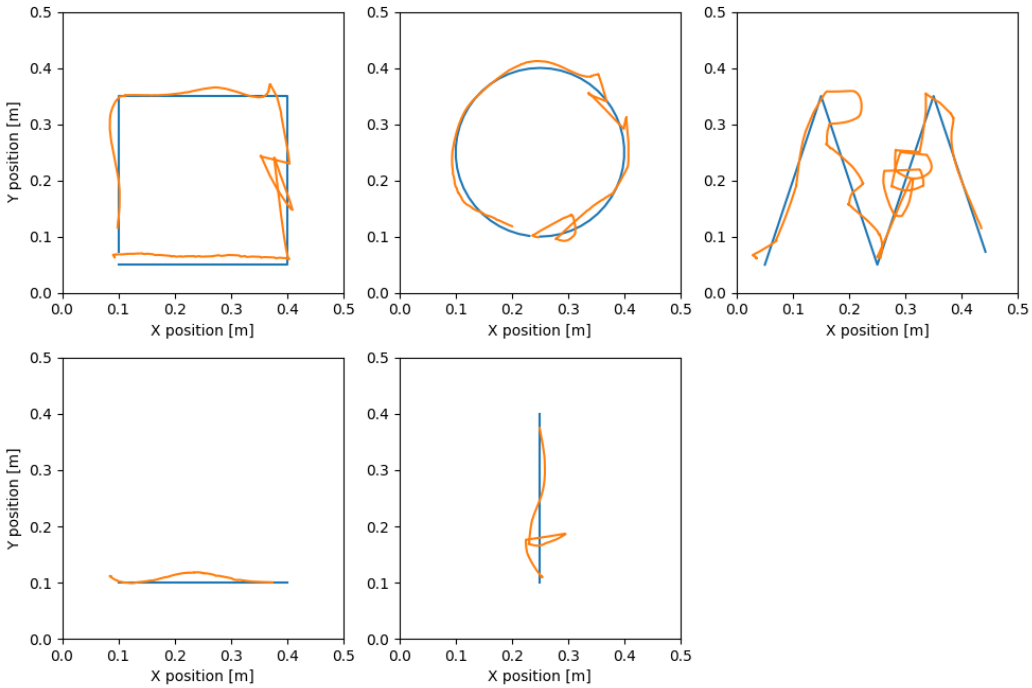


Figure 18: PID controller point tracking path (orange) with reference path (blue)

Table 8: Baseline controller point tracking trial summary

Trial Shape	**Run time [s]**	**Path Length [m]**	**Error**
Reference Trajectory			
Square	N/A	1.178	N/A
Circle	N/A	0.923	N/A
Zig Zag	N/A	1.241	N/A
Horizontal Line	N/A	0.300	N/A
Vertical Line	N/A	0.300	N/A
Differential CC Controller			
Square	136.36	2.190	85.9%
Circle	61.48	1.048	13.5%
Zig Zag	152.21	2.551	105.6%
Horizontal Line	105.48	0.817	172.3%
Vertical Line	34.03	0.411	37.0%
PID Controller			
Square	74.92	1.454	23.4%
Circle	87.58	1.197	29.7%
Zig Zag	127.87	1.988	60.2%
Horizontal Line	19.01	0.302	0.6%
Vertical Line	30.10	0.435	45%

4.2.2 Trajectory Tracking

Both baseline controllers were asked to execute five test trajectories in a set amount of time. Each trajectory is discretized into 50 way points with 0.2 s in between each. The robot is controlled to follow the way point as it progresses through each of the discretized points. The paths traced by each controller is shown in Figures 19 and 20. Table 9 highlights key metrics from each run including the total distance travelled by the robot and the RMSE between the robot path and the test trajectory.

4.2.3 Task Repetition

Controllers were tested on their ability to repeat the same task consistently. The trial was run from home position to the point (0.15 m, 0.2 m). It was repeated 10 times for each controller. The paths taken by each controller are shown in Figure 21. The maximal Fréchet distance [32] is calculated between each of the 10 trials for both controllers as a metric of controller consistency.

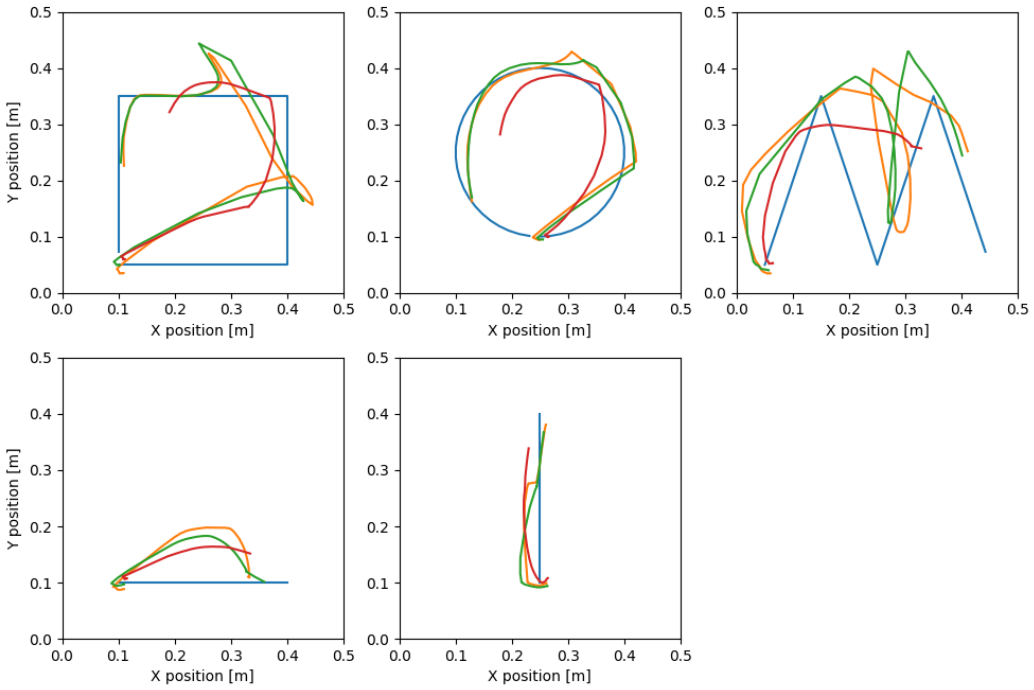


Figure 19: Differential CC controller trajectory tracking paths (orange, green, red) with reference path (blue)

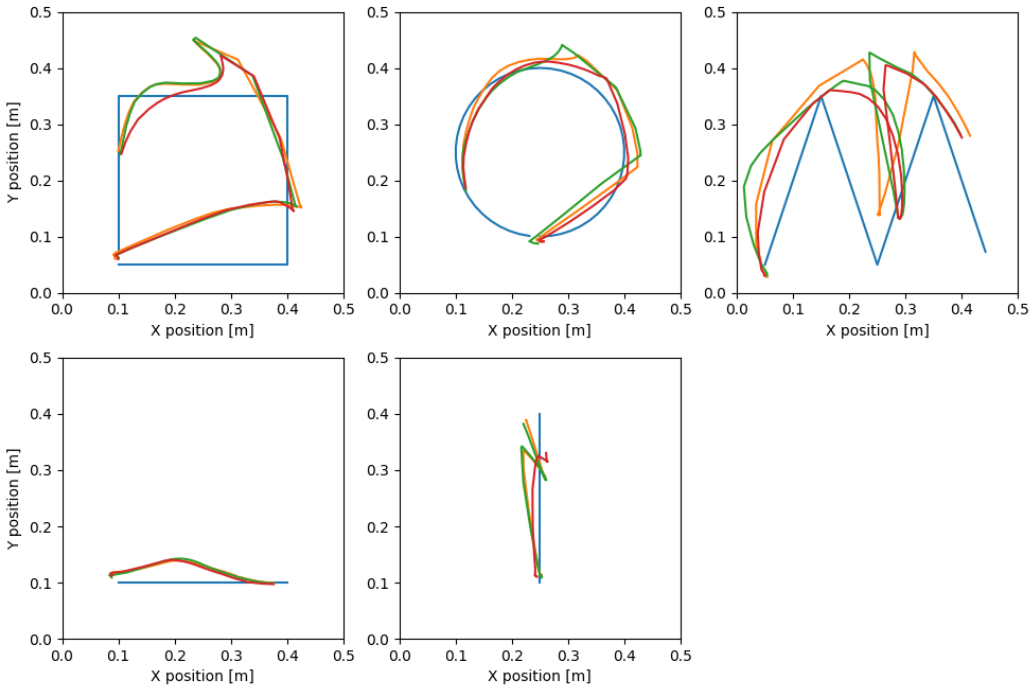


Figure 20: PID controller trajectory tracking paths (orange, green, red) with reference path (blue)

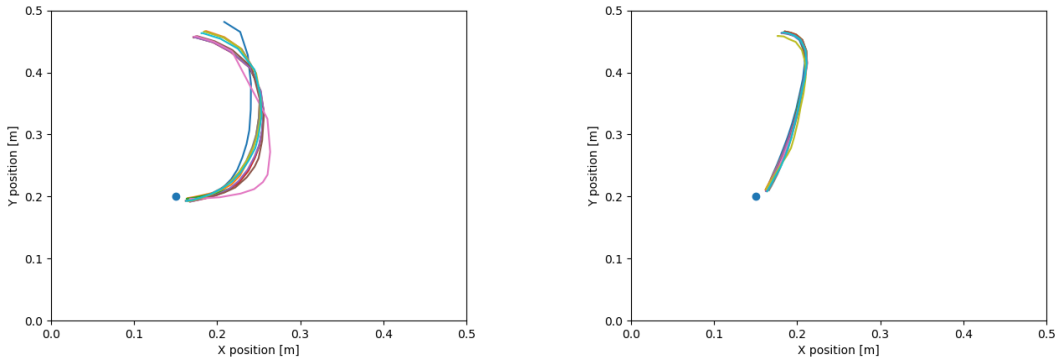
Table 9: Baseline controller trajectory tracking trial summary

Trial Shape	**Run time [s]**	**Path Length [m]**	**RMSE [m]**
Reference Trajectory			
Square	10.80	1.178	N/A
Circle	10.00	0.923	N/A
Zig Zag	10.40	1.241	N/A
Horizontal Line	10.20	0.300	N/A
Vertical Line	10.20	0.300	N/A
Differential CC Controller			
Square	12.25	0.959	0.144
Circle	11.34	0.748	0.137
Zig Zag	11.80	1.008	0.147
Horizontal Line	11.57	0.324	0.072
Vertical Line	11.57	0.305	0.055
PID Controller			
Square	12.22	1.031	0.131
Circle	11.31	0.833	0.124
Zig Zag	11.77	11.213	0.150
Horizontal Line	11.54	0.306	0.043
Vertical Line	11.54	0.360	0.054

The differential CC controller has a maximal Fréchet distance of 0.102 m while the PID controller has a maximal Fréchet distance of 0.027 m, indicating a much tighter distribution. This result is confirmed visually in Figure 21.

4.2.4 Timing Analysis

The runtime of four different necessary controller functions was studied for both baseline controllers and the best performing learned controller. This includes the controller constructor method, the command retrieval, the end point update, and the goal point update. Table 10 displays the results. A "per iteration" result is returned as well, which is a sum of the three functions that are required to run at each control loop.



(a) Differential CC controller task repetition

(b) PID controller task repetition

Figure 21: Baseline controller repeated motion trial paths. The target point is shown in blue.

4.3 Data Set

Over a span of three days, data was collected using the methods outlined in Section 3.4. A total runtime of 5.78 h was collected. The robot traveled 472 m and broke down four times.

4.3.1 Distributions

Presented in this section is both the task space and joint space distributions of the dataset. In task space, the position, velocity, and acceleration of the end effector are displayed in the histogram in Figure 23a. In joint space, the motor current is displayed along with the motor current's first and second derivatives in Figure 23b.

Figure 22 shows the task space position distribution overlaid on the robot's workspace. The densest distribution point at the middle-bottom of the workspace corresponds with the robot's home position. This is where the robot starts each time it is powered on and where it returns to in recovery mode.

Table 11 breaks down the sample termination cases for the dataset, along with relevant metrics for each. Note that all cases that ended in an error termination were manually removed due to data corruption.

4.4 Deep-Learning Controller

4.4.1 Model Architectures

A number of model architectures were trained in a coarse to fine grid sweep to determine an optimal learning model baseline for future experiments. A summary of the trial results is displayed

Table 10: Controller run times

Function	**Run time [μs]**
Differential CC Controller	
Current command retrieval	0.1
End point update	364.2
Goal point update	0.2
Total per iteration	**364.5**
Controller object construction	11.4
PID Controller	
Current command retrieval	0.1
End point update	24.5
Goal point update	0.8
Total per iteration	**25.4**
Controller object construction	211.0
Learning-Based Controller	
Current command retrieval	0.2
End point update	356.3
Goal point update	0.2
Total per iteration	**356.7**
Controller object construction	1,184.3

in Table 12. Only the best architecture from each trial is shown. For a full breakdown of each trial, see Appendix A. In many trials a clear winner is not found, and changing the parameter has little to no effect on the model performance. In these cases, the baseline model selection criteria favours smaller parameters to enable faster run times.

The baseline configuration chosen for future trials contains a backbone with two fully connected layers of 10 nodes each, an LSTM state estimator with 30 nodes, and a single fully connected head layer with 100 nodes. Other baseline parameters include: 512 batch size, 1000 epochs, 0.001 learning rate, a 0.2 s feedback horizon, and a 0.2 s prediction horizon.

4.4.2 Feedback Horizon

Figure 24 shows the model search for an optimal feedback horizon length. The model used for this trial is the baseline model described in Section 3.6.2. The feedback horizon was varied from 0 s (no feedback data is provided) to 15 s. The best validation loss achieved from a model trained

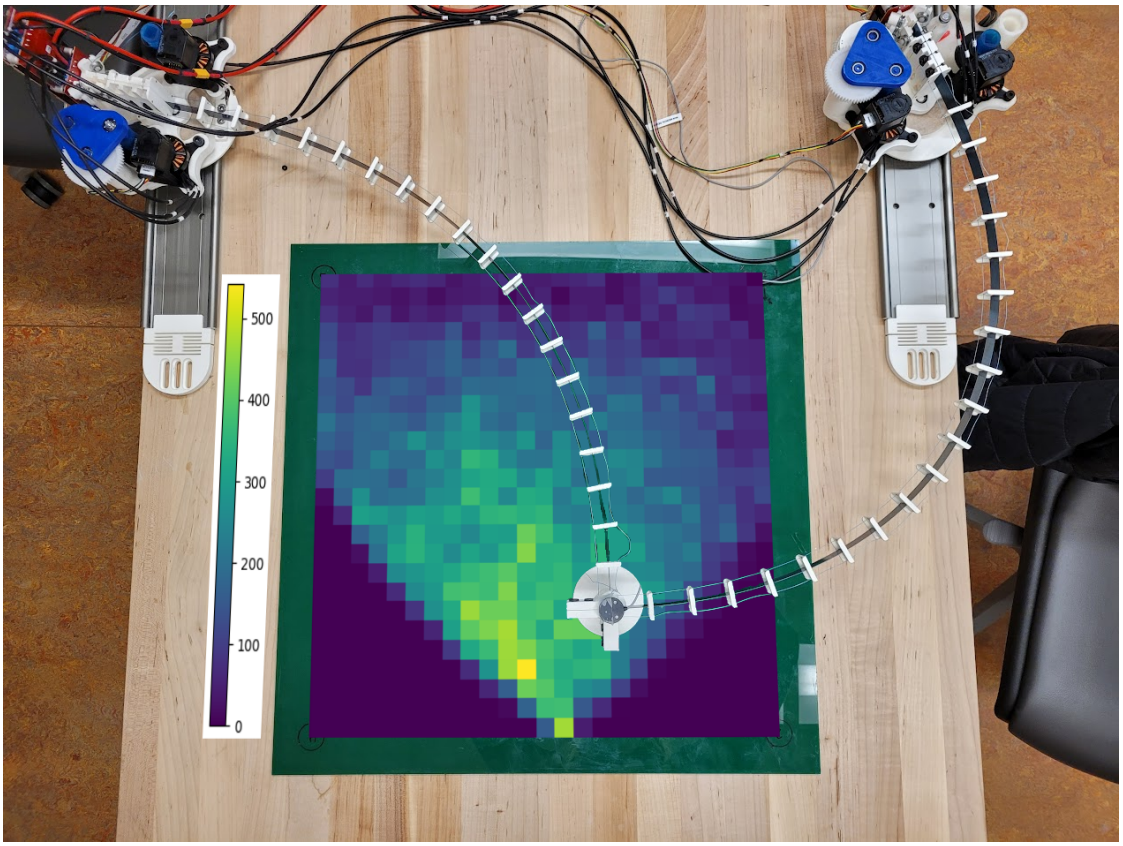


Figure 22: Data set position distribution overlaid on robot workspace

using each given feedback horizon is presented.

4.4.3 Configuration Parameters

Figure 25 shows the model search for an optimal configuration parameter size. The model used for this trial is the baseline model described in Section 3.6.2. The configuration parameter size was varied from 0 (no configuration parameter is provided) to 20. The configuration parameter size refers to a set of tunable values passed as an input to the model where each set of values is unique to the robot configuration that the data was collected on. Each time the robot broke, a new configuration is labelled.

4.4.4 Data Weighting

The data weighing trial compared model performance when trained using weighted prediction labels. Three weight distributions were tested alongside an unweighted baseline. Each weight function was trained with six different model architectures. The results of this trial are shown in Figure 26. The weight for each trials are sampled from a continuous function defined on the

Table 11: Data set run type summary

Data Run Type	**Average Distance [m]**	**Average Velocity [m/s]**	**Average Duration [s]**	**Total Distance [m]**	**Total Duration [s]**	**Total Runs**
Success	0.1038	0.0294	4.49	367.78	15,915	3543
Stalled	0.1142	0.0184	6.55	73.29	4,203	642
Aurora Lost	0.1281	0.0338	3.65	1.79	51	14
Recovery	0.3962	0.0591	6.69	22.77	401	60
Mode Switched	0.1157	0.0322	4.66	5.44	219	47
User Terminated	0.0811	0.0217	3.42	0.16	7	2

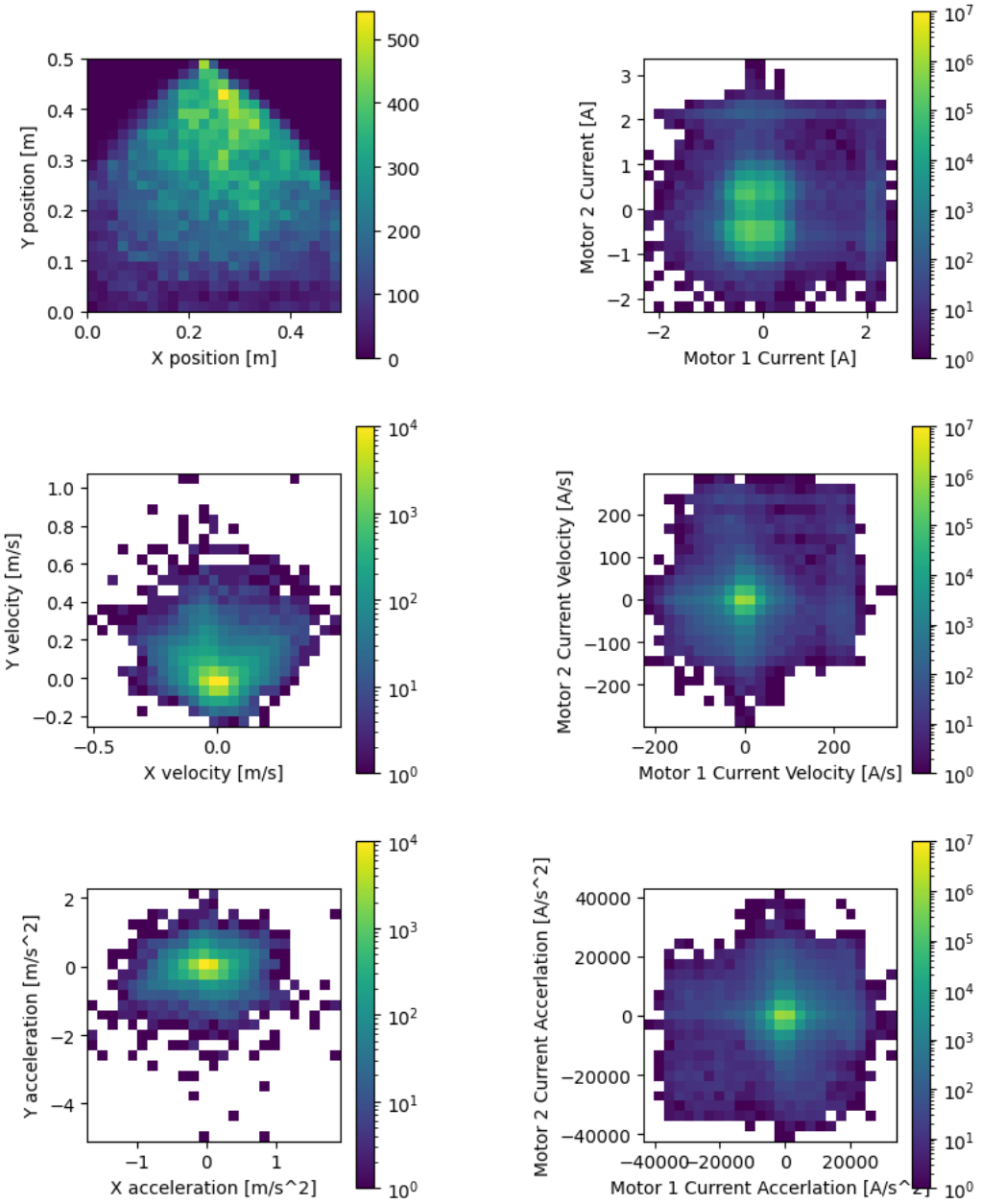
Table 12: Model architecture search trial summary

Trial Type	**Backbone Linear Layer Size(s)**	**Head Linear Layer Size(s)**	**LSTM Layer Size**	**Best RMSE Validation Loss [A]**
LSTM layer size search	10, 10	50	15	0.0220
Head layer size search	10, 10	0	20	0.0226
Head layer depth search	10, 10	25, 25	20	0.0224
Backbone layer size search	30	25, 25	20	0.0205
Backbone layer depth search	20, 20	25, 25	15	0.0221

prediction interval. Each set of weights is normalized so that the sum of all weights equals one. The functions used are defined in Table 13.

Table 13: Data weight definitions for data weighing trial

Weight Class	**Function (Pre-Normalization)**	**Interval**
Unweighted	$f(t) = 1$	N/A
Linear	$f(t) = \frac{t}{3}$	$[1, 3)$
Exponential	$f(t) = e^t$	$[0, 2)$
Quadratic	$f(t) = t^2 - 1.5t + 1.5$	$[0, 2)$



(a) Task space distribution

(b) Joint space distribution

Figure 23: Data set distribution histograms

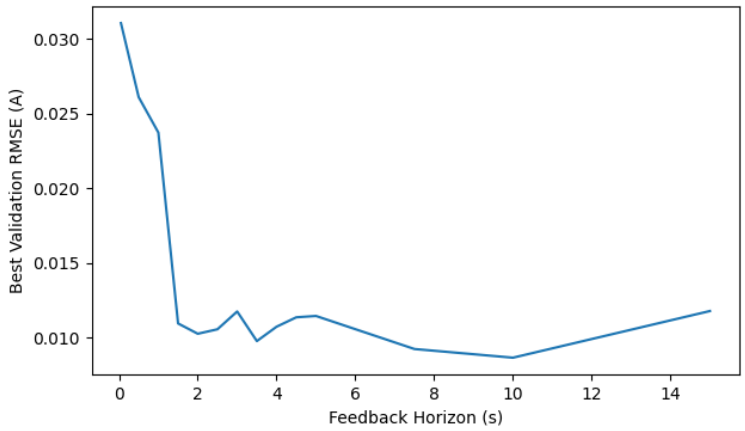


Figure 24: Best validation losses in feedback horizon trial

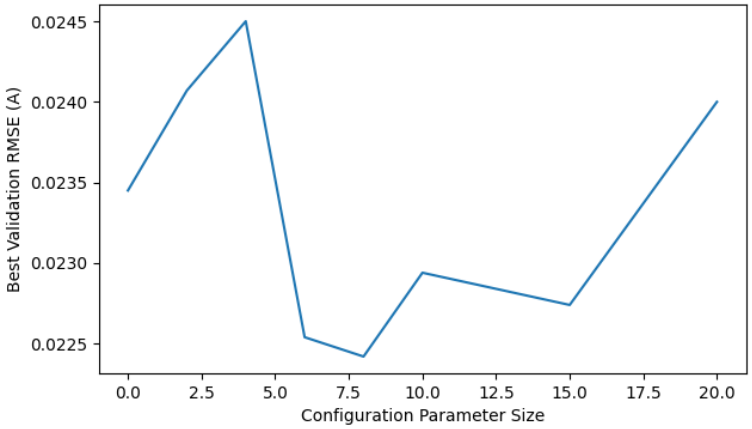


Figure 25: Best validation losses in configuration parameter trial

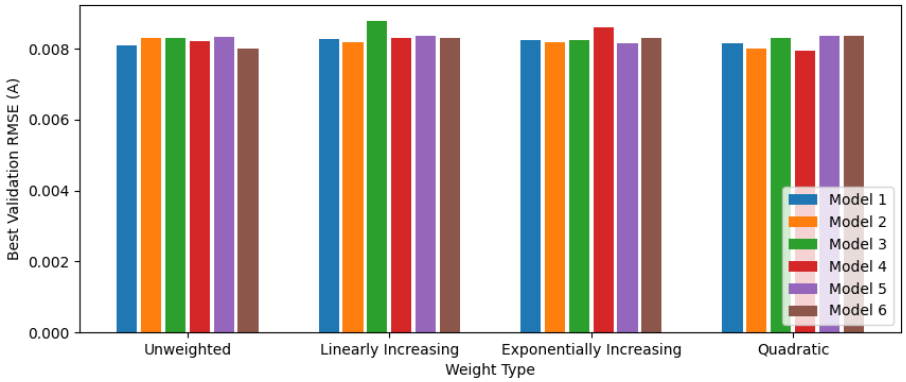


Figure 26: Best validation losses in prediction weighting trial

5 Discussion

5.1 Physical Prototype

5.1.1 Use Cases

The proposed robot does not have any intended direct applications in industry. The configuration of a planar PCR of this scale, with no ability to separate at the end effector has very little application in most contexts. It does however provide a significant use case in research of PCR control. Often PCRs that are researched intentionally have no external forces applied to their end effectors, they are operating in free space [2]. In this free configuration, the only force that control and estimation models must account for is gravity, which is related to the robot's mass. In the proposed setup, a friction force is applied to the end effector, which is related to the robot's velocity. For this component to be modelled and accounted for, researchers will be required to leave behind the semi-static assumption during modelling. The proposed platform thus provides an opportunity to expand PCR modelling in research to include a time component.

5.1.2 Robot Configurations

The robot is designed both in software and hardware to accommodate changing robot configurations without a large hassle. The arm's bases are mounted on a rail that can easily be translated along any axis that the rails are fastened in. For the setup discussed in this thesis, the bases can be moved along the y-axis. Additionally, the software and control loop supports the addition of arms. A third arm could be added for example to try and gain an angular degree of freedom with the end effector. In Figure 27, several possible workspace configurations with just the current setup are shown. Each would take less than a minute to change to. The addition of more arms can only shrink the usable workspace. The controllers presented can easily be adjusted to account for additional arms as well. The PID controller requires no modification while the differential CC controller would required the left inverse be taken in (4).

The addition of one or multiple arms can add complexity of control by allowing the user to apply arbitrary forces at the end effector at the expense of limiting the robot's workspace. A hypothetical configuration could see four arms operating; two that are used to control the robot and two that apply arbitrary end effector forces. Because of the tilting end effector issue mentioned in 3.1.1, the physical limitation of the current end effector is likely around four arms without a redesign.

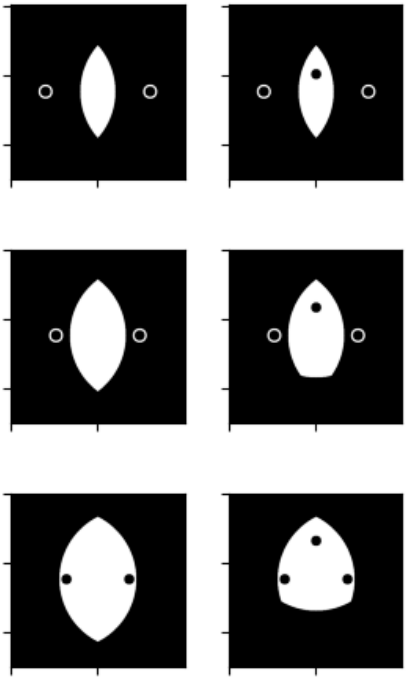


Figure 27: Simulated robot workspace with two (left) and three (right) arms in varying planar configurations

5.1.3 Limitations

Tendon slack arises on each arm due to manufacturing inconsistencies during the tendon tensioning process. To ensure the tendon stays routed the right way, four 0.75" nuts are placed on it to act as a tensioning weight (Figure 5). This slight amount of slack results in an operational "dead zone" between arm contraction and extension where the robot can actuate the motors with no resulting motion in the end effector. The idea of training learning-based models that have a tunable configuration parameter comes from this effect, as the robot will have a slightly different response to control inputs in this region between tendon installations.

Physical failures coming from 3D printed parts failing is a reality of ensuring this design is reproducible for third parties. Replacement parts machined out of stronger materials could greatly improve the lifespan of the robot but are far less accessible. This trade off is a reality of this design that results in limited consistency of the robot. This may make the platform unsuitable to some research applications.

5.2 Baseline Controllers

5.2.1 Overview

The baseline controllers proposed in Section 3.3 do not have good tracking capabilities. For even the simplest tracked trajectory the controllers were only able to achieve an RMSE of 0.043 m. This equates to a 5.3 % error at the end effector. Compared to the controllers presented in [16], [17], [26] which achieved relative errors of 0.4 %, 2.5 %, and 0.61 % - 0.76 % respectively, this is worse. Comparisons made between these proposals though must be taken with a grain of salt as there exists vast differences between the robot's used in each work. The slightly lower performance on this system could be due to several reasons, including less time spent tuning controller gains or system errors that result in larger discrepancies.

Both of the proposed baseline models achieved per iteration run times of faster than the 1000 Hz run time constraint, with the PID controller achieving the fastest average runtime. The differential CC controller sees a slower end point update step because of its requirement to numerically solve for the robot's curvature at each motor feedback step. Even the learning-based controller, even though it did not work, had run times that satisfied the 1000 Hz run time constraint.

5.2.2 Significance

The baseline models proposed here can only be used for motion of the robot in closed-loop scenarios, an application space that is hardly seen in the real world. Figure 21 demonstrates that even in the simple task of moving in a straight line, these controllers fall short. The static and dynamic effects of the system result in the addition of non-linear path errors when operating in task space. To counteract these errors, more complex modeling beyond these baseline's capabilities must be employed. As such, the proposed controllers do not contain much value beyond acting as a reference for future work and a motivating example for why learning-based solutions may be the optimal path forward for control of CRs.

5.2.3 Limitations

High control error leaves the baseline controllers little to be desired in most conventional control tasks. They struggle with small motions, unable to account for the spring force of the beam. More work could be done to improve the gains or runtime of the proposed controllers. This may result in marginal performance improvement, but they still fundamentally lack the ability to model the complex static and dynamic effects of this system. There is an upper bound to how well these baselines will be able to control the robot.

No convergence guarantee is given for the differential CC controller. This has not caused issues in practice on the machine used for this thesis, but is more likely if the controller is run on slower systems that reduce the solver runtime to ensure real time operation. Future work could explore theoretical guarantees of stability for each of these controllers.

The Python implementation of this entire control pipeline could have severe consequences for real time operation. In addition to having slower run times, Python does not have real-time capability. Python was the language of choice for this work primarily because of its fast development times compared to compiled languages such as C++. Lacking real-time support added additional runtime constraints and threading management to the code that would not be required if it were written in another language. For all of the items proposed in this thesis, Python was not a limiting factor. If future work looks to implement more complex algorithms or on slower systems, the language choice could become a large detractor of the project.

5.3 Learning Pipeline

5.3.1 Learning Tasks

Several behaviours were witnessed throughout data collection that pose interesting learning challenges for users of this dataset. By only providing the end effector position, no information is explicitly provided about the curvature of the robot. Cases arise where the robot can take on different shapes given the same end effector position by making use of the external forces applied on the end effector. An example of this is shown in Figure 28. The two arms shown would have different dynamic behaviour during operation and so it may be valuable to have systems that have state representations larger than just an end effector position. This idea is what motivated the use of a larger LSTM layer in the learning-based controller state estimation sub-model.

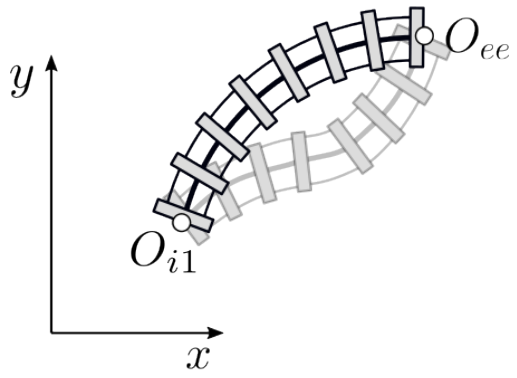


Figure 28: Example of two robot configurations with equal tendon displacements

Another behaviour relating to the tendon slack issue discussed in Section 5.1.3 is exacerbated in the dataset because of the beam spring effect. When the robot's arm is being contracted, a restoring force acts on the end effector. If this restorative force is large enough, when the robot switches between contracting and extension the end effector will jump slightly to achieve tension in the tendon on the other side of the arm. For this to happen, the restorative force must be great enough to overcome static friction between the end effector and the table. This behaviour can be very sudden and is what results in the significantly higher maximum velocity during robot extension. When the restoring force is not great enough to overcome static friction, a period of time will pass where the motor is actuated with no end effector motion. There are only certain states that allow for this stationary behaviour to occur and it is not clear how to explicitly find them, they are not solely related to end effector position. Figure 29 shows one such equilibrium state.

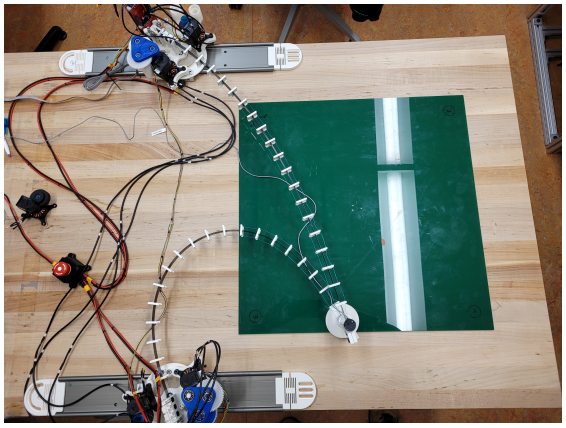


Figure 29: Example of an equilibrium position where the robot will remain stationary despite motor actuation

5.3.2 Expected Impact

Learning-based approaches have the potential to model arbitrarily complex dynamic systems. Using this dataset, the expectation is that learning-based controllers will greatly improve upon the baseline accuracy. This is especially true when the model is moving fast or bending far, where the baseline model assumptions are less accurate. These models will be able to more accurately estimate the system kinematics and dynamics, enabling real-time tracking in task space for this robot.

While there does not exist a direct comparison in the literature to the proposed parallel planar system, several references suggest that accuracy results in the range of 0.5-2% of the

robot's length could be expected with further controller development [15]–[17]. As this is the first step towards learning-based control being proposed for a PCR, accuracy comparable with current state-of-the-art for other CRs is only useful to provide a ballpark of what is possible.

5.4 Open Continuum Robotics

With the rise in popularity of continuum robots, the Open Continuum Robotics project was created to reduce barriers to entry into the field [29]. The work of this thesis will be open sourced and aims to contribute to the Open CR project. Throughout the design process, keeping the robot easy to reproduce and repair influenced the design to enable third parties to recreate the setup. All parts are either off-the-shelf or 3D printed, with all firmware being open-source. This work helps bridge the gap between CRs and machine learning while removing barriers to access for other researchers to explore this problem.

6 Conclusion

This thesis lays the foundation for future work in learning-based control for a planar parallel TDCR. A case is made for why learning-based control has a strong application in complex physical systems such as in a planar parallel TDCR. A robot prototype is designed, built, and validated as a research platform. It is open sourced with the intent of enabling easier entry into the field for other researchers while providing a platform that can be used for consistent result comparison. Two baseline controllers are proposed as an initial comparison point on this robot. A dataset is produced with data for learning both control and state estimation on the proposed prototype. All of this constitutes necessary steps towards improving research on learning-based applications for TDCRs, opening the door for future work on the topic.

6.1 Future Work

6.1.1 State Estimation

Both proposed models in this work assumes end effector positional feedback is available to the user. In general, this is not the case and adds the additional requirement of having a motion capture system set up in order to use the robot. This is the largest barrier to reproduce this robot for use in research or other projects. Learning a model that explicitly performs state estimation was deemed outside of the scope of this thesis but is a required component to further improve the usefulness of this model. Demonstrated state estimation for CRs include both non-learning-based methods and learning-based methods [15]. The proposed dataset enables furthering this research in learning to include state estimation of PCRs. If can also provide a reference set for non-learning-based state estimation pipelines.

6.1.2 Learning-Based Control

Successful learning-based control was not achieved in this project. Several negative experiment results were shown but none were able to sufficiently control the physical robot. This is the natural continuation of this work. This thesis provides all the necessary foundation to begin this learning-based control research including a robot prototype, a dataset, several baseline models for comparison, and negative result trials to inform future work. Learning-based control for CRs and PCRs remains an underdeveloped area and the hope of this thesis is to provide a foundation for this application.

References

[1] R. J. Webster, A. M. Okamura, and N. J. Cowan, "Toward active cannulas: Miniature snake-like surgical robots," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006, pp. 2857–2863. DOI: 10.1109/IROS.2006.282073.

[2] S. Lilge, K. Nuelle, J. A. Childs, K. Wen, C. Rucker, and J. Burgner-Kahrs, "Parallel continuum robots: A survey," In Revision.

[3] J. Burgner-Kahrs, D. C. Rucker, and H. Choset, "Continuum robots for medical applications: A survey," *IEEE Transactions on Robotics*, vol. 31, no. 6, pp. 1261–1280, 2015. DOI: 10.1109/TRO.2015.2489500.

[4] X. Dong, D. Axinte, D. Palmer, S. Cobos, M. Raffles, A. Rabani, and J. Kell, "Development of a slender continuum robotic system for on-wing inspection/repair of gas turbine engines," *Robotics and Computer-Integrated Manufacturing*, vol. 44, pp. 218–229, 2017, ISSN: 0736-5845. DOI: https://doi.org/10.1016/j.rcim.2016.09.004. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0736584516300801.

[5] C. E. Bryson and D. C. Rucker, "Toward parallel continuum manipulators," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 778–785. DOI: 10.1109/ICRA.2014.6906943.

[6] P. Rao, Q. Peyron, S. Lilge, and J. Burgner-Kahrs, "How to model tendon-driven continuum robots and benchmark modelling performance," *Frontiers in Robotics and AI*, vol. 7, 2021, ISSN: 2296-9144. DOI: 10.3389/frobt.2020.630245. [Online]. Available: https://www.frontiersin.org/articles/10.3389/frobt.2020.630245.

[7] K. Nuelle, T. Sterneck, S. Lilge, D. Xiong, J. Burgner-Kahrs, and T. Ortmaier, "Modeling, calibration, and evaluation of a tendon-actuated planar parallel continuum robot," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5811–5818, 2020. DOI: 10.1109/LRA.2020.3010213.

[8] S. Lilge, K. Nülle, G. Boettcher, S. Spindeldreier, and J. Burgner-Kahrs, "Tendon actuated continuous structures in planar parallel robots: A kinematic analysis," *Journal of Mechanisms and Robotics*, vol. 13, Nov. 2020. DOI: 10.1115/1.4049058.

[9] A. I. Károly, P. Galambos, J. Kuti, and I. J. Rudas, "Deep learning in robotics: Survey on model structures and training strategies," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 1, pp. 266–279, 2021. DOI: 10.1109/TSMC.2020.3018325.

[10] X.-B. Jin, R. J. Robert Jeremiah, T.-L. Su, Y.-T. Bai, and J.-L. Kong, "The new trend of state estimation: From model-driven to hybrid-driven methods," *Sensors*, vol. 21, no. 6, 2021, ISSN: 1424-8220. DOI: 10.3390/s21062085. [Online]. Available: https://www.mdpi.com/1424-8220/21/6/2085.

[11] M. Yan, Y. Zhu, N. Jin, and J. Bohg, "Self-supervised learning of state estimation for manipulating deformable linear objects," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2372–2379, 2020. DOI: 10.1109/LRA.2020.2969931.

[12] M. K. Al-Sharman, Y. Zweiri, M. A. K. Jaradat, R. Al-Husari, D. Gan, and L. D. Seneviratne, "Deep-learning-based neural network training for state estimation enhancement: Application to attitude estimation," *IEEE Transactions on Instrumentation and Measurement*, vol. 69, no. 1, pp. 24–34, 2020. DOI: 10.1109/TIM.2019.2895495.

[13] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger, "Learning-based model predictive control: Toward safe learning in control," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, no. 1, pp. 269–296, 2020. DOI: 10.1146/annurev-control-090419-075625. eprint: https://doi.org/10.1146/annurev-control-090419-075625. [Online]. Available: https://doi.org/10.1146/annurev-control-090419-075625.

[14] J. S. Toquica, P. S. Oliveira, W. S. Souza, J. M. S. Motta, and D. L. Borges, "An analytical and a deep learning model for solving the inverse kinematic problem of an industrial parallel robot," *Computers & Industrial Engineering*, vol. 151, p. 106 682, 2021, ISSN: 0360-8352. DOI: https://doi.org/10.1016/j.cie.2020.106682. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360835220304162.

[15] X. Wang, Y. Li, and K.-W. Kwok, "A survey for machine learning-based control of continuum robots," *Frontiers in Robotics and AI*, vol. 8, 2021, ISSN: 2296-9144. DOI: 10.3389/frobt.2021.730330. [Online]. Available: https://www.frontiersin.org/articles/10.3389/frobt.2021.730330.

[16] R. M. Grassmann, R. Z. Chen, N. Liang, and J. Burgner-Kahrs, "A dataset and benchmark for learning the kinematics of concentric tube continuum robots," in *Workshop on Learning from Diverse, Offline Data*, 2022. [Online]. Available: https://openreview.net/forum?id=DW9uz_GZ0og.

[17] M. Giorelli, F. Renda, M. Calisti, A. Arienti, G. Ferri, and C. Laschi, "Neural network and jacobian method for solving the inverse statics of a cable-driven soft arm with nonconstant curvature," *IEEE Transactions on Robotics*, vol. 31, no. 4, pp. 823–834, 2015. DOI: 10.1109/TRO.2015.2428511.

[18] J. Zhang, Q. Fang, P. Xiang, D. Sun, Y. Xue, R. Jin, K. Qiu, R. Xiong, Y. Wang, and H. Lu, "A survey on design, actuation, modeling, and control of continuum robot," *Cyborg and Bionic Systems*, vol. 2022, 2022. DOI: 10.34133/2022/9754697. eprint: https://spj.science.org/doi/pdf/10.34133/2022/9754697. [Online]. Available: https://spj.science.org/doi/abs/10.34133/2022/9754697.

[19] *Types of continuum robots*. [Online]. Available: https://www.cs.toronto.edu/~jbk/opencontinuumrobotics/101/2022/11/04/cr-types.html.

[20] A. Hemami, "Studies on a light weight and flexible robot manipulator," *Robotics*, vol. 1, no. 1, pp. 27–36, 1985, ISSN: 0167-8493. DOI: https://doi.org/10.1016/S0167-8493(85)90306-7. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167849385903067.

[21] I. Robert J. Webster and B. A. Jones, "Design and kinematic modeling of constant curvature continuum robots: A review," *The International Journal of Robotics Research*, vol. 29, no. 13, pp. 1661–1683, 2010. DOI: 10.1177/0278364910368147. eprint: https://doi.org/10.1177/0278364910368147. [Online]. Available: https://doi.org/10.1177/0278364910368147.

[22] D. Cao and R. W. Tucker, "Nonlinear dynamics of elastic rods using the cosserat theory: Modelling and simulation," *International Journal of Solids and Structures*, vol. 45, no. 2, pp. 460–477, 2008, ISSN: 0020-7683. DOI: https://doi.org/10.1016/j.ijsolstr.2007.08.016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020768307003253.

[23] A. A. Alqumsan, S. Khoo, and M. Norton, "Robust control of continuum robots using cosserat rod theory," *Mechanism and Machine Theory*, vol. 131, pp. 48–61, 2019, ISSN: 0094-114X. DOI: https://doi.org/10.1016/j.mechmachtheory.2018.09.011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0094114X18311777.

[24] S. M. H. Sadati, S. E. Naghibi, A. Shiva, I. D. Walker, K. Althoefer, and T. Nanayakkara, "Mechanics of continuum manipulators, a comparative study of five methods with experiments," in *Towards Autonomous Robotic Systems*, Y. Gao, S. Fallah, Y. Jin, and C. Lekakou, Eds., Cham: Springer International Publishing, 2017, pp. 686–702, ISBN: 978-3-319-64107-2.

[25] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989, ISSN: 0893-6080. DOI: https://doi.org/10.1016/0893-6080(89)90020-8. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0893608089900208.

[26] M. Li, R. Kang, D. T. Branson, and J. S. Dai, "Model-free control for continuum robots based on an adaptive kalman filter," *IEEE/ASME Transactions on Mechatronics*, vol. 23, no. 1, pp. 286–297, 2018. DOI: 10.1109/TMECH.2017.2775663.

[27] W. Xu, J. Chen, H. Y. Lau, and H. Ren, "Data-driven methods towards learning the highly nonlinear inverse kinematics of tendon-driven surgical manipulators," *The International Journal of Medical Robotics and Computer Assisted Surgery*, vol. 13, no. 3, e1774, 2017, e1774 RCS-16-0056.R1. DOI: https://doi.org/10.1002/rcs.1774. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/rcs.1774. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rcs.1774.

[28] A. Loquercio, M. Segu, and D. Scaramuzza, "A general framework for uncertainty estimation in deep learning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3153–3160, 2020. DOI: 10.1109/LRA.2020.2974682.

[29] *Open continuum robotics project*. [Online]. Available: https://www.cs.toronto.edu/~jbk/opencontinuumrobotics/.

[30] F. Grimminger, A. Meduri, M. Khadiv, J. Viereck, M. Wuthrich, M. Naveau, V. Berenz, S. Heim, F. Widmaier, T. Flayols, J. Fiene, A. Badri-Sprowitz, and L. Righetti, "An open torque-controlled modular robot architecture for legged locomotion research," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3650–3657, Apr. 2020. DOI: 10.1109/lra.2020.2976639. [Online]. Available: https://doi.org/10.1109%2Flra.2020.2976639.

[31] R. Grassmann, C. Shentu, T. Hamoda, P. T. Dewi, and J. Burgner-Kahrs, "Open continuum robotics – one actuation module to create them all," In Submission.

[32] *Fréchet distance*, [Online; accessed 20-April-2023], 2023. [Online]. Available: https://en.wikipedia.org/wiki/Fr%5C%C3%5C%A9chet_distance.

[33] K. Perlin, "Making noise," GDC Talk, 1999.

[34] R. M. Grassmann and J. Burgner-Kahrs, "Quaternion-based smooth trajectory generator for via poses in se(3) considering kinematic limits in cartesian space," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4192–4199, 2019. DOI: 10.1109/LRA.2019.2931133.

A Deep-Learning Trial Details

Trials were run to find optimal values for a series of parameters mentioned in Table 6. Each trial was run using the proposed dataset from this work split with 80% of the data used in training and 20% used in validation. Each trial began with random weights and had a 10 epoch early stopping. The loss values presented are from the lowest validation loss throughout the training curve. Most trials resulted in no clear advantage. Performance on the robot of the best learning-based model along with the lack of clear benefit from parameter changing in most cases suggests that the models are not actually learning the system dynamics but rather finding a local minima that has no real world value. More work is needed to find solutions that break free of this minima.

Table 14: LSTM layer search

Backbone Linear Layer(s) Size(s)	**Head Linear Layer(s) Size(s)**	**LSTM Layer Size**	**Best RMSE Validation Loss (A)**
10, 10	50	0	0.0942
10, 10	50	5	0.0281
10, 10	50	10	0.0230
10, 10	50	15	0.0220
10, 10	50	20	0.0222
10, 10	50	25	0.0218
10, 10	50	30	0.0229
10, 10	50	35	0.0224
10, 10	50	40	**0.0212**

Table 15: Head layer size search

Backbone Linear Layer(s) Size(s)	**Head Linear Layer(s) Size(s)**	**LSTM Layer Size**	**Best RMSE Validation Loss (A)**
10, 10	0	20	**0.0226**
10, 10	25	20	0.0236
10, 10	50	20	0.0236

Table 16: Head layer depth search

Backbone Linear Layer(s) Size(s)	**Head Linear Layer(s) Size(s)**	**LSTM Layer Size**	**Best RMSE Validation Loss (A)**
10, 10	0	20	0.0226
10, 10	25	20	0.0236
10, 10	25, 25	20	**0.0224**
10, 10	25, 25, 25	20	0.0233
10, 10	25, 25, 25, 25	20	0.0226

Table 17: Backbone layer size search

Backbone Linear Layer(s) Size(s)	**Head Linear Layer(s) Size(s)**	**LSTM Layer Size**	**Best RMSE Validation Loss (A)**
10	25, 25	20	0.0223
20	25, 25	20	0.0211
30	25, 25	20	**0.0205**
10	25, 25	15	0.0234
20	25, 25	15	0.0251
30	25, 25	15	0.0228

Table 18: Backbone layer depth search

Backbone Linear Layer(s) Size(s)	**Head Linear Layer(s) Size(s)**	**LSTM Layer Size**	**Best RMSE Validation Loss (A)**
5	25, 25	20	0.0225
5, 5	25, 25	20	0.0241
5, 5, 5	25, 25	20	0.0235
10	25, 25	15	0.0234
10, 10	25, 25	15	0.0243
10, 10, 10	25, 25	15	0.0233
20	25, 25	15	0.0251
20, 20	25, 25	15	**0.0221**
20, 20, 20	25, 25	15	0.0244
30	25, 25	15	0.0228
30, 30	25, 25	15	0.0224
30, 30, 30	25, 25	15	0.0238

B Alternative Path Planner

A digital model of the robot's task space and state configuration is used to generate trajectories to run on the physical robot. A random trajectory is generated by selecting a random point in the robot's task space and using A* to solve for a path between the current end effector position and the target point. A noise map is generated using Perlin noise [33] and used as a cost map in the A* algorithm to add task space variations in generated trajectories. The solution path from A* in discretized task space is converted to joint space using the baseline inverse kinematic model. A continuous-time joint trajectory is found using a similar method as [34] to ensure C^4 smoothness. C^4 smoothness is required for exact reproducibility of desired trajectories, as well as helps prevent structural oscillations and requires less energy [34]. A sample of this process is shown in Figure 30. This trajectory is sampled at a given command frequency and is used to set the reference motor velocities in the PID controller. This same method can be used to generate test trajectories consisting of a square, a circle, a sinusoidal pattern, and a zig-zag.

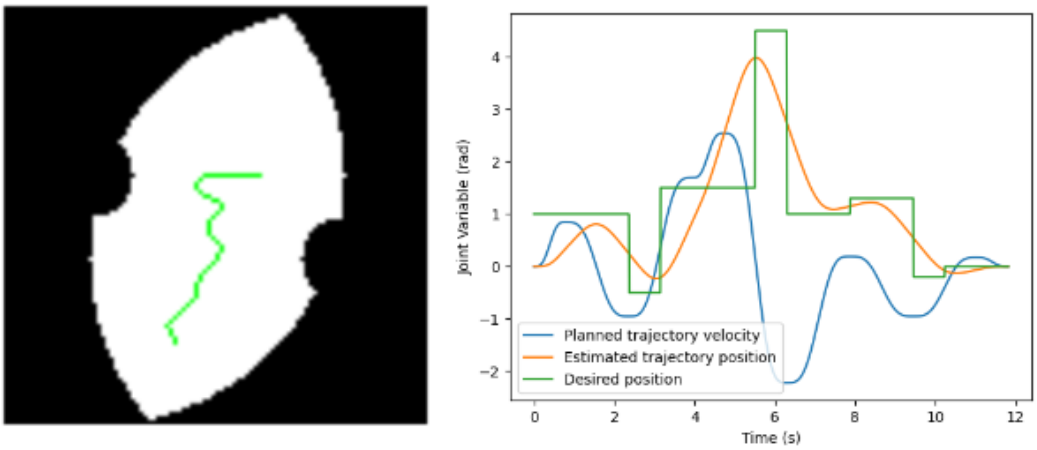


Figure 30: Trajectory A* search (left) and joint interpolation (right) results