LEARNED MODELING AND CONTROL OF CONTINUUM ROBOTS FOR SURGICAL APPLICATIONS

A Thesis
Submitted to the Faculty
in partial fulfillment of the requirements for the
degree of

Master of Science

in

Engineering Sciences

by Cameron Wolfe

Thayer School of Engineering
Guarini School of Graduate and Advanced Studies
Dartmouth College
Hanover, New Hampshire

May 2024

Examining Committee:

Chair_____
Ryan Halter, Ph.D.

Member_____
Michael Kokko, Ph.D.

Member_____
Minh Phan, Ph.D.

_____
F. Jon Kull, Ph.D.
Dean of the Guarini School of Graduate and Advanced Studies

Abstract

To reduce morbidity and mortality during surgery, surgeons have increasingly turned to Minimally Invasive Surgery (MIS), which involves passing instruments through small incisions or natural orifices to minimize patient trauma. Although MIS has significantly improved patient outcomes, it hinders a surgeon's dexterity and impairs visual and tactile feedback. These deficiencies have prompted the adoption of Robot-Assisted Surgery (RAS), in which surgeons control robots instead of using handheld instruments. While RAS has improved patient outcomes, robots struggle to navigate constricted spaces due to their rigidity, spurring the development of Continuum Robots (CRs). These flexible infinite degree-of-freedom robots move by bending, allowing them to snake through passageways without damaging surrounding tissue. However, this flexibility complicates modeling and control due to the nonlinear effects of material properties, external loads, and material degradation on movement. This dissertation continues the work of Carolina Lago Pena Maia, and Brook Leigh in developing a low-cost CR for surgical applications. Specific improvements include upgrading the robot by creating a new, more resilient, flexible spine, reducing the robot's frame volume from 5.86 to 0.66 liters, developing hardware to improve tip pose accuracy measurement, and writing robust control and communications software. With this enhanced robot, an Artificial Neural Network (ANN) was used to learn an accurate, lightweight forward kinematics model of the robot based on measured input/output data. The conventional modeling approach, known as constant curvature modeling, assumes the robots bend in a series of arcs, which makes modeling tractable but significantly degrades accuracy. The learned model outperforms existing constant curvature models, with an RMS position error of 2.2 mm vs 9.5 mm on our robot. This ANN-based model was then applied to develop optimal open-loop and Jacobian-based closed-loop control policies. The robot uses these control policies to follow a representative trajectory through the robot's workspace, with the open and closed-loop

policies achieving RMS position errors of 2.415 mm and 1.619 mm, respectively. This work demonstrates the potential of learned kinematic models for developing accurate and reliable CRs, which have the potential to lower costs and improve patient safety in the operating room.

Acknowledgements

I want to thank my advisors, Professor Ryan Halter and Professor Mike Kokko, for supporting me in this project. They have kept me focused and on track while acting as a sounding board for my ideas, pointing out flaws, and highlighting important insights. Without them, this work would not have occurred. I would also like to thank the members of the Halter Lab at the Thayer School of Engineering for providing me with a cutting-edge environment to pursue my work and a community that supported me throughout the duration of my thesis. I have benefitted immensely from my collaboration with you all. Finally, I would like to thank my Mom, my girlfriend Dagny, my Coach Brayton, and my teammates on the Dartmouth Ski Team for cheering me along throughout this project. I leaned on you many times, and your energy was integral to my completion of this thesis. I am incredibly lucky to have so many great people in my life.

Contents

1 Introduction **1**

2 System Design **8**

2.1 Mechanical Design 10

2.1.1 Spine Development 10

2.1.2 Actuation and Frame Design 28

2.2 Electrical Design 35

2.2.1 Actuation 36

2.2.2 Tracking 39

2.3 Software Design 39

2.3.1 Serial Communication 40

2.3.2 Cable Tensioning 44

2.4 Conclusion 46

3 Modeling and Control **49**

3.1 Registration 50

3.2 Modeling 56

3.2.1 Constant Curvature Modeling 57

3.2.2 Learned Modeling 69

3.2.3 Model Comparison 85

3.2.4 Model Sources of Error 86

3.3 Control 89
3.3.1 Open Loop Control 90
3.3.2 Closed Loop Control 94
3.3.3 Controller Performance 98
3.4 Conclusion 104

4 Conclusion **107**
4.1 Potential Future Areas of Inquiry 109
4.1.1 Physical Improvements 110
4.1.2 Modeling 110
4.1.3 Control 111

A Bill of Materials **113**

B Software and Code **117**

C Spine Parameters **118**

D Two Segment Training **119**

List of Tables

2.1 Arduino communication bytes 41

2.2 Example packet 41

2.3 Transmission flags, which inform the Arduino as to the type of packet sent, are sent in a byte after the start sequence. 42

2.4 Error codes aid in debugging, as they give the user an indication of what went wrong on the Arduino. 43

3.1 Constant Curvature DH parameters 59

3.2 Fitted Camarillo stiffness 68

A.1 Bill of materials for continuum robot system. 116

C.1 Parametric design parameters of flexible sections. 118

List of Figures

1.1 Robot-Assisted Surgery (RAS) robots 2

1.2 Robot comparison 4

2.1 Current design 9

2.2 System exploded view 11

2.3 Previous designs 13

2.4 3D printing iterative development 13

2.5 Robot spine 14

2.6 Disposable Trocars 16

2.7 Differential element in bending 18

2.8 Initial flexible section 20

2.9 V1 bent flexible section 21

2.10 V2 bent flexible section 21

2.11 Helical segment 22

2.12 Tip integrated sensor 23

2.13 Rigid section 24

2.14 Spine base 25

2.15 Robot tip 26

2.16 Two segment spine 27

2.17 Cable termination 29

2.18 Cable pulley 30

2.19 Frame assembly 31
2.20 Frame top 31
2.21 Frame wall 32
2.22 Cable guides 33
2.23 Registration hat 34
2.24 Electrical block diagram 35
2.25 Futaba U-305 37
2.26 Servo control hardware 38
2.27 Aurora field generator 39
2.28 A histogram showing the average time required to request and receive a packet. This shows that there is significant room for improvement. . 45
2.29 Tensioning data 46

3.1 SVD 52
3.2 Reference frames 53
3.3 Constant curvature parameters 58
3.4 Robot independent vs dependent parameters 58
3.5 Bending projection 60
3.6 Fitted Camarillo points 69
3.7 Constant curvature error 70
3.8 Simulation loss 79
3.9 Simulation model error 80
3.10 As can be seen, depending on the direction in which the spine moves, there can be a significant decay period before the spine tip reaches its final position. Specifically, on trajectories where the overall tension on the spine is loosened, the spine is responsible for springing into shape, which is a lot slower than the movement of the servos seen in the steep initial change in both trajectories. 82

3.11 Training data distribution 82
3.12 Slack data distribution 83
3.13 Real data loss 84
3.14 Real data model error 85
3.15 Learned test loss 86
3.16 Comparison of model errors 87
3.17 Control architectures 89
3.18 Desired trajectory 90
3.19 Simulated controller error 99
3.20 Real controller error 101
3.21 Camarillo distribution with hysteresis 102
3.22 Hysteresis effects 103

D.1 Two segment loss 120
D.2 Two segment error 122
D.3 Large two-segment model loss 123
D.4 Large two-segment model error 124

Glossary

ACK Acknowledge. 41, 43

ANN Artificial Neural Network. 71

BFGS Broyden-Fletcher-Goldfarb-Shanno. 92, 94

BSC Binary Synchronous Communication. 40, 44, 108

CAD Comptuer Aided Design. 32, 33

CRC Cyclic Redundancy Check. 41

DCM Direction Cosine Matrix. 74–76

DLE Data Link Escape. 41

DOF Degree of Freedom. 3, 15, 26, 39, 56, 57, 74

EIT Eletrical Impedance Tomography. 6, 23

ERR Error. 41, 43

ETX End Transmission. 41

FDM Fused Deposition Modeling. 17

FEM Finite Element Method. 5, 56, 70

FLE Fiducial Localization Error. 88

FRE Fiducial Registration Error. 88

ID Inner Diameter. 16

MCU Microcontroller. 37

MIS Minimally Invasive Surgery. 1, 2, 14–16, 107

MPC Model Predictive Control. 106

MSE Mean Square Error. 72

NOTES Natural Orifice Transluminal Endoscopic Surgery. 1, 107

OD Outer Diameter. 15, 23–25

PSA Prostate-Specific Antigen. 6

PWM Pulse Width Modulation. 37, 38

RAB Robot-Assited Bronchoscopy. 3

RAS Robot-Assisted Surgery. viii, 1–3, 108

RC Radio Control. 36

ReLU Rectified Linear Unit. 71, 74

RL Reinforcement Learning. 106

RMS Root Mean Square. 88, 100, 103, 104, 109

SLA Stereolithography. 17, 18

STX Start Transmission. 41

TORS TransOral Robotic Surgery. 3

TREIT Transrectal Electrical Impedance Tomography. 6

TRUS Transrectal Ultrasound. 6

XOR Exclusive Or. 41

Chapter 1

Introduction

Globally, around 300 million surgical procedures are performed each year. Of these surgeries, it is estimated that roughly 15% of patients experience significant postoperative morbidity [1]. Over the past decades, surgeons have adopted Minimally Invasive Surgery (MIS) techniques to achieve better outcomes for their patients. The most common forms, laparoscopy, and Natural Orifice Transluminal Endoscopic Surgery (NOTES), utilize existing orifices along with small, keyhole incisions that surgeons feed instruments and operate through [2]. Adopting MIS has led to significant reductions in recovery times, complication rates, and hospital stays compared to open surgeries, including hepatectomies and cholecystectomies [3][4]. As a result, surgeons are looking for ways to increase the number of MIS procedures performed.

MIS, however, is not without its drawbacks. Surgeons performing MIS report issues with dexterity, sensory feedback, and visualization [2], which have the possibility to negatively influence patient outcomes. To address these issues, surgeons are increasingly turning to Robot-Assisted Surgery (RAS). Surgical robots, which are generally operated by the surgeon from a console, attempt to remedy the drawbacks of MIS. The Intuitive Surgical daVinci robots, for example, provide manipulators with wrist-like dexterity, 3D views of the surgical field, and tremor filtration. Additionally,

platforms such as Hansen Medical's Sensei X robotic catheter system provide features such as haptic feedback to aid the surgeon during operation [2]. These platforms can be seen in Figure 1.1. Although differences between outcomes with RAS and traditional MIS are smaller than between MIS and open surgery, RAS has been shown to improve patient outcomes in a wide variety of procedures [5].

(a) Intuitive Surgical daVinci Xi patient cart.

(b) Hansen Medical Sensei X robotic catheter system [2]

Figure 1.1: Examples of current surgical robots used for RAS procedures.

Outside of medical applications, robots are used to perform a wide variety of tasks. Manufacturing has seen the most widespread adoption of robotics, as robots can perform the same action thousands of times quickly and with high repeatability [6]. While there is an initial investment required to set up and program robots, once they are up and running, they are significantly more productive than a human can be in repetitive tasks. Therefore, as production volume goes up, robots increasingly make more sense. However, even in high-volume production processes, robots are not always applicable for two reasons: adaptability and safety.

This same concern is in place in surgical applications. As surgeons attempt to pursue more complicated surgeries in more confined spaces in the human body, eventually, a robotic surgery system's size becomes the limiting factor. They simply do not possess the room needed to perform the procedure. A clear example of this is in otolaryngology, where the adoption of RAS lags significantly behind that of other

surgical specialties. The head, neck, and throat is a region that requires high levels of precision, making it an excellent candidate for RAS, leading to the development of TransOral Robotic Surgery (TORS) [7]. However, due to the space constraints imposed by human anatomy on robotic instruments, RAS cannot be used in all cases [8].

Similarly, procedures like bronchoscopies were, until recently, unable to benefit from RAS. The rigid instruments of standard robots cannot navigate the tight, snaking passageways of the lungs. However, the release of the Monarch by Auris Health in 2018 and the subsequent release of the Ion Endoluminal System by Intuitive Surgical in 2019 has opened up the field of Robot-Assited Bronchoscopy (RAB) [9]. These systems are not typical robots, however. Instead, they are known as continuum robots, which are infinite Degree of Freedom (DOF) robots that move by deforming their structure [10]. This is in contrast to standard robots, which avoid deformation in their structure at all costs and instead move through the use of rotational and prismatic joints.

Although first proposed in 1960, continuum robots did not receive significant attention until around 1990, when, in modeling hyper-redundant robot systems (see Figure 1.2), the approximation that these robots acted like continuum robots was used, and an early continuum robot prototype was developed, which sparked interest in their development [11]. Since then, continuum robots have continued to grow in popularity, as their dexterity and flexibility are enticing in a variety of scenarios. Most notably, they have experienced significant interest from doctors and biomedical engineers who see the broad applicability of a compliant, flexible robot interacting with patients safely [12].

While the variety in continuum robot designs is vast, they are all united by their utilization of bending for movement [11]. Common examples of continuum robots include tendon-driven continuum robots, pneumatically actuated continuum robots,

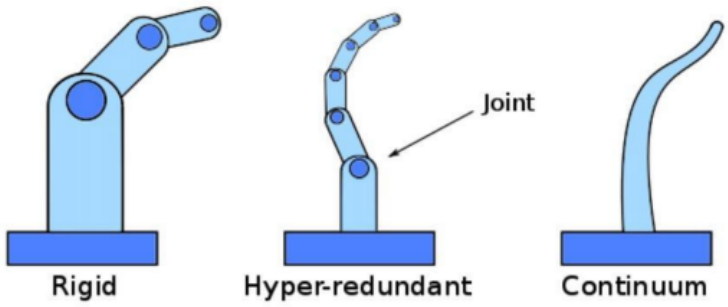


Figure 1.2: The different types of robots. As the number of joints increases, hyper-redundant robots approach the behavior of continuum robots. Figure from [13].

and concentric tube continuum robots. Tendon-driven continuum robots, as the name implies, are continuum robots that rely on the transmission of forces through cables to impart a bending moment on the structure. Pneumatic continuum robots, on the other hand, utilize pneumatic chambers along their length which can, by altering the pressure within them, change the shape. Finally, concentric tube continuum robots are collections of prebent tubes, which can be rotated and translated in and out of each other to assume highly complex shapes [10]. Additionally, there exist hybrid systems that combine multiple actuation strategies into one, such as combined pneumatic tendon-driven continuum robots [14]. As can be seen, the possible design space of continuum robots is vast and continues to grow.

However, continuum robots have seen relatively little adoption outside research settings and in limited medical settings. While their compliance makes them attractive, it is that same compliance that complicates both modeling and control of continuum robots [12]. Standard robots can be modeled as a series of rigid transformations. Rotary and linear encoders on the joints provide all of the information needed to model the tip pose using techniques such as Denavit-Hartenberg Parameters [15] or other kinematic modeling techniques. These techniques break down, however, when modeling continuum robots, as they don't have joints or links. While attempts to approximate continuum systems as a series of rigid links have been made [16], these

approximations require the use of large numbers of segments to achieve reasonable accuracy.

The two most commonly used modeling methods are constant curvature models and cosserat rod models. As the name implies, constant curvature models assume that a continuum robot bends in a series of constant curvature arcs. This simplifying assumption makes performing statics calculations tractable but decreases the model's accuracy. Examples of constant curvature models can be found in [10], [17], and [18]. Cosserat rod models, on the other hand, do not require the use of any simplifying assumptions. However, these models require solving a series of coupled nonlinear equations. These must be solved numerically, often requiring significant computation. Examples of cosserat models can be found in [19], [20], [21]. Other methods, such as Finite Element Methods (FEMs), have been applied. However, these are also computationally expensive [22].

In the last decade, increases in computational power and the availability of cheap digital storage have created an explosion in the use of techniques known as deep learning [23]. Deep learning is a type of machine learning that uses deep neural networks to train models to do everything from blocking spam emails and selling advertisements to autonomously driving cars and operating robots [23]. This revolution is built on these neural networks' ability to approximate any continuous function with arbitrary precision given a big enough network [24]. Given the challenges with modeling continuum robots and the non-linear approximation power of neural networks, there has been significant interest in the use of neural networks to develop models and controllers for continuum robots [25] [26].

Developing a continuum robot with an accurate controller enables surgeons to perform procedures and collect data that they otherwise would not be able to. For example, the detection of prostate cancer has the potential to be vastly improved through the use of a continuum robot system. In the United States, around one in

seven men will develop prostate cancer [27]. Currently, men are screened for the disease by measuring levels of Prostate-Specific Antigen (PSA) in the blood [28] and performing Transrectal Ultrasound (TRUS) and biopsies for patients with abnormally high PSA levels [29]. This procedure, however, has been shown to miss 10%-30% of cases [30]. This has led to the proposal of Eletrical Impedance Tomography (EIT), more specifically Transrectal Electrical Impedance Tomography (TREIT), to improve the performance of existing procedures by more accurately targeting areas for biopsy.

EIT is a non-invasive imaging technique that uses small, high-frequency AC currents to measure the impedance characteristics of tissue. Currents are pushed through the tissue, and arrays of electrodes measure the resultant voltages, which can be used to determine the conductivity of the tissue [31]. It has been shown that EIT has the ability to differentiate between cancerous and healthy tissue [32]. Therefore, by fusing EIT with other existing imaging techniques, surgeons can better discriminate healthy from cancerous tissue [33], leading to improved survival rates from the disease.

Integrating an EIT with continuum robots has the potential to provide large amounts of high-quality EIT data non-invasively. The precision of continuum robots would allow surgeons to image many locations on and around the prostate to maximize the coverage of prostate tissue. Additionally, the flexibility of continuum robots means that these measurements can be taken without being constrained by existing anatomy.

With the ultimate objective of creating a combined continuum TREIT system, this thesis builds upon the work of Brook Leigh [34] and Carolina Lago Pena Maia [35] in developing a low-cost continuum robot based on the open-source ENDO platform [36]. Specifically, improvements were made to the robot's physical hardware by developing a more resilient flexible spine, reducing the robot's overall size, improving tip pose measurements, and writing robust control and communications software. These improvements are discussed in Chapter 2. Additionally, neural networks were used to learn accurate forward kinematics models that significantly outperformed existing

modeling techniques. Finally, these models were used to control the robot both with and without feedback, highlighting their versatility. These efforts are discussed in Chapter 3.

Chapter 2

System Design

The current design of the robot can be seen in Figure 2.1. As an electromechanical device, development consisted of three main focuses: mechanical, electrical, and software. The mechanical design, presented in Section 2.1, had two primary objectives: improving the performance and reliability of the spine, and developing a modular frame that supports the robot. The development of the spine, including the development and manufacturing process, is covered in detail in Section 2.1.1. After developing the spine, work shifted to developing a frame that could house all parts of the robot and manage tasks like cable routing. Additionally, because of the modular design of the spine, the frame needed to be able to support single and multi-segment designs. This development can be seen in Section 2.1.2.

The electrical system (Section 2.2) is responsible for actuation and sensing. In order to drive the robot, internal cables are routed through the spine and pulled using servomotors. This actuation system, including motors and the hardware to control them, is covered in Section 2.2.1. Additionally, knowledge of the location of the robot tip frame is necessary for accurate navigation and control, along with any future applications of the system. Therefore, a system to measure the position and orientation of the robot was required. This development is covered in Section 2.2.2.

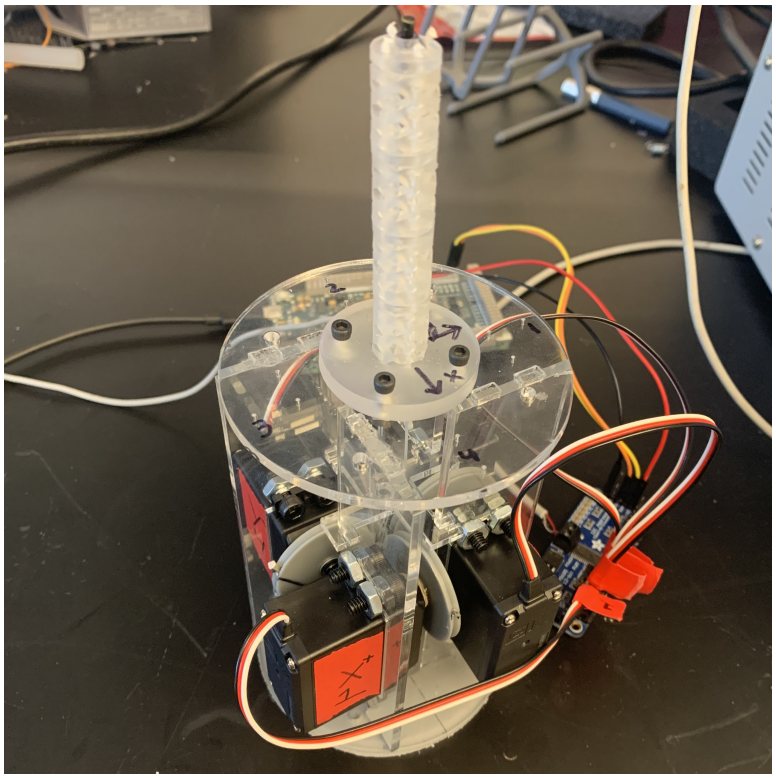


Figure 2.1: The current design of the robot consists of a spine, a frame, actuation mechanisms, components for position measurement, and control hardware. Each of these features will be discussed in detail in the following chapter.

Additionally, while the system has the hardware to measure the position of the tip of the robot in 3D space, the measurement is not usable in is current form. To remedy this, we need a method to register the robot (and its associated coordinate frames) in space. While the math behind this will be covered later in Section 3.1, the physical hardware for registration is covered in Section 2.1.

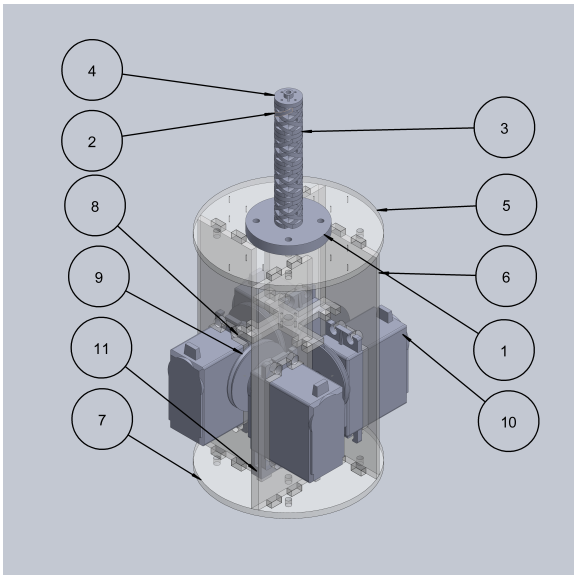
Finally, software allows mechanical and electrical systems to work together. It is responsible for orchestrating the various behaviors that the robot performs. This starts with the ability to send commands to the motors in a secure and reliable format, covered in Section 2.3.1. This allows the system to be controlled and the robot to move. Routines, such as automatic cable tensioning and, ultimately, servo-coordinated spine motion, are then built on top of this control foundation. This is covered in Section 2.3.2.

2.1 Mechanical Design

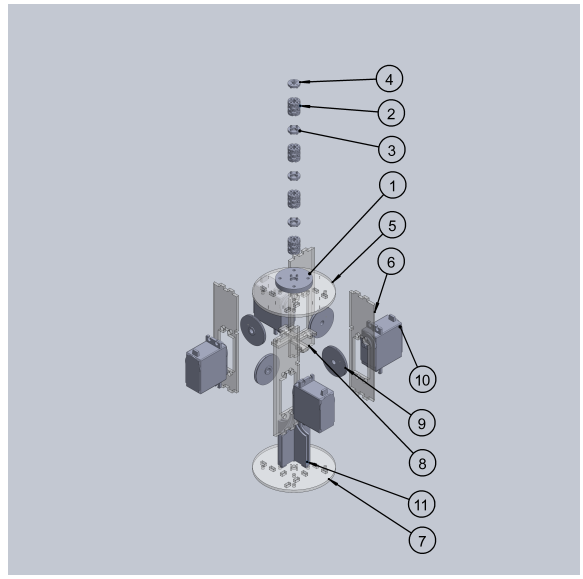
This continuum robot has two primary sub-assemblies: the spine and the frame. An exploded view of the system can be seen in Figure 2.2. The spine is what allows the continuum robot to move, providing flexibility for the internal tendons to control. As the spine of the robot largely determines its performance, this sub-assembly underwent many design iterations. This development process is covered in Section 2.1.1. The frame, on the other hand, provides structure and organization to the robot, supporting the various actuators and electronics. This development is covered in Section 2.1.2. A list of all of the components used in the system can be found in Appendix A

2.1.1 Spine Development

The spine of a continuum robot must be rigid enough to support itself and external loads while being flexible enough to stretch and bend to move around. As a result,



(a)



(b)

ITEM NO.	NAME	QTY.
1	Spine Base	1
2	Spiral Segment	4
3	Rigid Disc	3
4	Spine Tip	1
5	Frame Top	1
6	Frame Wall	4
7	Frame Base	1
8	Horiz. Cable Guide	1
9	Cable Pulley	4
10	Servo Motor	4
11	Vert. Cable Guide	1

(c)

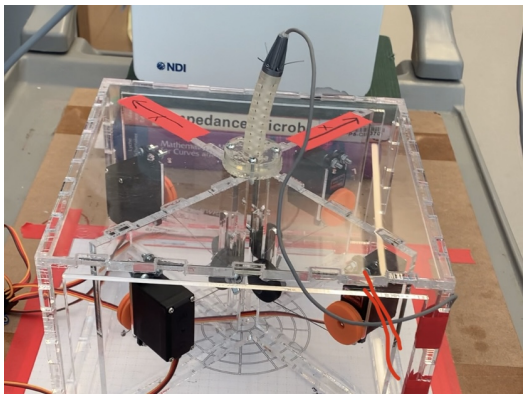
Figure 2.2: Exploded view of continuum robot, showing all of the individual parts of the system.

the materials used and the structure selected are crucial to maximizing performance. The current spine morphology extends the work done by Lago Pena Maia and Leigh, taking their design and adding improvements based on similar continuum devices reported in the literature. This section will begin by exploring existing designs and taking the best components of each. Next, it will cover in detail the design process behind the parts of the spine. Finally, it will highlight further areas for improvement.

Comparison with Existing Designs

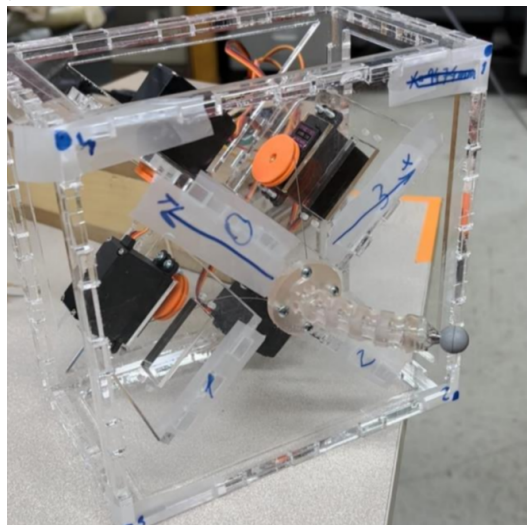
This thesis was a continuation of the work done by Lago Pena Maia and Leigh [35] and [34], in which they built a single-segment, cable-driven continuum robot based on the design by [36]. All three of these continuum robots can be seen side by side in Figure 2.3. The Lago Pena Maia and Leigh devices were designed focusing on developing a spine that was simple to manufacture. The spine was printed in a single piece on a Formlabs Form 3 SLA printer [37] using their Flexible 80A durometer resin in the Lago Pena Maia design, and 50A durometer resin in the Leigh design [38]. Printing in one piece simplified manufacturing allowing them to iterate on designs quickly. This iteration can be seen in Figure 2.4. However, one major drawback was that at high tensions, the cables would tear through the spine. This limited the robot's range of motion and any potential external loads it could support.

To address these issues, inspiration was taken from the ENDO open-source continuum robot design by Clark et al., the system on which Lago Pena Maia and Leigh based their designs [36]. This involved constructing a new spine with rigid discs along its length. These discs supported the cable tension, allowing the new spine, seen in Figure 2.5, to achieve much higher curvatures than the original design. One important distinction between the current design and the ENDO is that the flexible pieces of the ENDO are printed in one piece, and then the discs are glued to the side of these flexible pieces, whereas the current design is comprised of discrete flexible pieces that



(a)

(b)



(c)

Figure 2.3: Previous spine designs which served as inspirations to my design (a) Leigh design with aurora tracking coil [34] (b) Clark design with end effector ©2021 IEEE [36]. (c) Lago Pena Maia design with an optical tracking sphere [35].

Figure 2.4: The iterative spine development process enabled by 3D printing [35].

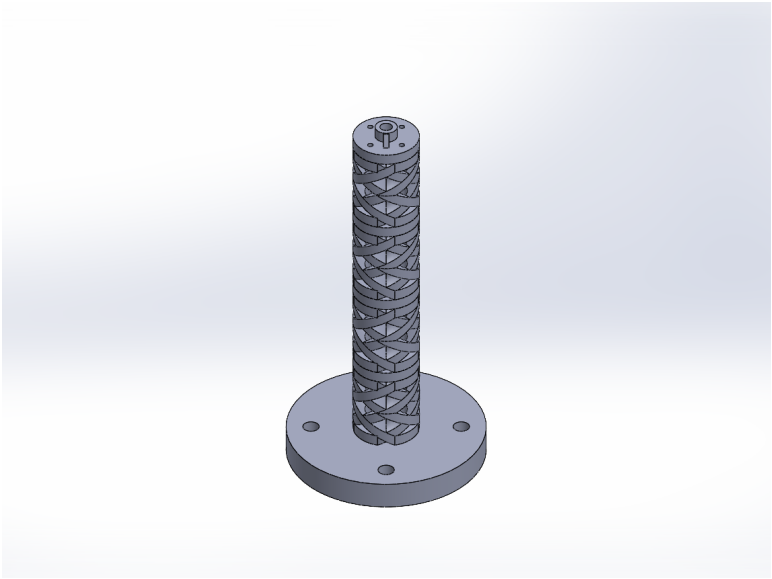


Figure 2.5: New spine design, with alternating rigid discs and flexible sections to allow for flexibility while preventing cable tear-through.

are glued to the rigid discs in an alternating fashion. While this does increase the number of parts per spine, it increases strength, provides design flexibility, and reduces variability between spines without sacrificing the resistance to cable tear-through of the Clark design.

While other methods of actuation were considered, such as combined pneumatic-tendon systems [39] and push rods [40], ultimately, tendons were chosen to control the robot. This design decision resulted from the intended application, MIS. In the case of pneumatic actuators, they can provide significant stiffness to the spines of continuum robots, which is useful for handling external loads and positioning the tip more precisely [14]. However, they also add significant design complexity, requiring an airtight component along the length of the spine, as well as the equipment to pressurize it. All of this extra overhead takes up more space and may have led to a much bulkier robot, which would have disqualified it from being used in MIS as it most likely would not have fit through a 12mm cannula.

Similarly, push rods are attractive because, unlike tendons, they can both push and pull the spine to move it [40]. Additionally, the rods add stiffness to the spine, which,

similar to pneumatic designs, is useful for load-bearing and positioning accuracy. However, push rods cannot be wound around pulleys like tendons can, meaning actuating them is more complicated. Additionally, routing rods through the same channels is more challenging as they interfere with each other, which makes adding DOFs by stacking segments much more challenging. Finally, using push rods also sacrifices much of the pliability that makes continuum robots attractive for MIS in the first place, as rods are generally made out of metal.

While new actuation mechanisms may be interesting to explore in future design iterations, they were not the project's ultimate goal and, due to time constraints, were not pursued. Once the overall form of the spine was selected, work could proceed on the details of the design, including developing the individual components of the spine.

Components of Spine

The spine can be broken into three components: a rigid base, flexible sections, and rigid discs. For these components to function together, they must share a collection of universal parameters. The first parameter is the number of cables per segment. To access all of its DOFs, each segment must be able to bend in two axes. If the continuum robot were driven by rods that could both push and pull, only two actuators would be needed. However, because tendons can only pull, a third (antagonistic) cable is needed that can bend the spine in the opposite direction. While, at minimum, three cables are required per section, adding additional cables lowers the maximum tension in the cables and adds an additional DOF, namely compression of the spine [41]. Therefore, a fourth cable per segment was added, as this provided the benefits of additional cables without significantly increasing the number of actuators required.

The second parameter that all components must adhere to is a maximum Outer Diameter (OD) of 12 mm. Because the robot is targeted for applications in MIS, it must be compatible with existing trochars and cannulas. A trochar, seen in Figure

Figure 2.6: Examples of disposable trochars used in standard MISs, gotten from [43].

2.6, is a device that acts as a portal for surgical instruments to pass through [42], minimizing damage to the surrounding tissue. While trocars can have up to an Inner Diameter (ID) of 15mm, trochars this size are less common than 12 mm ID trochars. Therefore, all components that are inserted into the patient (the flexible sections and rigid discs) are at most 12mm in size.

Finally, because the robot will be inserted through trochars, all sensors and cabling must be routed through the center of the robot. To achieve this, a pass-through hole with a diameter of 4mm was added to each component in the spine and frame. With these parameters established, the design of individual spine components follows.

Flexible Sections

The flexible sections allow the robot to bend and assume many different shapes. At the same time, however, these sections have to be able to support the weight of the robot and its payloads throughout its workspace. Development of the flexible sections went through two phases: an initial materials selection period, followed by a period of iterative development to arrive at a final design.

A flexible spine can be constructed in many ways, with the most intuitive being either developing a mold and casting or 3D printing using flexible materials. While casting does offer many benefits, such as isotropic material properties, a wider range of possible materials, the ability to fine-tune the material properties via additives, and increased strength, it ultimately was not selected. This is because casting requires significant time to develop a mold, drastically slowing the iterative design loop. With 3D printing, on the other hand, a design can be taken from CAD and made within hours. Considering how large the design space is, design flexibility took priority and 3D printing was used. After finalizing the design, however, casting should be considered.

Additionally, there are also many different types of 3D printing. The two most common techniques in 3D printing are known as Fused Deposition Modeling (FDM) and Stereolithography (SLA). FDM is the most well-known form of 3D printing and involves extruding melted thermoplastic filament into layers. As they are printed, each layer fuses to the layer below it as the part is slowly built from the bottom up. SLA, on the other hand, is one of the oldest forms of 3D printing and uses light and photopolymer resin to stack layers on top of each other [44]. Parts are immersed in a bath of UV-curable resin, and light is selectively applied to sections of the model, which forms a thin layer of cured resin. These layers are then cured, one on top of the other, to form the part. After this initial printing step, the parts must be post-processed by washing and another round of curing [45].

FDM and SLA both have benefits and drawbacks over the other. FDM, for example, is a much less involved process, requiring significantly less time and effort to produce a part. Additionally, FDM parts can use a wider range of materials and are, on the whole, stronger than SLA parts [46]. However, SLA can produce parts with higher dimensional accuracy, especially for small, intricate parts [47], and the parts have similar strength in all directions, even normal to layer lines [46]. Ultimately, because

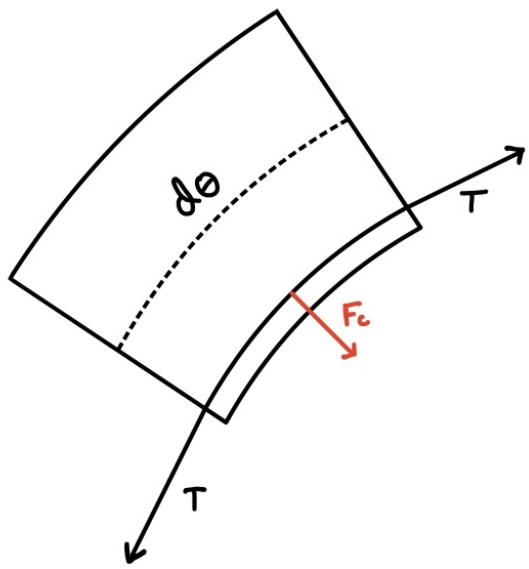


Figure 2.7: A differential element of the spine undergoing bending due cable tension.

this robot involves very small parts and needs to bend consistently in all directions, SLA was selected to create the flexible sections of the robot.

Based on the work in [35] and [34], Formlabs Elastic 50A resin was used. It is their most flexible resin [48], which means that it will be able to best handle the large bending displacements the flexible parts of the robot will undergo. For example, it is common to see bending angles along the length of a segment greater than 90 degrees. Additionally, because the lateral force applied to the spine of the robot is proportional to the cable tension, a more flexible material will require less tension to bend, meaning less friction is applied to the cables as they traverse the length of the robot, leading to improved performance [49]. This can be seen by analyzing the forces in a $d\theta$ element of a spine undergoing bending from a cable applying a bending moment, as seen in Figure 2.7.

The cable friction f can be found by finding the magnitude of the contact force F_c applied to the spine by the cable tension.

$$F_c = 2T \sin\left(\frac{d\theta}{2}\right) \tag{2.1}$$

Using the small angle approximation for $\sin x$ we find

$$F_c \approx T d\theta \tag{2.2}$$

For a given $d\theta$, increasing the tension T directly increases the contact force F_c. Because kinetic friction is proportional to the contact force

$$f_k = \mu_k F_c \tag{2.3}$$

Increasing the cable tension increases the friction experienced in the spine. Therefore, using a material that bends more easily will limit the effects of cable friction.

The first attempt at a flexible section can be seen in Figure 2.8. It included four vertical columns placed between each of the four cable holes. This design was very simple, but ultimately, it did not perform very well. Instead of bending consistently, this design would remain relatively stiff and then buckle suddenly. Not only was this sudden buckling detrimental to performance, but it repeatedly put much higher stress on the parts, causing them to tear. Additionally, bending achieved by pulling adjacent tendons concurrently, which was in the direction between the two cables, was significantly stiffer than bending in line with cables. This would have increased the complexity of modeling the system using constant curvature models (covered in Section 3.2.1). The inconsistency in the performance of these sections over their operating range would have made accurate control more difficult, causing this design to be rejected.

The next iteration of the design, seen in Figure 2.9, attempted to address the shortcomings of the previous design by adding a bend of 30 degrees from vertical in the center of each of the four columns. This meant that the flexible sections could flex much further before buckling. Additionally, bending between and in line with cables required roughly the same force. Therefore, this second design fixed many of

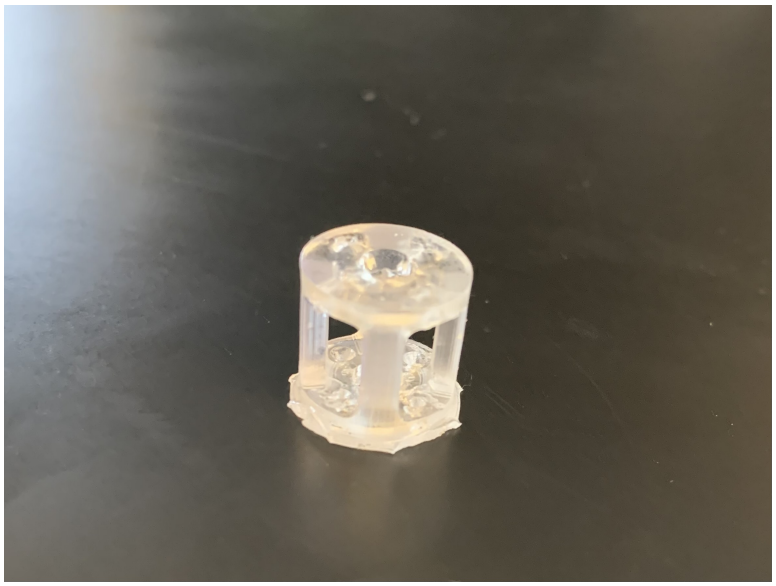


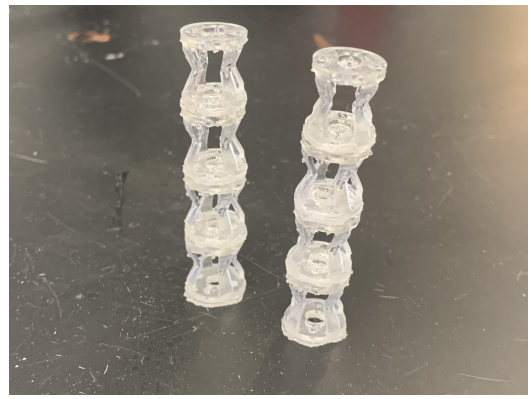
Figure 2.8: The initial design of the flexible section. It comprised straight columns, which would often buckle while bending. Therefore, this design was scrapped.

the problems of the first. However, the weakness that the bend introduced also meant that the spine struggled to support its own weight and would lean to the side. Thus, the design was not fully finished.

The next iteration of the design, seen in Figure 2.10, retained much of the geometry of the previous version. The only main difference is that the columns are twice as thick and that the bend angle of the columns was changed to 25 degrees. The part was designed parametrically so that all of the parameters of the design, for example, section length, bend angle, column thickness, etc., could be experimented with to produce the best results. A description of all of the parameters, as well as the current values, can be found in Appendix C.

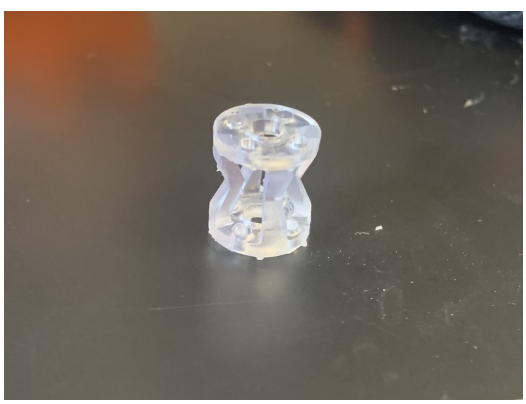
Ultimately, however, three main drawbacks were discovered while testing this design on the actual robot. This version of the flexible segment, although better than the straight column version, still did not bend uniformly in all directions. Additionally, this iteration of the flexible segment was prone to buckling under tension. Finally, this design was still relatively weak and could not support any external load. Therefore, the column-based design paradigm of the Clark robot was discarded, returning instead to

(a) An individual flexible section after printing.

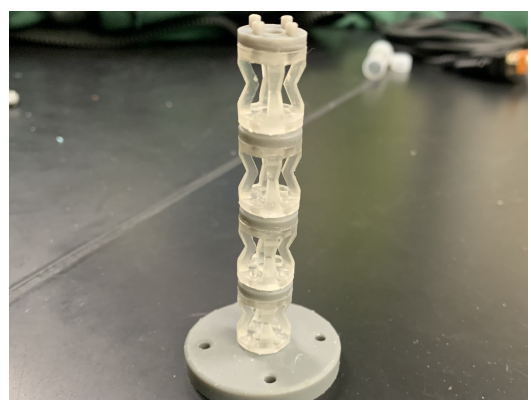


(b) The spine struggles to support its own weight.

Figure 2.9: Version one of the bent column flexible section. This design addressed the shortcomings of the previous design but was not strong enough to support its own weight, especially in a taller spine.



(a) A solitary flexible section with thicker columns.



(b) The flexible sections integrated into a spine, showing their rigidity.

Figure 2.10: Second iteration of the bent section. As can be seen, the spine can support its own weight without deflecting, which is an improvement over the previous iteration.

(a) (b)

Figure 2.11: The new helical spine segment bends more uniformly than the column design and can support higher loads.

the helical Leigh design. This design was much stiffer than the column designs and, for the most part, bent uniformly. However, as can be seen, this is difficult to characterize fully and is an area for further investigation. To prevent cable tear-through, the rigid sections of the previous designs were kept. This final segment design can be seen in Figure 2.11.

After much iterative development, the result is flexible sections that allow the spine to move while being strong enough to support its weight, plus that of payloads. An example of a sensor integrated into the tip can be seen in Figure 2.12

Rigid Sections

As stated above, the primary function of the rigid sections of the spine is to support the lateral forces of the cable tension to prevent the cables from tearing out of the spine, as this was a recurring issue in [35] and [34]. However, there are other requirements that the rigid sections must adhere to. First and most obvious, multiple cables need to be able to pass through the rigid sections freely. Second, because the spine can no longer be printed as one continuous piece, the rigid sections must also provide features to align the flexible sections. This alignment ensures the segments are coaxial and prevents rotation to ensure the cables are properly lined up throughout

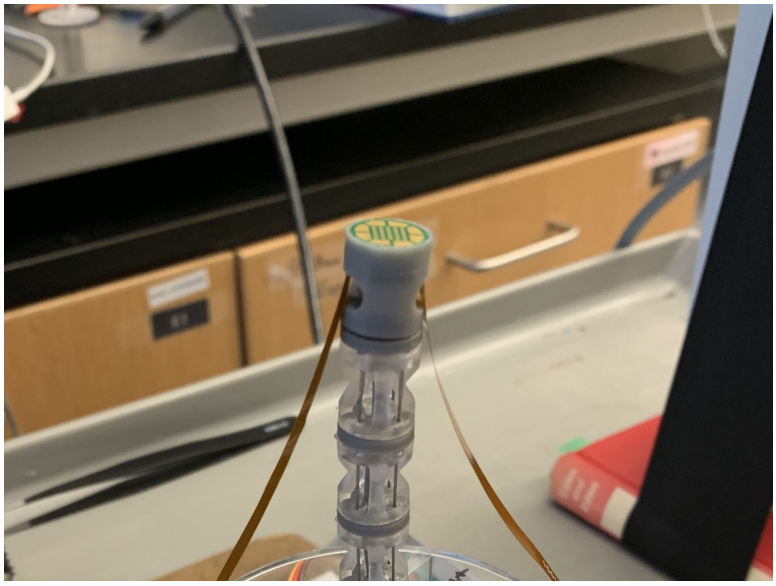


Figure 2.12: An example of an external load the system has to support. This is a prototype EIT probe attached to the robot's tip. As can be seen, it can be supported by an older, weaker design.

the length of the spine. Finally, the rigid sections must also provide ample surface to secure flexible sections together. The current design of the rigid sections can be seen in Figure 2.13.

In selecting a material, there were fewer constraints than with the flexible sections. All that was needed was a rigid resin. For this, Formlabs grey and clear resins were used depending on availability [50] [51], which provided ample strength to handle the cable tensions in the design. With a material selected, the design began with a cylinder of 12mm (matching the OD of the flexible sections), and the rest of the features were added.

The cable used in this robot is 0.38mm diameter fishing line. This was chosen over metal cable or other cables as it provided sufficient strength while remaining flexible. Because the eventual goal is a system with multiple segments, each rigid section must fit multiple cables through its cable routing holes. Each cable hole is offset 4mm from the center of the robot and is 1.1mm in diameter. This allows two cables to have ample room to slide right next to each other, ensuring that there is no unnecessary

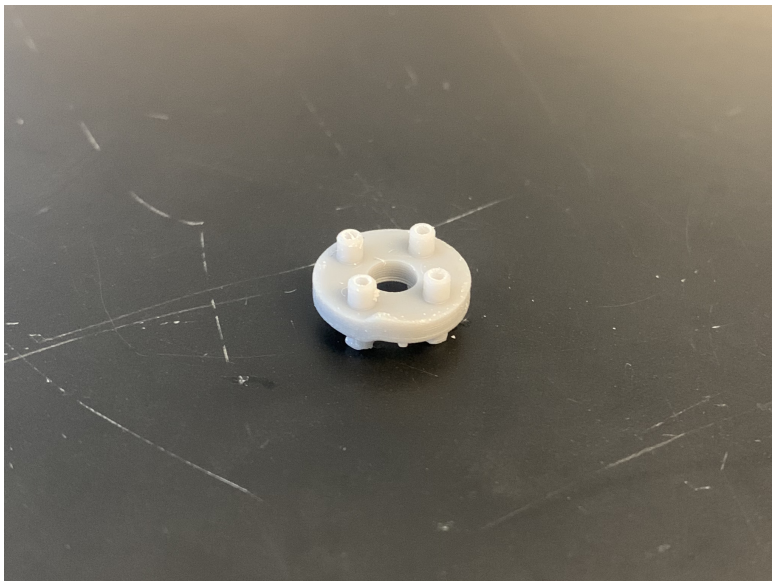


Figure 2.13: The current rigid section design, with cable holes and alignment features, provides a solid attachment point for flexible sections.

friction in the system.

As stated above, the robot's spine needs features to align each component during assembly. To achieve this, the fact that cables were already passing through both components was leveraged, adding 2mm cylindrical extrusions with an OD of 2mm located coaxially with the cables. These extrusions are matched by 2mm diameter holes in the flexible sections, which means that the flexible sections are locked into position laterally and rotationally, ensuring that the spine is aligned during manufacturing.

Finally, the spine needed to remain in one piece under actuation stresses. With this in mind, the discs were designed to provide ample surface area for gluing. All the flat parts of the discs and the alignment features could be coated in Loctite Ultra Liquid Control super glue during assembly. This ensured that other sections of the robot would break before the glue failed, as the glue's bond is significantly stronger than the flexible resin.

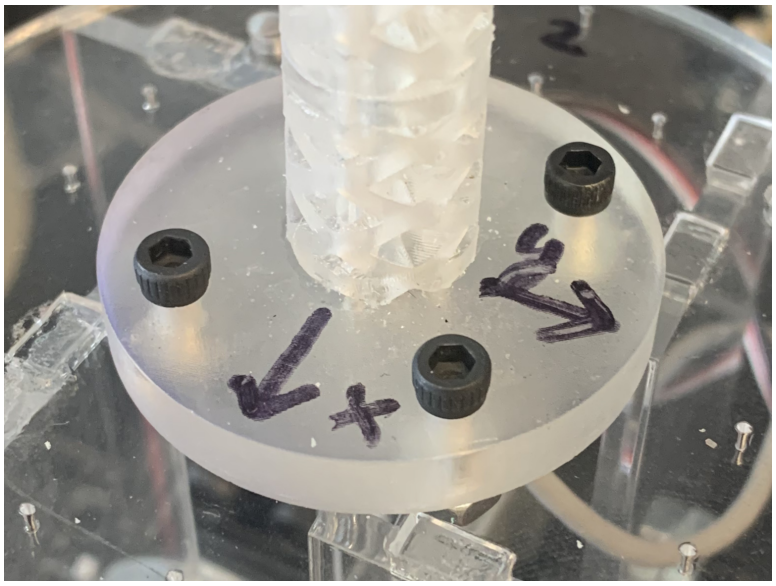


Figure 2.14: The robot's base provides a solid platform for the robot to stand on and is used to secure the spine to the frame.

Robot Base and Tip

In addition to the standard discs along the length of the spine, both the base and the tip of the robot have specialized designs that are based on the standard discs above. The robot's base, seen in Figure 2.14, is widened and thickened to an OD of 36mm and a height of 5mm. This provides stability to the robot and ensures it will not tip over during actuation. Additionally, four 4-40 screw holes are located in the base, allowing it to be attached to the frame rigidly. All of this provides a solid structure for the robot to be attached to and move from.

The robot's tip, shown in Figure 2.15, has three main requirements. It needs to provide space to terminate the cables, it needs to be able to hold a sensor for measuring position, and it needs to be precisely oriented for registration. The cables must be secured at the robot's tip and at the actuation wheels, the latter of which is covered in Section 2.2.1. At the robot's tip, each cable was tied into a double square knot, preventing the cable from being pulled back through the spine. These knots are secured to the tip disc through the tension in the cable, requiring no additional

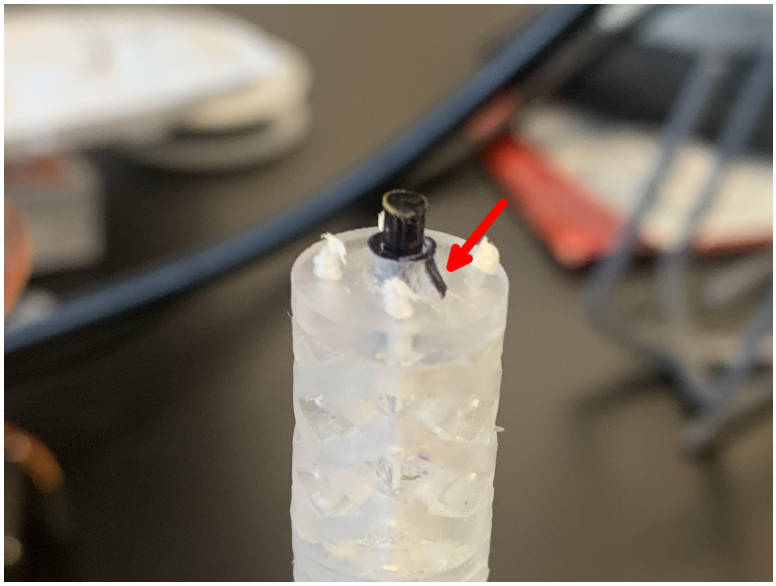
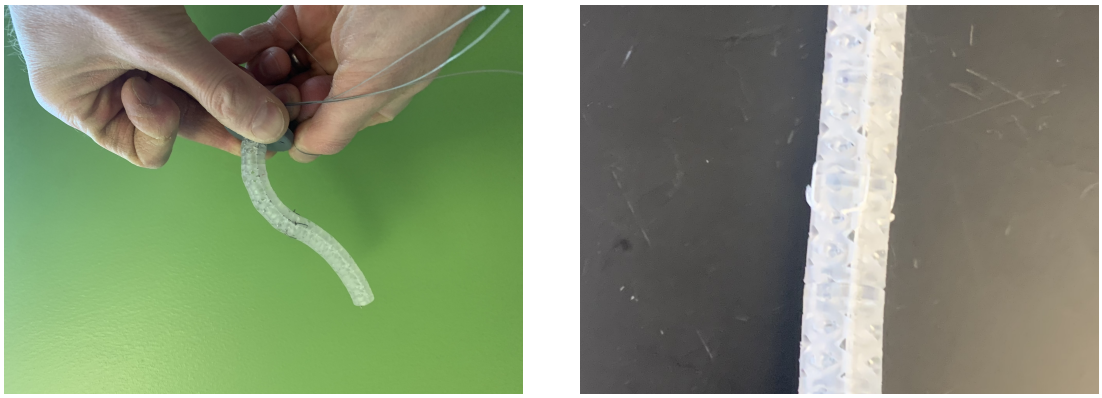


Figure 2.15: The tip of the robot, where cables are terminated, and position is sensed. Additionally, it has a rib (indicated by the arrow), which is useful for fixing the orientation during registration.

adhesives. This makes sure that the cables stay in their expected positions.

The system uses the NDI Aurora tracking system to measure the robot's tip position. While the details of this system will be covered in more detail in Section 2.2.2, it is an important part of the tip design. A 6 DOF Aurora tracking coil (p/n: 10002260), which is protected in a small plastic cylinder, is secured in the robot's tip via a friction fit. There is enough friction to keep the coil in place while the robot moves, but the coil can be removed from the tip if needed. This is crucial because the Aurora coils are delicate, meaning avoiding damage is paramount. The tip of the robot, therefore, allows the robot to measure its pose accurately and repeatably.

Finally, a vertical rib is added to the tip of the spine. This feature is used to fix the position and orientation of the robot's tip during registration. While registration will be covered in more detail in Section 3.1, this rib is used to find transformations between the tip frame and coil frame of the robot. Because the position and orientation of the robot's tip can be fixed using this rib feature, the pose of the Aurora coil can be located relative to the robot's tip. This allows one to determine where the tip of



(a) (b)

Figure 2.16: Two segment versions of the continuum spine. (a) The spine undergoing complex bending. (b) A zoomed-in view of the midway cable termination on the spine.

the robot is at any time based on measurements from the Aurora coil. This is then used later to measure the robot's tip in 3D space. The rib added to the robot's tip is crucial for making these processes accurate and repeatable, which reduces possible sources of error in the design.

Multisegment Spine Development

Because of the modularity of the spine design, creating a multi-segment spine is relatively straightforward, as more segments can be stacked onto the spine. This can be repeated for any arbitrary spine length so long as it can support its own weight. A two-segment spine has been tried (shown in Figure 2.16), and it is likely that a three-segment spine would still function correctly. To control the closer segment of the spine, one set of cables is terminated halfway along the spine, while the other set of cables goes all the way to the end of the spine. The cables halfway along the spine are tied to a rigid disc in the middle of the spine, which the cable can pull against to bend the spine. The current process for threading the cables into the spine could certainly use improvement, as it is incredibly tedious.

2.1.2 Actuation and Frame Design

To move around and interact with its environment, a continuum spine requires an actuation mechanism to drive it and a frame to secure it in place. In this design, actuation is provided by pulley-driven cables. The pulleys, which are turned by servo motors, apply the necessary tensile force to cause the robot to bend. The frame supports the servo motors so that they can apply these forces. The development of the actuation systems and the frame is covered in the following sections.

Actuation

As stated above, the spine needs an actuation mechanism to move. Many possible actuation mechanisms exist, including pneumatics, push rods, and cables. The robot uses cables, representing the simplest and most space-efficient actuation mechanism of the three. A cable system needs two things: cables and a way to apply tension to the cables.

The current design uses Berkley FireLine Super Line Crystal 30 lb fishing line with a diameter of 0.38 mm. This line was chosen due to its strength, small size, and flexibility. Each cable is secured on both ends to prevent the cables from sliding through the frame. Near the robot's tip, the cable is tied into a double square knot; the cable is crimped with a trimmed electrical connector on the end near the pulleys. These can be seen in Figure 2.17. A knot was chosen for the tip, as the Aurora probe is housed at the very tip of the robot. Surrounding the probe with metal can interfere with the magnetic field, and thus corrupt the position measurement. Additionally, the knots take up much less space than the crimp, which is helpful when attaching specialized tips and registration hardware. The other end of the cable was crimped, however, as tying the knots with the cables is incredibly tedious. It is also difficult to tie the knot without lots of extra slack in the system. Because the servos used do not have infinite travel, the rotation used to remove slack takes away from the rotation

(a) (b)

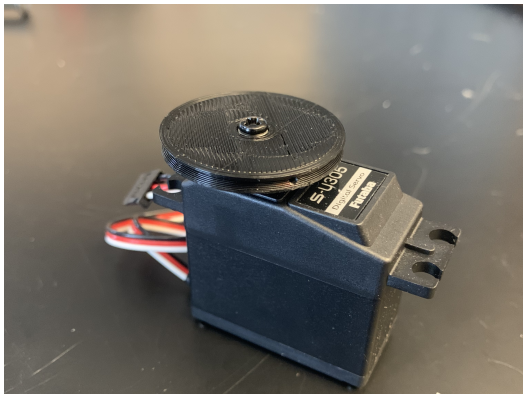
Figure 2.17: Termination methods for the cables. (a) Crimps on pulley end of cables. (b) Knots on tip end of cables.

used to drive the robot.

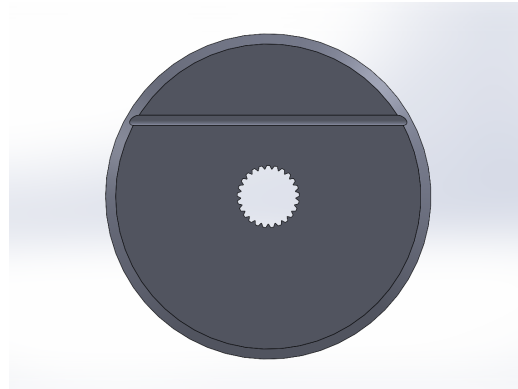
Finally, the cables are attached to the motors using custom pulleys FDM printed using PLA. These pulleys, seen in Figure 2.18 act to guide the cables and allow the rotation of the servos to be translated into linear motion. The pulleys are mounted to the servos via a friction fit with the spline on the servo output shaft. The pulleys contain a channel that stretches around the outside of the pulley, which provides a groove for the cables to sit in. To attach the cables to the pulleys, they are fed through a hole in the pulley and crimped so they cannot be pulled through. This provides a secure attachment to the pulley and helps route the cables into the groove so they do not fall off. This is one of the many features of the robot dedicated to preventing cables from falling off pulleys. The others, which are part of the frame, will be discussed later.

Frame

The robot's frame provides the structure that attaches the spine to the motors and other hardware. Additionally, it handles the routing of the cables used to drive the robot. While this is a relatively simple component, it has two improvements over the previous design. First, it is simplified, removing components like a cable tensioning



(a) A pulley attached to a servo motor. It allows the robot to pull on the cables to move itself.



(b) Section view of the pulley, showing the hole that travels through the pulley to secure the cable

Figure 2.18: Pulley design used by the servos to pull on the cables.

mechanism that added additional sources of error to the robot. Second, it was made much smaller, reducing the frame volume from 5.865 L to 0.661 L, allowing the robot to be more portable. The frame consists of three main components: the top, the motor mounts, and the cable guides, which were laser cut from 3 mm acrylic. The parts were then bonded together using the same Loctite superglue used in the spine. The frame and its components can be seen in Figure 2.19.

The top of the frame is the primary attachment point for the spine. It is connected to the base of the spine using 4-40 bolts and nuts. Additionally, the frame has holes for the cables and the Aurora wire to pass through, matching the hole pattern that runs up the length of the spine. It also has cutouts for the fingers of the motor mounts, allowing them to slot into place and remain square to the top of the frame. One benefit of using this approach is that the top is agnostic to what the motor mount looks like. This means that the top remains unchanged when adding additional segments to the robot, along with the necessary motors to drive that segment. Instead, the walls/ motor mounts accommodate the new components. Finally, a series of twelve registration points are inscribed on the top of the frame. The frame top can be seen in Figure 2.20.

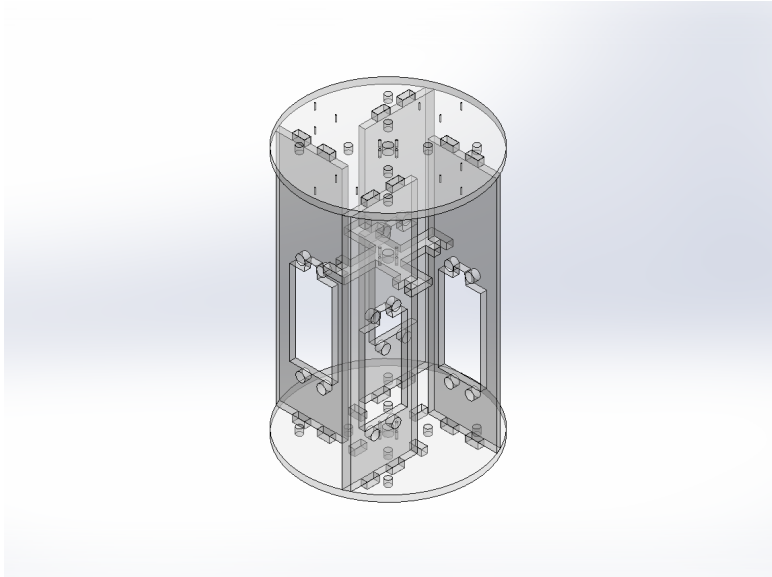


Figure 2.19: The frame assembly. Each component is laser cut from acrylic, assembled, and secured with Loctite superglue.

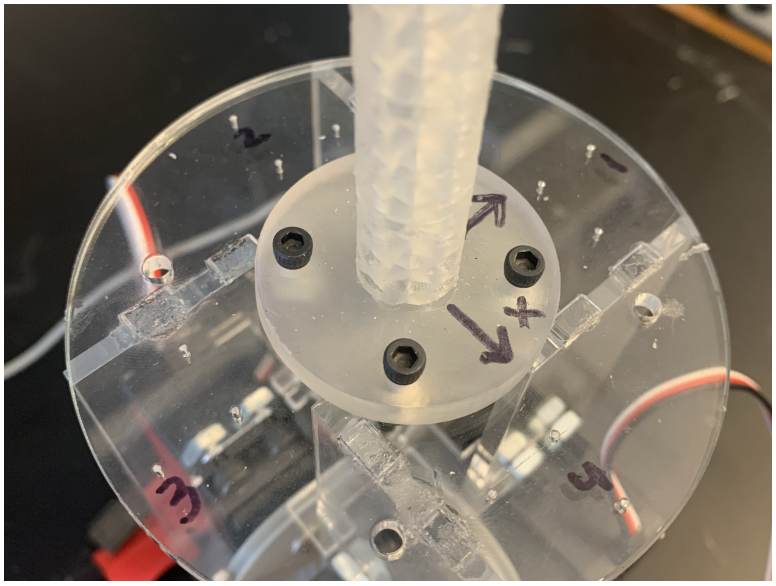


Figure 2.20: The top of the frame serves as the primary attachment point for the spine while allowing cables to pass through.

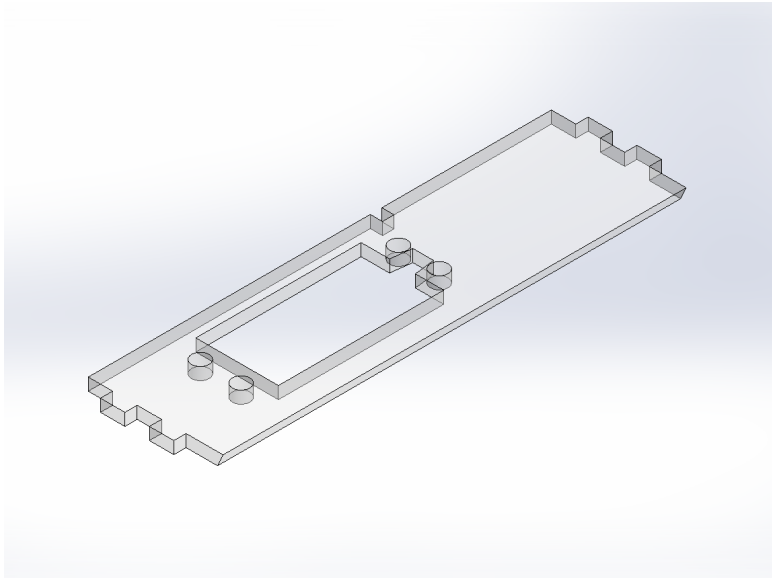
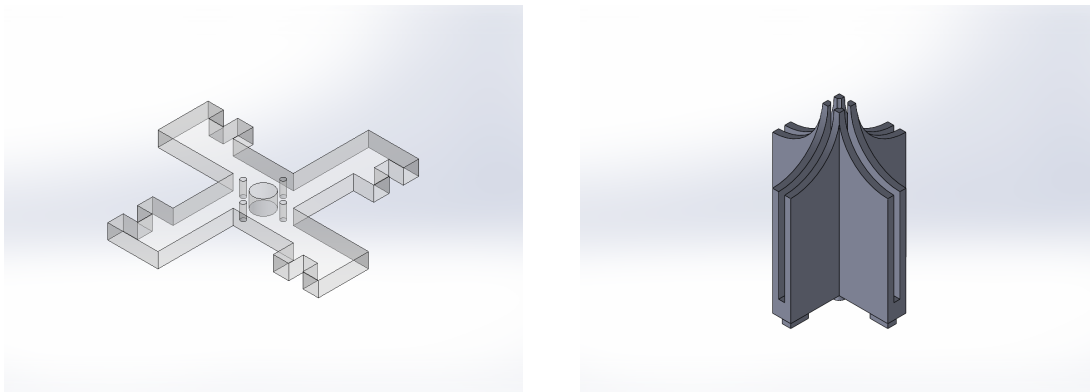


Figure 2.21: The walls of the frame provide a mounting point for the servos and connect the top of the frame to the bottom of the frame.

The motor mounts, which also serve as the walls of the frame, are responsible for securing the motors and aligning them with the cables coming out of the spine. The Comptuer Aided Design (CAD) model of the motor mount can be seen in Figure 2.21. To mount the motors, cutouts are provided for the servo's body to go through, as well as 8-32 clearance holes to secure the motors to the frame. The walls are aligned on the frame so that the pulleys are aligned with where the cables enter the spine. To minimize the effect of the metal in the servos on the Aurora field measurements, the servos were placed as far down the walls as possible, and an additional 30 mm of height was added as a buffer to the top of the walls. To add motors for additional segments, these walls can be lengthened, and a second set of motor cutouts can be placed below the first, providing a new motor attachment point.

In the previous design, the motor mounts also contained a mechanism to tension the cables. This was created because any slack in the cable modifies how the robot behaves, requiring all slack to be removed from the system before it behaves as expected. This was removed from the current design as this same tensioning can be performed electronically using the motors and the Aurora to detect changes in



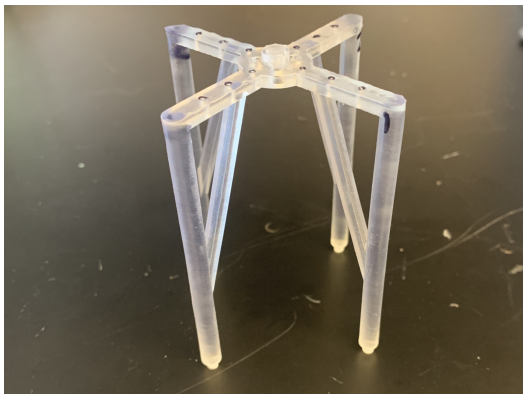
(a) (b)

Figure 2.22: Horizontal and vertical cable guides.

position. This is covered in more depth in Section 2.3.2.

Additionally, there are two sets of cable guides to prevent the cables from falling off the pulleys. The first is made of acrylic and sits above the pulleys in the middle of the four walls. It has the same hole pattern as the top of the frame, and the cables are routed through it to constrain the cables laterally. The second is 3D printed out of rigid resin and is attached to the base of the frame. This vertical cable guide lines up with the edges of the pulleys, which prevents the cables from slipping off the side of the pulley. These two guides work together to prevent the cables from coming off the pulleys, which can occur when slack is in the cables. The CAD models of these two components are shown in Figure 2.22.

Finally, there is one more set of registration hardware that interfaces with the frame. In order to measure the state of the tip of the robot, an Aurora coil is rigidly coupled to the tip of the robot. However, simply using the raw measurements from the Aurora coil is not sufficient to determine where the tip of the robot is. The coil may be rotated such that it does not line up with the axes of the tip frame. Additionally, the coil may not be located at (0, 0, 0) in the tip frame. To determine this, registration is also performed on the tip of the robot. This registration, however, requires that the tip be fixed relative to known points.



(a) The hat structure, with its registration points and feet that slot into the top of the frame.

(b) The end of the robot can slide into the bottom of the hat and is locked in place by friction.

Figure 2.23: The hat, both on its own and coupled to the robot. It provides a rigid structure to perform registration of the tip frame.

To secure the tip and provide known registration points, a fixture was developed that can be secured to the top of the frame and is coupled with the tip of the robot. This fixture, dubbed a hat, also provides touchpoints for registration. This provides an accurate and repeatable way to determine the position of the tip frame, which can then be used to find the transform between the tip frame and the coil frame. The current design of the hat can be seen in Figure 2.23.

As can be seen, the frame, while relatively simple, is a crucial part of the robot. It provides structure, organizing the various components of the robot so that they are aligned and interact properly. By removing unnecessary components and bringing components together, its footprint was significantly reduced: the volume of the structure decreased by 88.7 % (from 5.87 L to 0.66 L), making the system more portable. Additionally, because the frame was designed with an emphasis on modularity, parts are easily swapped out and exchanged for newer designs.

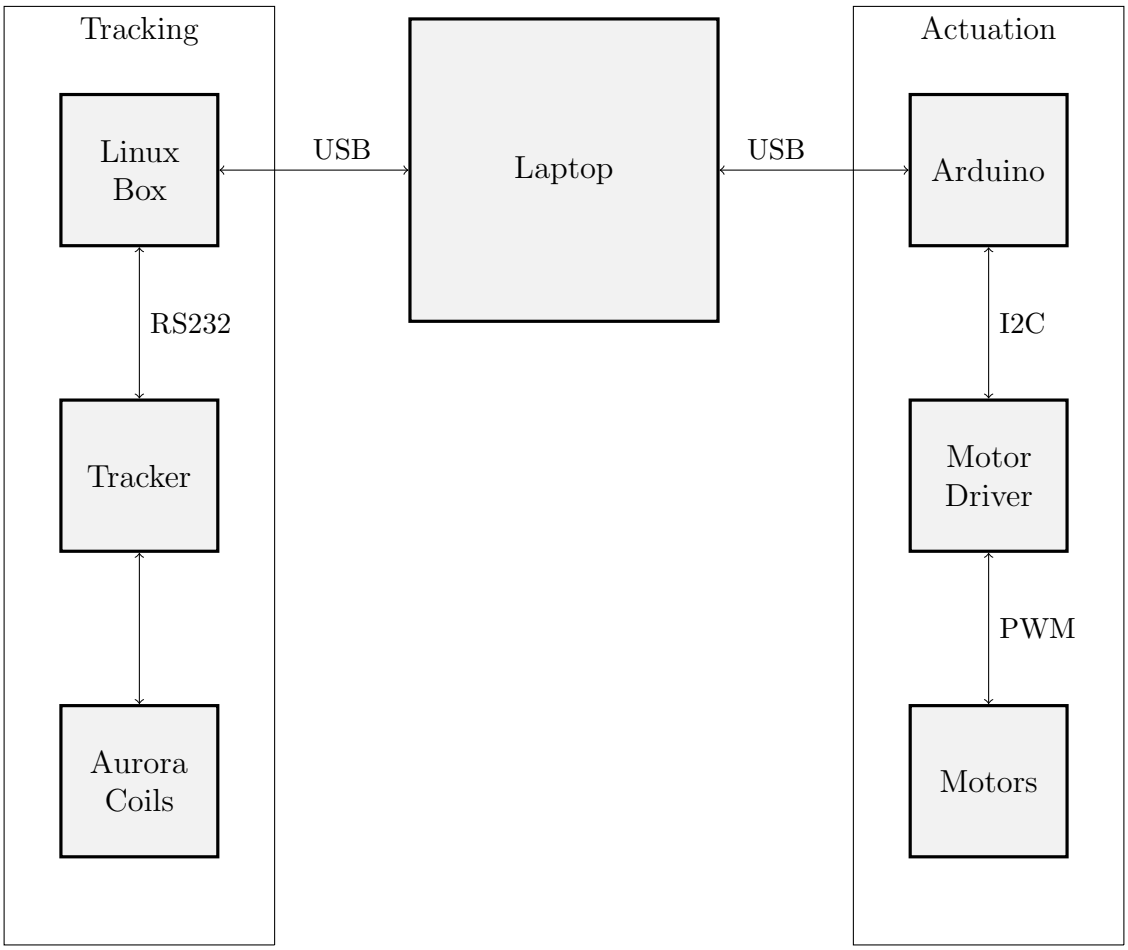


Figure 2.24: High level block diagram of the system. All communication starts from the laptop and travels through the two subsystems: Tracking and Actuation.

2.2 Electrical Design

The electrical components of the system provide the hardware needed to use the robot, giving it motion and the ability to know where it is. The electrical system comprises multiple components working together, but, as can be seen in Figure 2.24, the components can be split into two categories: actuation and tracking. In order for the robot to move, it needs a way to pull on the cables, which is achieved using the actuation system, covered in Section 2.2.1. Additionally, we need a way to figure out where the robot is in 3D space so that it can perform precise maneuvers. This job is performed using the tracking system, covered in Section 2.2.2.

2.2.1 Actuation

The actuation system is what allows the robot to move. It consists of two main categories of components. The first is the motors. The motors perform the actual movement, translating input signals to positions and rotations. The motors used, and the reasons for selection are covered in greater detail in the following section. The motors, however, cannot be controlled on their own. To command the motors, special control hardware, the second category of components, is used. The control hardware translates the setpoints and commands sent to it to the voltages and currents needed by the motors to move.

Motors

To control the robot, there are many motors to choose from. For the scale of our system, the three most common types used are DC motors, stepper motors, and Radio Control (RC) servo motors. DC motors are the most affordable and provide the highest torque of the three types of motors. However, our robot requires position control on its output. While this can be implemented with a DC motor, setting this up and designing the controller for it would have distracted from the development of the rest of the robot, disqualifying DC motors from selection. Stepper motors, on the other hand, provide highly accurate position control and are generally simple to work with. This high level of accuracy, however, comes at the cost of lower torque, a larger form factor, and a higher price. Servos can be thought of as somewhere in the middle between DC motors and steppers. They are DC motors with highly geared-down outputs that contain the hardware needed to provide position control. They are a fully integrated package of the system that we would have needed to develop for a DC motor. Additionally, they are generally much smaller than stepper motors, meaning that they are easier to integrate into small frames. For these reasons, we ultimately selected servo motors to control the robot. More specifically, we selected the Futaba

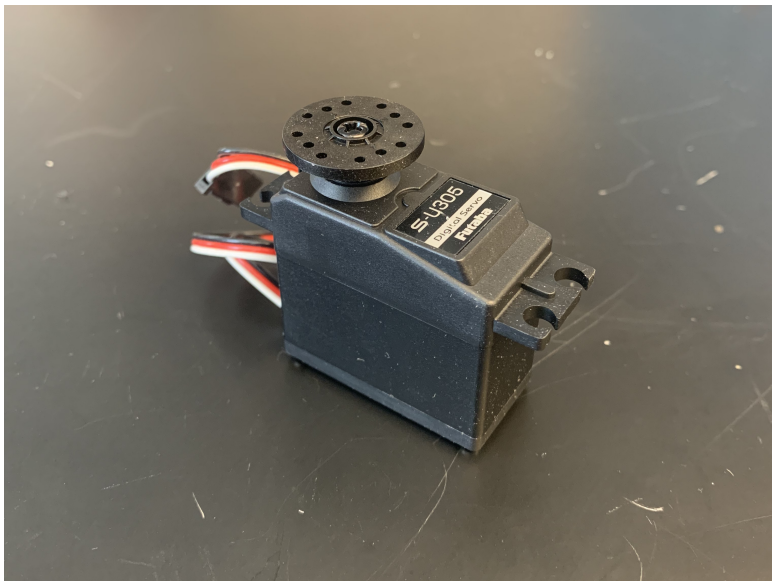


Figure 2.25: The Futaba U-305 Universal servo is a metal geared, digital servo with a sweep of $\pm 60°$.

U-305 Universal servo for this purpose, which can be seen in Figure 2.25.

The Futaba U-305 Universal servo was selected over other servos because it is touted for its reliability. Originally, other servos were used. However, after extended periods of use, these servos would oscillate around their setpoints and did not act reliably. This degradation in performance was unacceptable for a high-precision system, leading to the selection of higher-grade servos with better durability.

Control Hardware

To control a server motor, access to lower-level hardware than a normal computer can provide is generally required. As a result, Microcontrollers (MCUs), small processors with the ability to control the low-level operations of the chip, are generally used. For this project, we selected the Arduino Due, a small, portable MCU based on the ATSAM3X8e ARM Microcontroller, designed for easy programming and use. To control the servo motor, Pulse Width Modulation (PWM) signals and sufficient current are needed. While the Arduino can generate PWM signals, they generally can only generate a small number of independent signals. Additionally, the Arduino

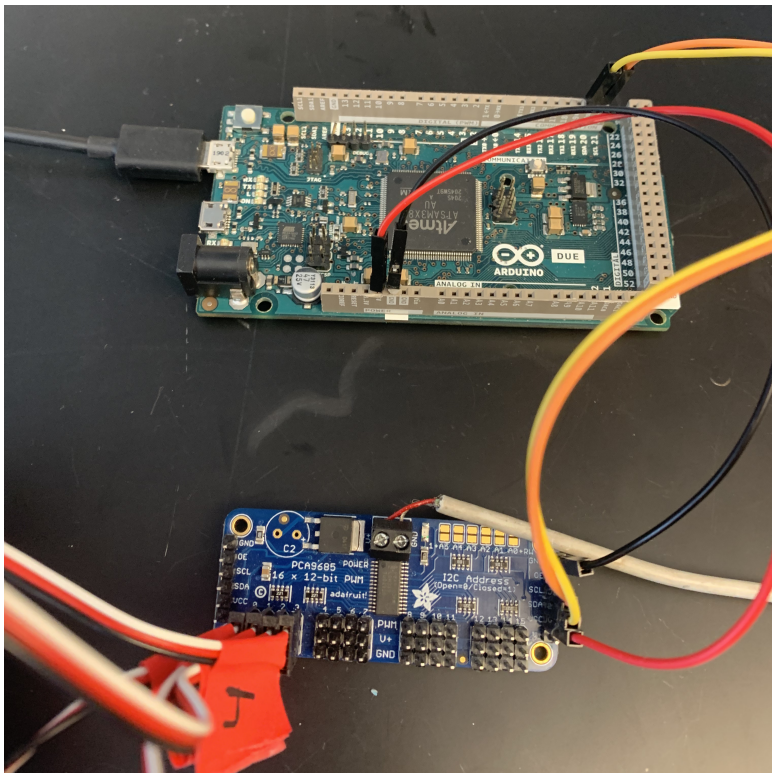


Figure 2.26: The components used to control the servo motors. The motors are connected to the Adafruit 16-channel 12-bit PWM driver. The driver is then connected to an Arduino Due, which can communicate with a computer and send the necessary signals to the PWM driver.

can only source up to 15 mA of current per pin, meaning that connecting the servos directly to the Arduino often causes the chip to brown out and lose power when the motors move. Therefore, it was decided that a dedicated circuit, the Adafruit 16-channel 12-bit PWM driver board, would be used to control the servo motors. This board, which is built on top of the PCA9685 PWM driver, communicates with the Arduino using I2C and can generate up to 16 independent PWM signals, meaning that we can control 16 servos at once. Additionally, it has terminals to connect directly to a power supply, which provides the current needed to drive the servos independent of the Arduino's logic supply. The control hardware, both the Arduino and the PWM driver, can be seen in Figure 2.26.

Figure 2.27: Our system uses the Aurora Planar 20-20 Field generator, along with 6 DOF coils to sense the tip position of the robot.

2.2.2 Tracking

To develop models of the robot and eventually control it, knowledge of the pose of the robot's tip in 3-D space is needed. This was achieved using the Aurora, by NDI, which is a tracking system that uses a magnetic field generator and coils that sense the magnetic field to determine the position and orientation of the coil relative to the Aurora. The Aurora system, shown in Figure 2.27, can track up to four coils simultaneously, and each coil can be integrated into different parts of the robot. In its current configuration, only one coil at the tip is used, but more could potentially be used along the length of the spine. This would provide better information about the shape that the robot assumes and could be an interesting area for further research.

2.3 Software Design

The robot's software provides a way to control and interact with it. There are two important details to highlight regarding the system software. The first is the communication protocol developed to communicate between a laptop, and the Arduino

used to control the system robustly. The second is software, which is used to tension the pulleys automatically. These are covered in Sections 2.3.1 and 2.3.2, respectively.

2.3.1 Serial Communication

Because the system is made up of multiple separate components, namely the laptop, the Arduino, the Linux box, and the NDI Aurora, a communication scheme between each of the components, radiating outward from the laptop, is needed. The scheme used for communication with the Arduino underwent significant development from its form in the Leigh system and is covered in the subsequent section. Following that, a discussion of the current communication protocol with the Linux box and Aurora is discussed, as well as its latency due to going through the Linux box for all interactions with the Aurora.

Arduino Communication

To control the system from a computer, there is the need for the ability to send position commands to the Arduino. In the past, this was achieved by sending the raw motor commands over Serial and using those values to control the motors. While this system worked, it was not robust. Errors in the transmission meant that incorrect commands were sent to a motor or the wrong motors received commands. Additionally, with this simple communication scheme, the system could not correct itself after a single missed byte, as the entire packet would be shifted as a result, jumbling the data sent. Because the ultimate goal is for this system to be used in surgical settings, the system's fragility and lack of robustness were unacceptable.

To address this issue, a Serial communication protocol based on the Binary Synchronous Communication (BSC) protocol [52] was implemented. This protocol, which is based on a set of control characters, allows for the encoding of information, such as the start of a packet, whether a packet was received correctly, and many other

Command Characters	
Name	Byte (Hexadecimal)
Data Link Escape (DLE)	0x10
Start Transmission (STX)	0x02
End Transmission (ETX)	0x03
Acknowledge (ACK)	0x06
Error (ERR)	0x15

Table 2.1: Command characters used in communication with Arduino.

Example Data Packet							
DLE	STX	Flag	Data Length	Data	CRC	DLE	ETX
0x10	0x02	0x05	0x08	...	0x3A	0x10	0x03

Table 2.2: An example of the packet sent to the Arduino to control the motors. The command for each individual motor is encoded into two bytes, which are sent in the data section of the packet.

important properties. This is achieved using a series of control characters encoded as unique bytes. The bytes selected can be seen in Table 2.1. Looking at the STX byte, however, it is impossible to distinguish it from data. Therefore, all control characters are preceded by the DLE byte. This allows for control characters to be recognized and used to synchronize communication.

However, what if the data sent happens to contain the same byte as a DLE? Without handling this case, the system would think of the following data byte as a control character. To avoid this, DLE stuffing is used, where any time a DLE is encountered in the data, an additional DLE byte is inserted directly following it. This allows for the receiver to distinguish between an actual DLE and standard data. An example packet can be seen in Table 2.2. Each packet comprises a start sequence, a flag indicating the type of packet being sent (shown in Table 2.3, the number of bytes in the data payload, the data itself, a Cyclic Redundancy Check (CRC) [53], and finally, an end sequence.

The CRC uses repeated Exclusive Ors (XORs) on the packet's payload (all bytes excluding the start and end sequences) to ensure the packet's integrity. The current

Transmission Flags for Arduino Communication	
Name	Byte (Hexadecimal)
Motor Command (CMD)	0x04
Communication (COM)	0x05
Number of Motors (NUM)	0x07
Oscillator Frequency (OHZ)	0x08
Servo Frequency (SHZ)	0x09

Table 2.3: Transmission flags, which inform the Arduino as to the type of packet sent, are sent in a byte after the start sequence.

system implements the CRC-8 algorithm (shown in Algorithm 2.1), which has the ability to detect all burst errors of length less than or equal to eight, as well as catching longer bursts with an approximate probability of 99.6% [54].

```
def crc_add_bytes(CRC: int, payload: bytearray):
    """
    Calculates a CRC-8 checksum given a bytearray.

    Args:
        CRC: Starting CRC value (Set to 0 for transmission)
        payload: The byte array for which we are generating
        the CRC

    Returns:
        The CRC of the payload
    """

    CRC = CRC & 0xFF  # Ensure CRC is treated as uint8
    for byte in payload:
        for bit_num in range(8, 0, -1):
            thisBit = (byte >> (bit_num - 1)) & 1
```

```
        doInvert = (thisBit ^ ((CRC & 128) >> 7)) == 1
        CRC = (CRC << 1) & 0xFF  # Ensure result is uint8
        if doInvert:
            CRC = CRC ^ 7
return CRC
```

Algorithm 2.1: Python implementation of CRC-8 algorithm

Error Codes for Arduino	
Name	Byte (Hexadecimal)
Length mismatch	0x01
CRC mismatch	0x02
Unknown flag	0x03
System unitialized	0x04
Motor number mismatch	0x05
Incorrect parameter length	0x06
Invalid parameter	0x07
Invalid communication	0x08

Table 2.4: Error codes aid in debugging, as they give the user an indication of what went wrong on the Arduino.

Finally, after receiving a packet, the receiver parses it, and if it is received correctly, the receiver sends an ACK packet back to the transmitter. Otherwise, it transmits an ERR packet, along with an error code describing what went wrong with the packet. These error codes are shown in Table 2.4. The transmitter waits for this signal and will send another packet on receiving an ERR packet or no response.

This communication protocol allows for robust communication, ensuring that the correct commands are sent and that no transmission errors occur. In the case that they do occur, however, the system can detect the errors and request that the data be retransmitted. This guarantees that the system acts as expected, enhancing the safety of a patient and the system itself.

Aurora Communication

The Aurora utilizes a similar communication scheme, also built on top of BSC. However, all communication between the laptop and the Aurora is routed through a Linux box, which acts as a relay. This was initially put in place during previous phases of the project, as it was thought that there would be no laptop performing calculations, and instead, all computation would take place on the Arduino. However, the amount of calculation required and the optimization needed to implement it on a microcontroller has necessitated the use of the laptop.

Whenever the system needs a transform from the Aurora, a packet is first sent from the laptop to the Linux box requesting the transform data from the Aurora. The Linux box processes this request, builds a packet based on the request, and then sends a packet to the Aurora. The Aurora parses this, collects a measurement, packages it, and returns it to the Linux box. This is then parsed by the Linux box, repackaged, and finally sent to the laptop. All of this extra communication overhead contributes to unnecessary latency in the system.

The Aurora can collect and transmit measurements at a rate of 40 Hz. To test the performance of the current system, many measurements were requested and timed to see how long, on average, requesting data took. This is shown in Figure 2.28. On average, each request took 185 ms, corresponding to a frequency of around 5 Hz. This highlights the costs of the redundancy, and remedying it is an important future step in the development of the system.

2.3.2 Cable Tensioning

While commands can be sent to the robot using the above protocol, they are just for arbitrary motor positions. To be able to control the robot, a method is needed to determine when the cables are under tension to effectively utilize the whole actuation space of the robot. In the past, this was done by hand using a set of tensioning pulleys,

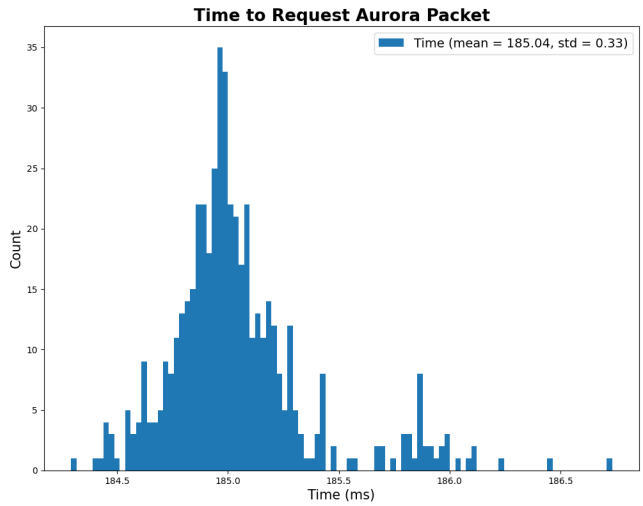


Figure 2.28: A histogram showing the average time required to request and receive a packet. This shows that there is significant room for improvement.

which could be positioned to make the cables taught. This process, however, lacked accuracy, as the only way of knowing if the cable was tight was by looking at it.

For this design, a cable tensioning algorithm was implemented, using a change in the robot's tip position to indicate that the cable was under tension. To start, all cables are slack, and the robot's tip position is measured. Then, each motor individually slowly rotates its pulley, taking a small step and pulling on the cable. After each step, the position of the robot is measured and compared with the initial position. If the displacement between the two positions exceeds a predefined threshold, then that cable would be considered tight, the displacement recorded, and the cable loosened again. Data collected during the tensioning routine can be seen in Figure 2.29.

As can be seen, the tensioning routine acts in a repeatable manner. There is a period of relatively little movement, which is followed by a marked increase in the rate of tip displacement vs cable displacement. To improve this algorithm, instead of determining that a cable is tight looking at the raw displacement, one could instead look for a change in the slope of the tip displacement vs cable displacement graph

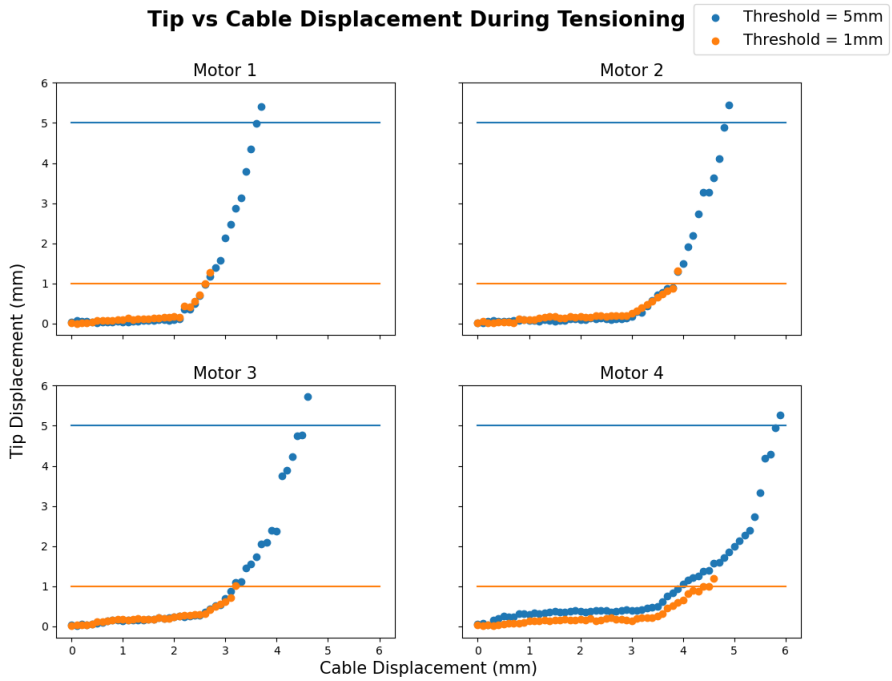


Figure 2.29: Tip displacement as a function of tendon displacement for all four motors during tensioning. Different thresholds can be used (1 mm vs 5 mm), but the behavior is the same

using a finite difference approximation for the slope. This appears to be a better signal that the cable is under tension. However, the current implementation provides a repeatable way to tension the cables to the correct length, establishing a set point for motor positions.

2.4 Conclusion

The design of a continuum robot encompasses mechanical, electrical, and software design. Each one of these three systems must work together in harmony to produce a working final product. From the mechanical perspective, designing a continuum robot is a balancing act. On one hand, a flexible spine is needed that can move and reach every part of the workspace. On the other hand, the spine must be rigid enough to support its own weight and remain in the same place after traveling there. Finally,

it needs to strike this balance continually throughout the life of the robot. As such, much work was done in this area, and there is still much more that can be done here. Possible areas for future inquiry include materials testing for the spine, redesign of the flexible sections, and improvements to frame robot coupling.

For the flexible sections of the robot, Formlabs 50A elastic resin was used as it was used previously in this project [34], and it performed well enough to not require changing it. Additionally, due to the need for significant design iteration, the flexibility of 3D printing made using this material an obvious choice. However, as the section design matures, investigating other materials and manufacturing methods could be interesting. For example, the parts could be cast using silicone or other flexible materials, allowing one to tweak the material properties to optimize spine performance. This would involve defining what optimal spine performance means, such as resistance to transverse loads, the force required to achieve a desired bending radius, and properties that affect the transient response of the system undergoing an external load.

Additionally, improvements could be made to the frame of the robot. For example, providing a more precise kinematic coupling between the spine and the frame than bolts could be useful. Because there is a need for some clearance for bolts to pass through, the spine can translate slightly in its position on the frame. This is compensated for later in the model, but not having this issue in the first place would be incredibly beneficial. This could be achieved using features that precisely couple the spine's base to the top of the frame.

Finally, as stated before, there are improvements that could be made to the low-level software of the system. For example, the tensioning routine, while an improvement over the by-hand tensioning, is not without its drawbacks. It is relatively repeatable, but it is hard to determine what the tip displacement threshold should be. Too low, and the cables still have some slack. Too high, and the spine will be under

constant compression. Tests involving multiple displacement thresholds could be run to determine how much tension is optimal.

Chapter 3

Modeling and Control

Robot development generally occurs in two distinct yet overlapping phases. The first phase is concerned mainly with developing the physical robot. This development involves the design of actuators and electrical hardware that will be used to move the robot around and provide it power to do so. The second phase is concerned with taking the robot that has been built and developing models and control architectures to extract the robot's utility.

To develop accurate models and controllers for robots, the data collected must be in a usable format and of sufficient quality to unearth the relationships that govern a robot's behavior and dynamics. Even with an Aurora coil in the tip of the robot, the data it produces cannot be used until knowledge of how the Aurora coil relates to the rest of the system is determined. This is achieved using a technique known as registration, which involves the translation of known positions in one frame to known positions in another. Registration in the context of our continuum robot system is covered in Section 3.1.

After registering the Aurora frame to the various frames of the robot, data collection can begin in earnest. With this, an investigation into existing models of continuum robots is performed. Due to their compliant nature and infinite degrees of freedom,

modeling approaches in the past have generally been characterized by simplifying assumptions to make the problem of modeling the system tractable. One of the most common types of model for continuum robots is known as a constant curvature model, which assumes that the robot bends in constant curvature arcs. While this obviously simplifies the system's kinematics, they provide a good intuition about how continuum robots function.

In section 3.2.1, two constant curvature models are developed and tested. Later, they are compared with other models in Section 3.2.3. These constant curvature models represent more classical attempts at developing a model for a continuum robot. However, this is not the only way to develop a model of a continuum robot. Instead of analytically deriving a model of a continuum robot, a model can instead be learned from data. This approach, which utilizes universal function approximators, is covered in Section 3.2.2 and tested in Section 3.2.3.

After developing and testing these models, they are applied in the development of controllers. This is covered in Section 3.3. This chapter discusses the difference between open-loop and closed-loop controllers and highlights how each can be applied to the control of a continuum robot. These controllers are then tested using models developed in simulation and models based on the real robot to control the robot and have it follow a representative trajectory through its workspace. The performance of these controllers is covered in 3.3.3.

3.1 Registration

In order to develop a functional model of the robot, a known reference is needed for all of the pose measurements captured. This reference must remain the same, regardless of the robot's relative position and the Aurora field generator. Therefore, a way is needed to translate the measurements from the Aurora to this reference

frame, called the Model frame. This can be achieved using a technique known as rigid registration.

Rigid registration is the process of taking a set of points in one frame and then performing a rotation and a translation to line up the points with the same set in another frame [55] [56]. More formally, let $x \in \mathbb{R}^{3\times n}$ represent the set of n points in the unregistered frame U, and $y \in \mathbb{R}^{3\times n}$ represent the same set of points in the model frame M. For each point x_i, y_i, they can be related by

$$y_i = R_U^M x_i + t + n_i \tag{3.1}$$

Where $R_U^M \in \mathrm{SO}(3)$ is a rotation, $t \in \mathbb{R}^3$ is a translation, and n_i represents noise. The goal is to find an R_U^M and t that minimizes the squared error loss function, seen in Equation 3.2, between the registered and model points.

$$L = \sum_{i=1}^{n} \left| y_i - (R_U^M x_i + t) \right|^2 \tag{3.2}$$

Let $\bar{x}$ be the mean of x and $\bar{y}$ be the mean of y. Each point x_i, y_i can be written as the sum of the mean of the set of the points and its position relative to the mean.

$$x_i = \bar{x} + x_i' \tag{3.3}$$

$$y_i = \bar{y} + y_i' \tag{3.4}$$

Plugging this into the loss function allows for decoupling rotation and translation, as the means are constant.

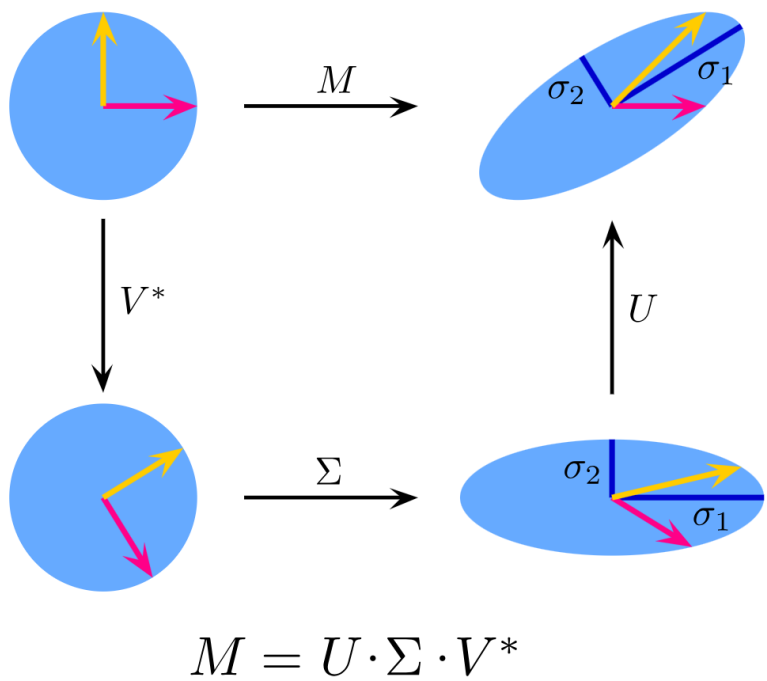


Figure 3.1: A visualization of SVD. Georg-Johan / CC-BY-SA 3.0 [57].

$$L = \sum_{i=1}^{n} \left| y'_i - R_U^M x'_i + \bar{y} - R_U^M \bar{x} - t \right| \tag{3.5}$$

$$t = \bar{y} - R_U^M \bar{x} \tag{3.6}$$

$$L = \sum_{i=1}^{n} \left| y'_i - R_U^M x'_i \right| \tag{3.7}$$

The covariance matrix $C_{x'y'} = x'(y')^T$ represents the correlation between x' and y'. Because the only difference between the point clouds is rotation, this covariance matrix can be used to determine the rotation between the two. Using SVD produces

$$C_{x'y'} = U \Sigma V^T \tag{3.8}$$

U represents a rotation, Σ scales by the singular values, and V^T is another rotation. This relation can be seen in Figure 3.1.

Therefore, the rotation between the two point clouds can be found by taking the

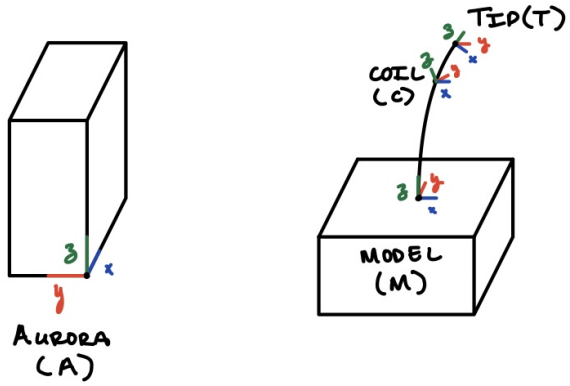


Figure 3.2: The various reference frames in the continuum robot system.

SVD's rotational components and disregarding the scaling term Σ. However, it is important to note that the rotations represented by U and V^T are active rotations. Therefore, the final result is transposed (inverted) to get the proper passive rotation, resulting in Equation 3.9.

$$\hat{R}_U^M = VU^T \tag{3.9}$$

The transformation matrix T_U^M can then be assembled as shown below.

$$\hat{T}_U^M = \begin{bmatrix} \hat{R}_U^M & \hat{t} \\ 0^{1\times 3} & 1 \end{bmatrix} \tag{3.10}$$

There are four main frames in our system: the Aurora frame (A), the Model frame (M), the Coil frame (C), and the Tip frame (T). These frames can be seen in Figure 3.2. To match any models developed, the pose of the robot's tip relative to the model, the transformation T_T^M, is needed. By taking measurements from the Aurora, T_C^A is measured. Therefore, to get T_T^M, the following transformation is used.

$$T_T^M = T_A^M T_C^A T_T^C \tag{3.11}$$

Therefore, T_A^M and T_T^C must be determined by registration. For the Aurora to Model transform, twelve points are taken along the top of the frame, which are then registered using the above algorithm. This provides T_A^M. For the Tip to Coil transform, however, there is no way of directly registering that transform like above. Therefore, an intermediate frame is needed. Much like the Model frame, the Tip frame can be registered to the Aurora frame as long as it is stationary, producing T_A^T. This is achieved using the Aurora "hat" described in Section 2.1.2. Additionally, while each point is collected for the tip registration, the Coil produces a transform T_C^A. However, there is some noise associated with this transformation. Therefore, all of the gathered transforms can be averaged together to better estimate the true tip transformation.

As a transformation is composed of a rotation and a translation, each part must be averaged independently. This is quite simple for the translation, as a standard average can be performed. However, it is not that simple for the rotation matrix. A rotation matrix must remain orthogonal and right-handed to represent a valid rotation. Averaging, however, gets rid of this property. This is overcome by converting the rotation matrices into quaternions. A quaternion $q = q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}$ that is unit length can be used to represent a rotation of θ about a vector $\mathbf{v} = v_x\mathbf{i} + v_y\mathbf{j} + v_z\mathbf{k}$, where

$$
\begin{aligned}
q_0 &= \cos\left(\frac{\theta}{2}\right) \\
q_1 &= \sin\left(\frac{\theta}{2}\right) v_x \\
q_2 &= \sin\left(\frac{\theta}{2}\right) v_y \\
q_3 &= \sin\left(\frac{\theta}{2}\right) v_z
\end{aligned}
$$

are the components of the quaternion. While quaternions are not additive, i.e., the compound rotation $q_{1,2}$ represented by q_1, then q_2 is not $q_1 + q_2$. However, for infinitesimal rotations, $q_1 + q_2 \approx q_{1,2}$ so long as it is normalized after the addition. Because all of the Coil to Aurora transforms are basically the same, the quaternions for each are roughly the same. Therefore, averaging them is akin to performing a slight rotation halfway between the two orientations. While there are more accurate algorithms for averaging quaternions, such as in [58], the above method is sufficient for similar quaternions.

Therefore, to find the average rotation between the Coil frame and the Aurora frame, the transformations are taken in their quaternion form, averaged, renormalized, and then converted back to a rotation matrix. This results in an average Coil to Aurora transform

$$
\hat{T}_C^A = \begin{bmatrix} \hat{R}_C^A & \hat{t}_C^A \\ 0^{1\times 3} & 1 \end{bmatrix} \tag{3.12}
$$

which can be used to determine

$$
T_T^C = (T_C^A)^{-1}(T_A^T)^{-1} \tag{3.13}
$$

which produces the final transform needed. Now, the necessary transformations between the system frames can be performed, providing the pose of the robot's tip

relative to the Model frame.

3.2 Modeling

Due to their infinite DOFs, continuum robots require more complex modeling architectures than standard robots. While there are various approaches for producing a model of a continuum robot, they can be broken into two categories: analytical and learned models. Analytical models are derived from first principles and use simplifying assumptions to make the math tractable. The most ubiquitous of these is constant curvature modeling, which assumes that the segments of the continuum robot bend in circular arcs [10]. These models have been formulated using both pure kinematics [18] [59] and statics [41] [17] [60]. While these are simple and easy to work with, their accuracy is relatively low, prompting the development of more advanced models, such as Cosserat rod models [61][21][62] or FEM based models [22][63], that depart from the constant curvature assumption. However, these models end up being much more complicated than their constant curvature counterparts and often require significant computation to predict the pose of the robot. While these issues are not insurmountable, they constrain these models' implementation in real-world systems.

On the other hand, learned models instead treat the continuum robot system as a black box and look to find an input and output model that represents the system. These models take data collected from a real-world system and use it to create a lightweight model that can predict the pose of the robot's tip. This has been used to learn both kinematics models [64] and dynamics models [65] with high degrees of accuracy. However, this model-free, learned approach is not a panacea. While learned models are significantly less complex to develop, they require large amounts of data to train [66]. In other fields, this has led to the adoption of techniques such as pretraining [67] and fine-tuning [68]. However, it has seen relatively little coverage in continuum

robots.

What follows is an investigation of the two types of modeling: analytical and learned. The analytical models will be a collection of constant curvature models, with an analysis of their performance. These constant curvature models will then be compared with learned models, looking at their performance in estimating the tip pose of the robot.

3.2.1 Constant Curvature Modeling

Constant curvature models serve as an excellent introduction to the kinematic modeling of continuum robots. They are built on the assumption that the robot bends in a series of smooth, continuous arcs [10], which correspond to the segments of the robot. This constant curvature bending can be seen in Figure 3.3. This allows for the development of simple models, which, to first order, can predict the pose of a continuum robot's tip. While this constant curvature assumption does not hold for any real-world continuum robot due to variable material properties, friction, and external loads, it represents one of the first attempts at modeling the complex kinematics of an infinite DOF continuum robot [18].

A constant curvature model can be broken down into two parts: a robot-specific and a robot-independent part, highlighted in Figure 3.4. Because the continuum manipulator is being modeled using a series of constant curvature arcs, each can be uniquely characterized using a length, a curvature, and a rotation about the z-axis (ℓ, κ, ϕ). These robot-independent arc parameters, which are determined by the geometry and actuators of the robot, can then be converted into a transformation matrix $T \in \mathrm{SE}(3)$, which represents the pose of the tip of that arc relative to its base. This is done using D-H parameters, described in Table 3.1, resulting in the transformation seen in Equation 3.14 [10].

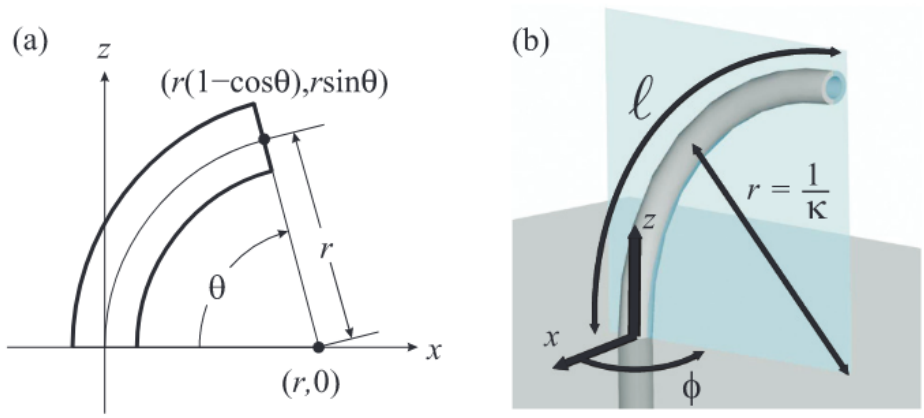


Figure 3.3: The parameters that characterize the assumed constant curvature arcs. ℓ is the length of the segment, κ is the curvature of the segment, and ϕ represents a rotation about the z-axis. When $\phi = 0$, the arc lies in the x-z plane, shown in (a). Figure from [10].

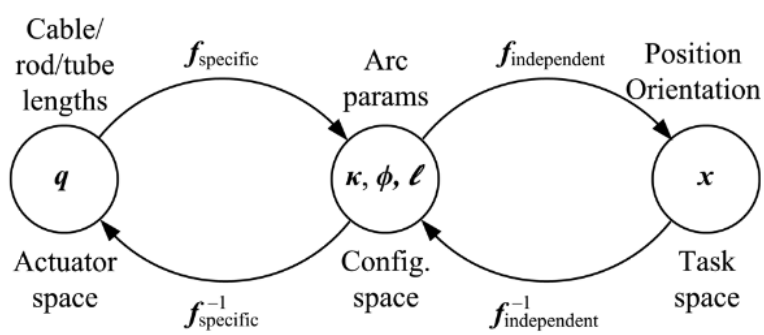


Figure 3.4: The mappings and spaces associated with a continuum robot. The actuators (motors) drive the robot into constant curvature arcs described by the arc parameters. These parameters can then be used to model the tip pose of the robot. Figure from [10].

DH Parameters for Constant Curvature Arc				
Link	θ	d	a	α
1	ϕ	0	0	$-\pi/2$
2	$\kappa\ell/2$	0	0	$\pi/2$
3	0	$(2/\kappa)\sin(\kappa\ell/2)$	0	$-\pi/2$
4	$\kappa\ell/2$	0	0	$\pi/2$
5	ϕ	0	0	0

Table 3.1: The DH parameters used to construct a transform from the segment's tip frame to the base frame [10].

$$T_T^B = \begin{bmatrix} \cos^2\phi(\cos\kappa\ell - 1) + 1 & \sin\phi\cos\phi(\cos\kappa\ell - 1) & \cos\phi\sin\kappa\ell & \frac{\cos\phi(1-\cos\kappa\ell)}{\kappa} \\ \sin\phi\cos\phi(\cos\kappa\ell - 1) & \cos^2\phi(1-\cos\kappa\ell) + \cos\kappa\ell & \sin\phi\sin\kappa\ell & \frac{\sin\phi(1-\cos\kappa\ell)}{\kappa} \\ -\cos\phi\sin\kappa\ell & -\sin\phi\sin\kappa\ell & \cos\kappa\ell & \frac{\sin\kappa\ell}{\kappa} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.14}$$

Unsurprisingly, each robot requires its own robot-specific model. These models have many different formulations. However, they can generally be divided into two categories: pure kinematics models and statics-based models. The pure kinematics models are usually the simplest but cannot account for factors like axial compression and cable stretching. The following sections will contain an example of each type of model.

Pure Kinematics Model

As stated above, an arc is characterized by its length ℓ, curvature κ, and plane rotation ϕ. However, this is not the only set of parameters that achieves this goal. For the following model, which is based on the model from [34], the parameter set (ℓ, θ, ϕ)

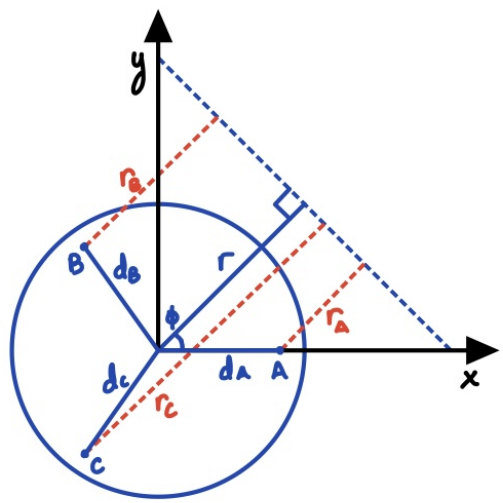


Figure 3.5: The bending radius of each cable is based on its position relative to the center of the spine.

is instead used. θ is related to κ by

$$\theta = \ell\kappa \tag{3.15}$$

Because the length of the spine ℓ is already given from the properties of the system, the displacement of at least two cables is needed to determine the amount of bending that the spine undergoes (described by θ), and the rotation of the bending plane from the $x - z$ plane (defined by ϕ). If more cable displacements are known, the combination of cable displacements that produces the greatest amount of bending is used.

Looking at a section view of the robot, shown in Figure 3.5, the location of the cables relative to the center of the spine determines the length of their bending radii. The original bending radius r is defined by Equation 3.16

$$r\theta = \ell \tag{3.16}$$

Let cable A have a position (d_{Ax}, d_{Ay}) relative to the center of the spine. Similarly,

let the position of cable B be (d_{Bx}, d_{By}). The bending radii of the cables, r_A and r_B, can be related to the bending radius of the spine by finding the projected distance of the cable to the center of the spine's arc. The following equations will be in terms of cable A, but the same applies to cable B.

$$r_A = r - (d_{Ax} \cos\phi + d_{Ay} \sin\phi) \tag{3.17}$$

Additionally, the length of each cable ℓ_A and ℓ_B can be related to the original length of the spine by the amount each cable is pulled.

$$\ell_A = \ell + \Delta\ell_A \tag{3.18}$$

Like the spine, the relation of the length of each cable to its bending radius is

$$r_A\theta = \ell + \Delta\ell_A \tag{3.19}$$

Combining Equations 3.16, 3.17, 3.18, and 3.19 yields

$$\Delta\ell_A = -\left(d_{Ax} \cos\phi + d_{Ay} \sin\phi\right)\theta \tag{3.20}$$

As stated above, cable B is governed by the same relationship, meaning

$$\Delta\ell_B = -\left(d_{Bx} \cos\phi + d_{By} \sin\phi\right)\theta \tag{3.21}$$

To isolate ϕ, divide Equation 3.20 by Equation 3.21.

$$\frac{\Delta\ell_A}{\Delta\ell_B} = \frac{d_{Ax} \cos\phi + d_{Ay} \sin\phi}{d_{Bx} \cos\phi + d_{By} \sin\phi} \tag{3.22}$$

and simplify to solve for ϕ

$$\phi = \tan^{-1}\left(\frac{\Delta\ell_A d_{Bx} - \Delta\ell_B d_{Ax}}{\Delta\ell_B d_{Ay} - \Delta\ell_A d_{By}}\right) \tag{3.23}$$

ϕ can then be plugged into either Equation 3.20 or Equation 3.21 to solve for θ. Thus, arbitrary cable displacements can be converted into robot-independent parameters.

While this kinematics model is simple, it also has some shortcomings. Most notably, it is independent of the length of the spine. Therefore, it cannot handle compression of the spine and assumes that it is completely rigid, which is not valid. Additionally, no elongation of cables or other non-rigidity is captured. Because of this, attempts have been made to incorporate elastic behavior into a constant curvature model.

Statics Model

The following model, first proposed for a single segment in [41] and then expanded into multiple segments in 3D in [17], uses elastic relationships to relate axial and bending strains (ϵ_a and ϵ_b) to cable tensions, which can then be linked to cable displacement via the elasticity in the cables. While the cables undergo negligible elongation, it is used to determine the tension in the cable, which can then apply a load to the spine, causing its deformation. Let T be the tension in a given cable. The axial strain is related to the cable tension by an axial stiffness constant k_a

$$\epsilon_a = \frac{T}{k_a} \tag{3.24}$$

This axial stiffness constant captures the elastic deformations that the spine undergoes in the axial direction. In addition to applying a tensile force to the spine, the cable also applies a moment. This moment comes from the cable being a distance d from the center of the spine and is related to the bending strain at a distance x by another stiffness constant k_b

$$\epsilon_b = \frac{dT}{k_b} x \tag{3.25}$$

This bending strain is also related to the curvature of the spine by

$$\kappa = -\epsilon_b x \tag{3.26}$$

Because the axial and bending strains are linear, the total axial and bending strains in the system are a superposition of the effect of each tendon. Therefore, these relationships can be collected into the following matrix equation

$$\begin{bmatrix} k_b & 0 \\ 0 & k_a \end{bmatrix} \begin{bmatrix} \kappa \\ \epsilon_a \end{bmatrix} = \begin{bmatrix} -d_0 & -d_1 & \dots & -d_n \\ 1 & 1 & \dots & 1 \end{bmatrix} \begin{bmatrix} T_0 \\ T_1 \\ \vdots \\ T_n \end{bmatrix} \tag{3.27}$$

$$\tag{3.28}$$

The above equation describes the bending of a spine in a given plane. However, this can be expanded into three dimensions by projecting the curvature κ into the x and y axes, leading to the following form of the equation. Additionally, the signs of the cable positions can be adjusted to make κ_x represent curvature in the direction of the x-axis instead of about the x-axis.

$$\begin{bmatrix} K_b & 0 & 0 \\ 0 & K_b & 0 \\ 0 & 0 & K_a \end{bmatrix} \begin{bmatrix} \kappa_x \\ \kappa_y \\ \epsilon_a \end{bmatrix} = \begin{bmatrix} d_{0x} & d_{1x} & \dots & d_{nx} \\ d_{0y} & d_{1y} & \dots & d_{ny} \\ 1 & 1 & \dots & 1 \end{bmatrix} \begin{bmatrix} T_0 \\ T_1 \\ \vdots \\ T_n \end{bmatrix} \tag{3.29}$$

$$K_m q = D\tau \tag{3.30}$$

The parameters used to characterize a segment in this model, namely κ_x, κ_y, and ϵ_a, can be related to the standard (ℓ, κ, ϕ) parameter set (referred to as Webster parameters going forward) as follows.

$$\ell = \ell_0(1 - \epsilon_a) \tag{3.31}$$

$$\kappa = \sqrt{\kappa_x^2 + \kappa_y^2} \tag{3.32}$$

$$\phi = \tan^{-1}\left(\frac{\kappa_y}{\kappa_x}\right) \tag{3.33}$$

In its current form, the model is not that useful as it relies on knowing the cable tension in the system. The tension force, however, can be related to the strain in a cable by a stiffness constant k_t, assuming that it behaves elastically.

$$\epsilon_t = \frac{T}{k_t} \tag{3.34}$$

This stiffness coefficient can be estimated from the dimensions and material properties of the cable, namely the Young's modulus G and the area A

$$k_t = GA \tag{3.35}$$

By changing the length of the tendon by $\Delta\ell$, tension is applied to it, which is

balanced by the stresses experienced by the spine. Let y be the collection of the cable length changes.

$$y = \begin{bmatrix} \Delta\ell_0 \\ \Delta\ell_1 \\ \vdots \\ \Delta\ell_n \end{bmatrix} \tag{3.36}$$

This combined tension and elongation can be collected into the following equation, where L_0 is a diagonal matrix containing the original length of the spine for each component of q, L_t is the original length of the tendons, and K_t is a diagonal matrix containing the stiffness constant of each cable.

$$y = D^T L_0 q + L_t K_t^{-1} \tau \tag{3.37}$$

$$= \left(D^T L_0 K_m^{-1} D + L_t K_t^{-1}\right) \tau \tag{3.38}$$

$$= C_m \tau \tag{3.39}$$

Thus, the cable displacements can be related to the tension in the cables. Finally, Equations 3.30 and 3.39 can be combined to remove cable tension and relate displacement to the robot's bending.

$$y = C_m D^+ K_m q \tag{3.40}$$

$$y = Aq \tag{3.41}$$

As can be seen, this requires calculating the pseudoinverse D^+ of D. This is because, unless there are only three cables in the system, the system is over-defined. The simplest way to achieve this is using the Moore-Penrose pseudoinverse, which

produces the minimum norm solution of the equation [69]. Other strategies, such as calculating the solution with the lowest maximum cable tension, are seen in [41].

The Camarillo model works with the convention that a positive cable displacement implies pulling on the cable, whereas the pure kinematics (Mike) model uses a negative cable displacement. Therefore, the cable displacements are negated before being run through the model to match the behavior of the pure kinematics model.

The overall form of the equation does not change when this model is extended to multiple segments. Special consideration, however, must be applied to the D and L_t matrices. Because the cables for distal segments pass through proximal segments, the cables affect each segment until they are terminated at the end of their segment. Therefore, the overall D matrix is structured to consider this fact by collecting all of the individual segment D matrices into an upper triangular matrix.

$$D = \begin{bmatrix} D_0 & D_1 & \dots & D_m \\ 0 & D_1 & \dots & D_m \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & D_m \end{bmatrix} \tag{3.42}$$

Similarly, the cable lengths L_t must be updated to consider the length of each segment they pass through.

$$L_t = \begin{bmatrix} \mathrm{diag}(\ell_0) & & & 0 \\ & \mathrm{diag}(\ell_0 + \ell_1) & & \\ & & \ddots & \\ 0 & & & \mathrm{diag}(\sum_{i=0}^{m} \ell_i) \end{bmatrix} \tag{3.43}$$

Thus, this model proposed by Camarillo et al. provides a way to model constant curvature bending and compression of the spine in multiple segments. However, in its current form, it still cannot handle one factor: Slack cables. Currently, there

is no restriction on whether cables can push or pull, in other words, whether the tension in the cable is negative or positive. However, a negative tension is impossible, as the cable goes slack with zero tension. To remedy this, Camarillo proposes an optimization-based protocol to eliminate the slack from the cable. Let y be the actual length of the cable, $\hat{y}$ be the measured length of the cable, and δ be a correction factor.

$$y = \hat{y} + \delta \tag{3.44}$$

To determine the values of the correction factor, the following function, based on the definition of the tension in a cable Equation 3.39 is minimized:

$$\min_{\delta} \delta^T C_m^{-1}(\hat{y} + \delta) \tag{3.45}$$

Subject to the following constraints

$$C_m^{-1}(\hat{y} + \delta) \geqslant 0 \tag{3.46}$$

$$\delta \geqslant 0 \tag{3.47}$$

For all slack cables, the tension in them should be zero, which comes from $C_m^{-1}(\hat{y}+\delta)$, and for all cables under tension, δ should be zero. Therefore, by minimizing the above function, a configuration is found that models slack cables in the forward kinematics of the system. This leads to the final Camarillo model, which can handle multiple segments, multiple cables, and slack cables.

Parameter Fitting for Statics Model

The above statics model proposed by Camarillo et al. requires knowledge of the three stiffness coefficients k_t, k_a, and k_b. As stated above in Equation 3.35, k_t can

Fitted Camarillo Parameters	
Parameter	Value
k_a	801.200
k_b	4969.148
k_t	31521.259

Table 3.2: The fitted stiffness parameters used in the Camarillo et al. model. k_t was derived using Young's modulus and cable area, whereas all other coefficients were estimated from data

be estimated given Young's Modulus G and the cross-sectional area of the cable A. However, there is no way to derive k_a and k_b analytically. Therefore, they must be obtained from data.

To do this, all of the cables on the robot were loosened. Then, the robot was programmed to step through a range of 24 different cable lengths for one of the cables. The cable tightened a step, and then five position measurements were made. This was repeated for each cable. A loss function was then developed, which could be used to determine the correct parameter values. The loss function is defined as follows, where y is the measured tip position, f is the function relating the change in cable length $\Delta\ell$, the axial stiffness k_a, the bending stiffness k_b and the cable angle θ to the position of the tip based on the Camarillo model.

$$L(y, \Delta\ell, k_a, k_b, \theta) = (y - f(\Delta\ell, k_a, k_b, \theta))^T (y - f(\Delta\ell, k_a, k_b, \theta)) \tag{3.48}$$

The additional angular parameter was added to account for the cables not being perfectly aligned with where they are in the CAD model ($\theta = -180°, -90°, 0°, 90°$). The loss function was then optimized for each cable on the robot, and the resulting stiffness coefficients were averaged to produce the values in Table 3.2. The fitted data for the first cable projected onto the bending plane can be seen in Figure 3.6

Thus, parameters for the Camarillo Model have been determined, which can be used in future modeling.

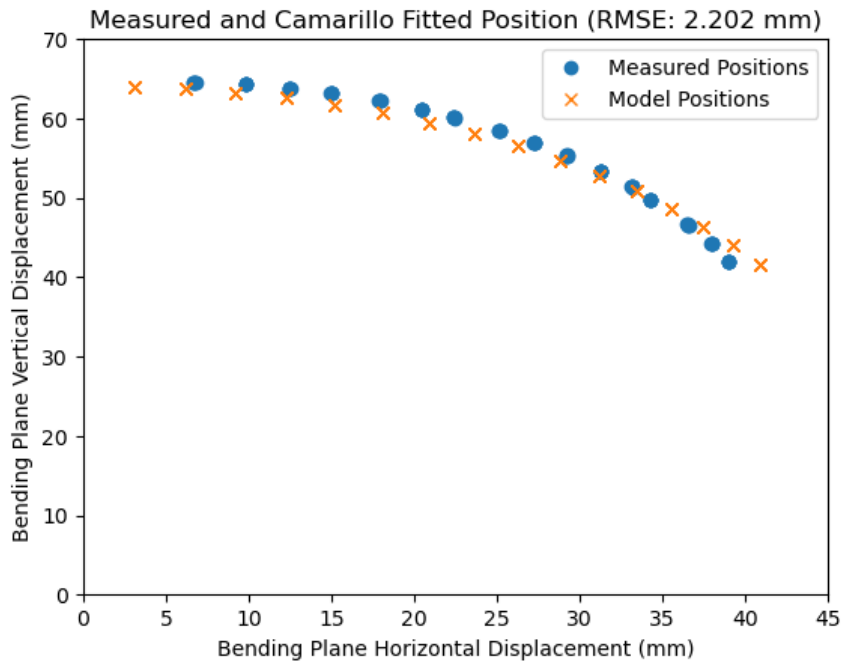


Figure 3.6: Measured points and Camarillo fitted points projected onto the bending plane. Although the Camarillo model matches the rough shape, there is a deviation from the measured behavior.

3.2.2 Learned Modeling

The constant curvature models described above are useful because they are easy to understand and provide an intuition about the robot's behavior. By looking at the D matrices in the Camarillo model, one can envision how each cable affects both the bending and axial strain of the robot. For these models to be tractable, however, they require simplifying assumptions to constrain the problem space to a solvable region. That assumption is announced in its name for constant curvature models: it assumes that each of the segments bends in constant curvature, circular arcs due to a constant moment applied by the tendons along the spine's length. While these models may account for compressive forces in the cables, ultimately, they are constrained by their initial assumption and cannot account for more complicated factors, such as variable cable geometry, anisotropic material properties, and cable friction. This can be seen

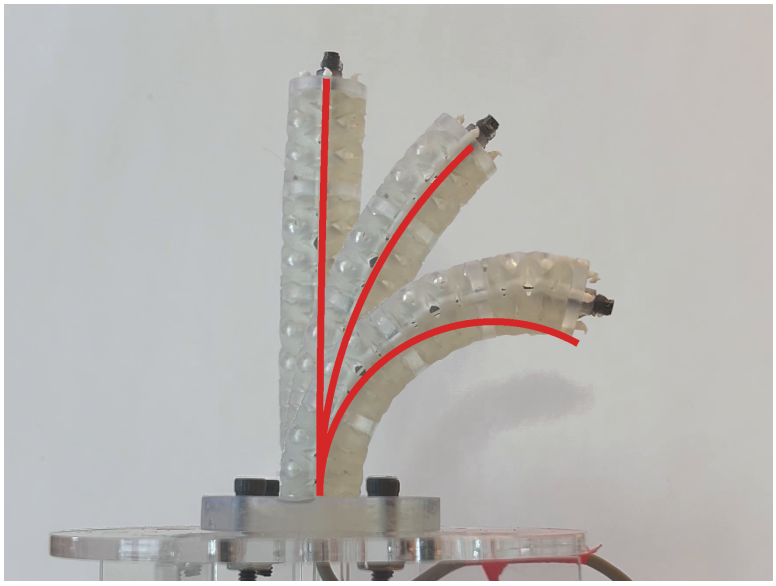


Figure 3.7: As a spine undergoes higher cable displacements, constant curvature models deviate from true behavior due to factors such as cable friction.

in Figure 3.7.

While there are models that forgo the constant curvature assumption, most notably models based on the Cosserat Rod formulation [61] [62] [70] and FEM based models [22]. These models trade the simplicity and intelligibility of constant curvature models for higher accuracy, resulting in collections of highly coupled nonlinear equations in the case of cosserate rod models, or high dimension matrix equations in the case of FEM models, which must be solved numerically. While this is not to say that these approaches are unworkable, they lose much of the benefit of simpler analytical models while still not matching all of the idiosyncrasies of the real-world system. Therefore, researchers have attempted to solve the problem from another angle, namely, learning a model purely from data.

This learned modeling approach is attractive because, in theory, it will produce a model that is better able to capture a system's complex dynamics and properties. Continuum manipulators, in particular, are good candidates for these types of models because they often have much more complicated behavior than standard robots.

Introduction to Artificial Neural Networks (ANNs)

A model learned from data is generally built on top of some non-linear function approximation scheme. Classic examples include Fourier Series [71], and polynomial basis functions [26]. In recent decades, the use of ANNs for this task has exploded. ANNs are a collection of "artificial neurons" which are roughly modeled off the neurons in the human brain, and can model any function with arbitrary precision given that the network used is large enough [24]. Each neuron is a function relating a series of inputs $u = [u_1 u_2 \dots u_n]$ and an output y. In the brain, connections between neurons can be strong or weak, weighing certain inputs to the neuron more than others. Similarly, the artificial neuron stores with it a collection of weights $w = [w_1 w_2 \dots w_m]$ that are used to favor inputs over others. This is achieved through a weighted sum or dot product of the inputs and the weights. Some neurons in the brain are also generally more active than others. Therefore, the artificial neuron also has a bias b, which is added to the weighted sum. Finally, all of this is passed through a nonlinear activation function f, which introduces nonlinearity into the models. Some common activation functions are sigmoids, hyperbolic tangents, and Rectified Linear Units (ReLUs) [72]. All of this is combined into a single function for each neuron.

$$y(u, w, b) = f(w \cdot u + b) \tag{3.49}$$

Alone, an individual neuron cannot do much. However, by combining neurons into larger networks of neurons and combining their outputs, more complex behavior can be modeled. These networks can be created by stacking neurons in parallel into a layer or by using the output of previous neurons as inputs to subsequent neurons. By adjusting the weights and biases of all of the neurons in a model, one can adjust the output of the function to match any desired behavior. However, it is not obvious from the outset how to modify the weights and biases to improve the model's performance.

The first step to figuring out how to improve the model is to characterize how well the model is performing. This is achieved using a scalar loss function, which compares the output of the function $y \in \mathbb{R}^m$ to the desired output $y^* \in \mathbb{R}^m$ for each input in the dataset. A common form of loss function, $L(y)$, is the Mean Square Error (MSE) between the output and the truth, where m is the number of output dimensions.

$$L(y) = (y - y^*)^T (y - y^*)/m \tag{3.50}$$

As y approaches y^*, the loss function approaches 0. Therefore, by tweaking the model's parameters, namely the weights and biases, one can find the optimal model by minimizing the loss function. This is achieved using a technique known as stochastic gradient descent [73]. Let $g : u \to \mathbb{R}$ be an arbitrary multivariate scalar function where $u \in \mathbb{R}^n$. The gradient of g, ∇g, is calculated by taking the partial derivative of g with respect to each input variable u_1, u_2, ..., u_n.

$$\nabla g = \begin{bmatrix} \frac{\partial g}{\partial u_1} & \frac{\partial g}{\partial u_2} & \dots & \frac{\partial g}{\partial u_n} \end{bmatrix} \tag{3.51}$$

Using the gradient of a function, one can "step" down the function, adjusting the parameters a small amount in the direction of minimal loss. This stepping can be repeated until reaching a local minimum. It is analogous to walking down a hill. By finding the steepest direction on the hill and walking down it, one can arrive at the bottom. In the case of the loss function, the goal is to change the weights by some Δw that decreases the current loss. Let δ be a scaling factor that controls the size of the steps taken

$$\Delta w = -\frac{\partial L}{\partial w}\delta = -\nabla L \delta \tag{3.52}$$

The gradient of the function can be found in many ways. The simplest way is to estimate it via finite difference, which is analogous to taking a very small step in each

direction and seeing whether the value of the function increases or decreases. This then allows one to figure out which direction to move in. While this method is simple, it is only approximate and is expensive, requiring multiple calculations to estimate the gradient. Instead, a technique known as backpropagation is generally used [73].

Backpropagation is an exact method for calculating the gradient based on the chain rule. The chain rule is a technique used to calculate the derivative of functions of functions. Let $F = f_1 \circ f_2 \circ \cdots \circ f_\ell = f_1(f_2(\ldots(f_\ell(x))))$ be an arbitrary function which is a composition of the functions $f_1, f_2, \ldots, f_\ell$. The derivative of F with respect to x is calculated as follows

$$\frac{dF}{dx} = \frac{df_1}{df_2}\frac{df_2}{dx} = \frac{df_1}{df_2}\frac{df_2}{df_3}\cdots\frac{df_\ell}{dx} \tag{3.53}$$

The loss function has the exact same structure as F: it is a series of compositions of functions. It is composed of the output of the neural network y, which is composed of each neuron's parameters and the output of previous neurons in the network. Therefore, by starting at the loss function and calculating its derivative with respect to its input, one can propagate its derivative to the output layer of the neural network. This process is then repeated for each layer in the network, propagating derivatives backward to calculate the gradient of the loss function with respect to all of the parameters of the network.

With the structure of neural networks outlined, what follows is a discussion of how they are used in the context of continuum robots.

Forward Kinematics Model

As discussed above, there exists some nonlinear function $f(\Delta\ell)$ which relates changes in cable lengths to the tip pose of the robot. This function is not known beforehand. Therefore, a model of the system $\hat{f}(\Delta\ell)$ must be developed, which approximates the kinematics of the real system. This is achieved using a neural

network with an input dimension equal to the number of cables in the robot, two hidden layers of 32 neurons each, and uses ReLU activation functions on the input and hidden layers. The size and number of hidden layers were based on similar networks seen in literature [25]. Similarly, ReLUs were used as they have been found to perform better than other activation functions, such as sigmoidal functions or hyperbolic tangents, while still retaining a very simple gradient for backpropagation [72].

The output of the network is comprised of two components. The first is a vector in $\mathbb{R}^3$ representing the position of the tip of the robot in model space, and the second is another vector in $\mathbb{R}^3$, which is a representation of the orientation of the tip of the robot relative to the model frame.

There are many different ways to represent orientations, such as Direction Cosine Matrices (DCMs), quaternions, and Euler angles [74]. DCMs are one of the most common representations of orientations and are a member of the SO(3) Lie group. They are orthogonal 3×3 matrices with a determinant of 1 (meaning that they do not scale any vectors that they are multiplied with) and an inverse that equals the transpose of the matrix. Additionally, they can be chained into compound rotations through matrix multiplication, making them easy to work with. This ease of use, however, comes at a cost as they require 9 terms to represent a 3 DOF orientation, which means that there are 6 constraints on the values of the terms of the matrix. These constraints prevent singularities but make DCMs prone to stop functioning correctly in the presence of noise or non-idealities. Because the goal is to approximate the orientation of the robot's tip using a neural network, the values output by the network will not be exact, meaning that the DCMs it would form would not represent valid orientations and, therefore, were not used.

Quaternions are another parameter set used to represent orientations. They are made up of 4 terms, which are normalized to unit length. For a given rotation θ about an axis $e = e_1\hat{i} + e_2\hat{j} + e_3\hat{k}$, the equivalent quaternion is constructed as follows

$$< q_0, q_1, q_2, q_3 >= \left\langle \cos\left(\frac{\theta}{2}\right), e_1 \sin\left(\frac{\theta}{2}\right), e_2 \sin\left(\frac{\theta}{2}\right), e_3 \sin\left(\frac{\theta}{2}\right) \right\rangle \tag{3.54}$$

Quaternions contain no singularities and have relatively simple methods for creating compound rotations, so they are generally popular, especially in robotics. However, they were ultimately not selected because they have the constraint that only a unit quaternion represents a valid rotation and would, therefore, be susceptible to noise and approximation errors.

To avoid constraints on the neural network's output, a parameter set that uses 3 values to represent orientations is needed. The most widely known parameter set of this type is Euler angles. Euler angles are a series of three successive rotations about the body axes of an object. While there are many possible sets of Euler angles, the most common are 3-2-1 Euler angles, colloquially known as Yaw-Pitch-Roll Euler angles. They represent a rotation ψ about the third or z-axis, followed by a rotation of θ about the second axis and a rotation of ϕ about the first axis. This set of Euler angles is commonly used in aircraft and provides an intuitive understanding of constructing the rotation that other parameter sets do not. However, this intelligibility comes at a cost that similar orientations can be represented by vastly different values of ψ, θ, and ϕ. This makes them susceptible to noise and approximation errors like the above parameter sets, so they were not used.

Returning to DCMs, they are members of the Lie group SO(3). Each Lie group also has what is known as a Lie algebra, which can be thought of as a tangent space to the group about the identity element of the group. The basis functions of these Lie algebras, known as generators, come from the derivatives of the group around the identity in specific directions. In the case of SO(3), the Lie algebra is so(3), and the generators come from the differentiation of the rotation matrices about each axis [75]

evaluated at identity ($\theta = 0$). These rotation matrices are collected in Equation 3.55, and the generators are collected in Equation 3.56.

$$R_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} R_2 = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} R_3 = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.55}$$

$$G_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} G_2 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix} G_3 = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{3.56}$$

To relate the coordinates $(\omega_1, \omega_2, \omega_3) \in \mathrm{so}(3)$ to its corresponding DCM $R \in \mathrm{SO}(3)$, the following skew-symmetric matrix is formed, which is a linear combination of the basis elements

$$\tilde{\omega} = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \tag{3.57}$$

This skew-symmetric matrix is related to R by an exponential map, which is constructed using the Taylor series expansion for e^x.

$$R = e^{\tilde{\omega}} = I + \tilde{\omega} + \frac{\tilde{\omega}^2}{2!} + \frac{\tilde{\omega}^3}{3!} + \dots \tag{3.58}$$

To invert this relationship and extract the tangent space coordinates for a given rotation matrix, a matrix logarithm is used to get the skew-symmetric matrix $\tilde{\omega}$, and the coordinates are extracted from it.

$$\tilde{\omega} = \ln R \tag{3.59}$$

Using Equations 3.58 and 3.59, one can go between the rotation matrices, which are part of the transformation matrices used to represent frame poses, and the tangent space representation. Additionally, this parameter set performs well in neural networks and other approximations because it is unconstrained, and small changes in any one coordinate correspond to small changes in the final rotation matrix. While this coordinate set is singular around a rotation of 180° which implies that $|\langle \omega_1, \omega_2, \omega_3 \rangle| < \pi$, ultimately this is easy to avoid, leading to this Lie algebra representation of an orientation being used as part of the output of the neural network, seen in Equation 3.60

$$y = \begin{bmatrix} t_x \\ t_y \\ t_z \\ \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} \tag{3.60}$$

The loss function for the model is a weighted mean square error loss function. Given a model output $\hat{y}$, a target output y, and a weighting matrix W, the loss is calculated in Equation 3.61.

$$L(\hat{y}, y, W) = (y - \hat{y})^T W (y - \hat{y})/6 \tag{3.61}$$

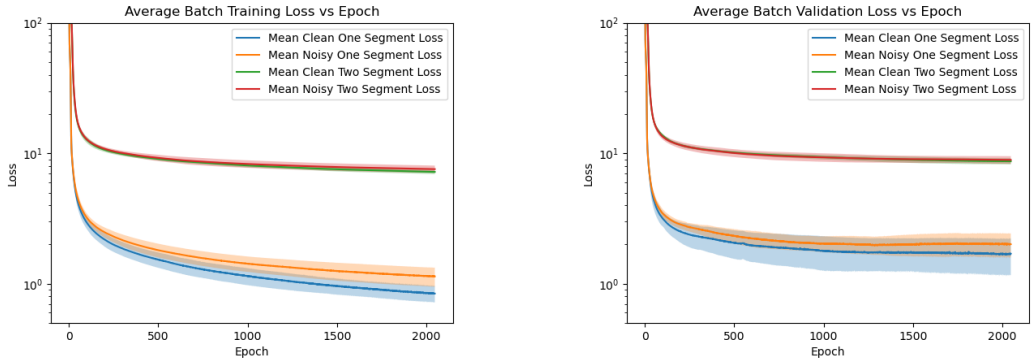
The weighting matrix is a diagonal matrix introduced to scale the relative contribution of terms to the loss function. Because the position is roughly constrained to the interval $(-32, 32) \times (-32, 32) \times (0, 64)$ and the orientation is constrained to the

interval $(-\pi, \pi)^3$, each orientation is scaled by a factor of 10. This can be adjusted by shrinking or scaling this factor to favor positional or orientation accuracy over the other. W is shown below.

$$W = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 0 & 0 & 10 \end{bmatrix} \tag{3.62}$$

With the loss function and model architecture established, training data-based models could commence. Before attempting this training on the real system, an experiment was devised to validate the efficacy of the model architecture and loss function. To do this, Camarillo style constant curvature models without slack compensation of a one and two-segment version of the continuum robot were used, including the fitted stiffness coefficients found in Section 3.2.1. Using random cable lengths, the position and orientation of the robot tip are calculated for $2^{14} = 16,384$ points.

There are two options to generate the random cable lengths: a uniform distribution or a normal distribution. A uniform distribution provides a more even coverage of the workspace of the robot than the normal distribution. However, the normal distribution produces more points with smaller cable displacements than the uniform distribution, meaning the model will be more accurate in this small displacement regime. The distribution used to generate a training dataset depends on the desired application. Ultimately, however, it was decided that better coverage of the entire workspace would be more useful in later control applications and produce a more accurate model overall. Therefore, a uniform distribution of cable lengths over the interval (−12mm, 12mm) was used going forward. For every timestep, each cable selected a random value from



(a) (b)

Figure 3.8: Simulation training and validation loss for all four scenarios. The shaded region represents one standard deviation above and below the mean of the ten models.

this distribution; the cable displacements were then put through the Camarillo model, and the tip pose was measured.

After generating a dataset for both the one-segment robot and the two-segment robot, a copy of each dataset was made, and noise was added to the measurements. The noise was generated using a zero mean normal distribution with a standard deviation of 0.5mm for the position and 0.05 radians for the orientation. This was used to more accurately simulate real measurements gathered from the system.

For each dataset, ten different models were trained, each using the ADAM optimizer [76] with a learning rate of 10^{-3}. All models were developed in PyTorch. Due to the small size of the models, loading onto the GPU and training on it was slower than the CPU, therefore all training was done using the CPU. The models used 75% of the data as training data and the remaining 25% as validation data. Training data is data used to update the model's parameters, whereas validation data is used to monitor the training of the model and prevent overfitting. The partition was created randomly for each model. Each model was trained for 2048 epochs, and the epoch with the lowest validation loss was selected. The average training and validation loss across the forty models can be seen in Figure 3.8.

After training the models, all were tested on a separate test set comprised of

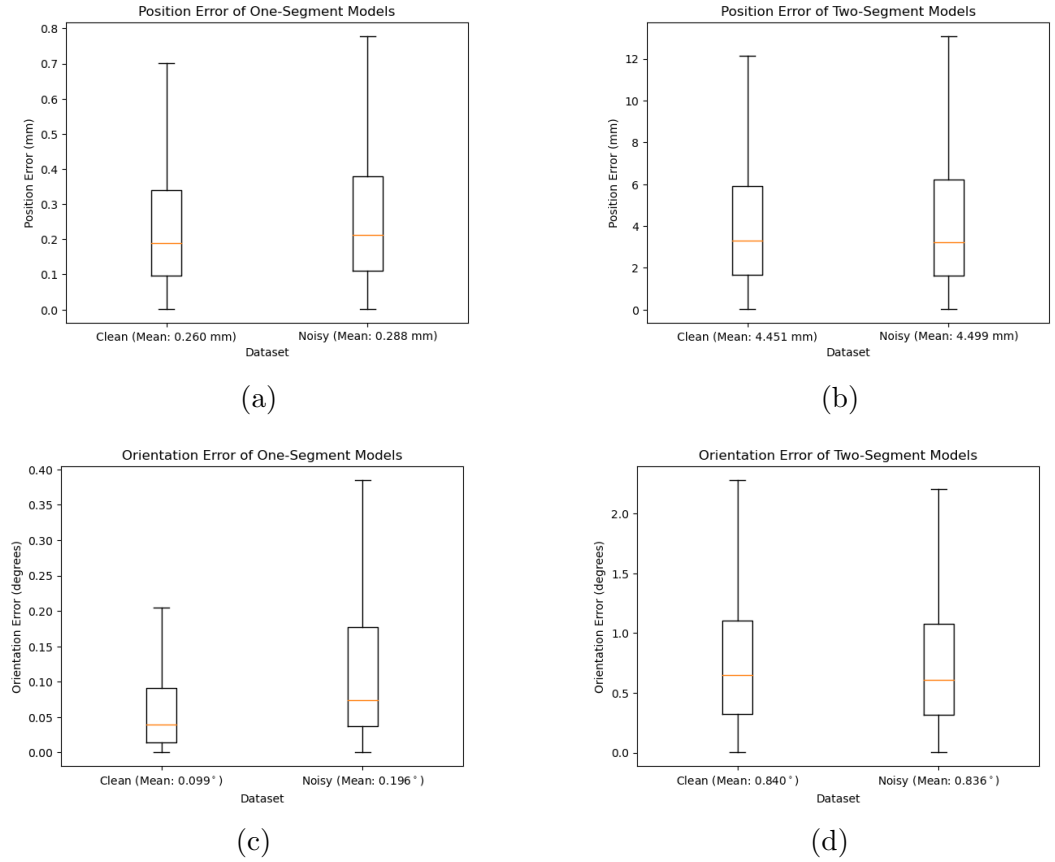


Figure 3.9: Position and orientation errors comparing the clean and noisy models for each model.

512 random cable length measurements. These were not corrupted with noise. The position and orientation errors were then calculated. These can be seen in Figure 3.9. The one-segment models are generally more accurate than their two-segment counterparts. This makes sense as the input space of the one-segment models is four-dimensional, whereas it is eight-dimensional for the two-segment models. This increase in dimensionality means that the training data covers less of the available workspace for the robot. It also means that a larger model may be required. Therefore, it is expected that these models will perform worse. An investigation into improving this performance is covered in Appendix D.

On the other hand, noise does not seem to greatly impact the model's accuracy. This is a good sign, as the models are trained on real data, which invariably contains

noise.

Training on the real robot follows a similar procedure to the above. A dataset of $2^{14} = 16,384$ random cable length data points is collected. To ensure that there are no dynamic effects on the measured tip pose, each data point is collected four seconds after the motors are commanded to move. Originally, each measurement was collected two seconds after being commanded, which led to poor performance of learned models. This is because, as is shown in Figure 3.10, the spine requires time to settle fully. More specifically, the movement of the servos and the cables requires around 0.25 to 0.5 seconds, which is characterized by the steep rate of change in the initial part of the transient as they are pulling on the spine. However, when the cables are loosened over the trajectory, a point is reached where the cables are no longer pulling on the spine, and the elasticity of the spine dominates the transient response. This elasticity is much slower than the movement of the cables, taking around 3.5 seconds to settle fully.

Data collection is performed in one roughly 17-hour session. After collecting the data, they are cleaned to remove any invalid measurements. This is done by looking for all NaNs in the dataset and all points more than 75 mm away from the origin or with an invalid orientation (any component greater than π), as these are impossible values to reach and indicate corruption of the measurements. For the dataset collected in this experiment, this meant that eight points or $\sim 0.05\%$ of the measurements were removed from consideration. The measured points can be seen in Figure 3.11, as well as the corresponding points generated in simulation using the constant curvature model.

As can be seen, the position distribution in the real spine is not uniform despite a uniform distribution of cable displacements. This non-uniform distribution and its causes merit further discussion before continuing with training. The main cause of this behavior is the effect of slack cables. Pulling on a cable compresses the length

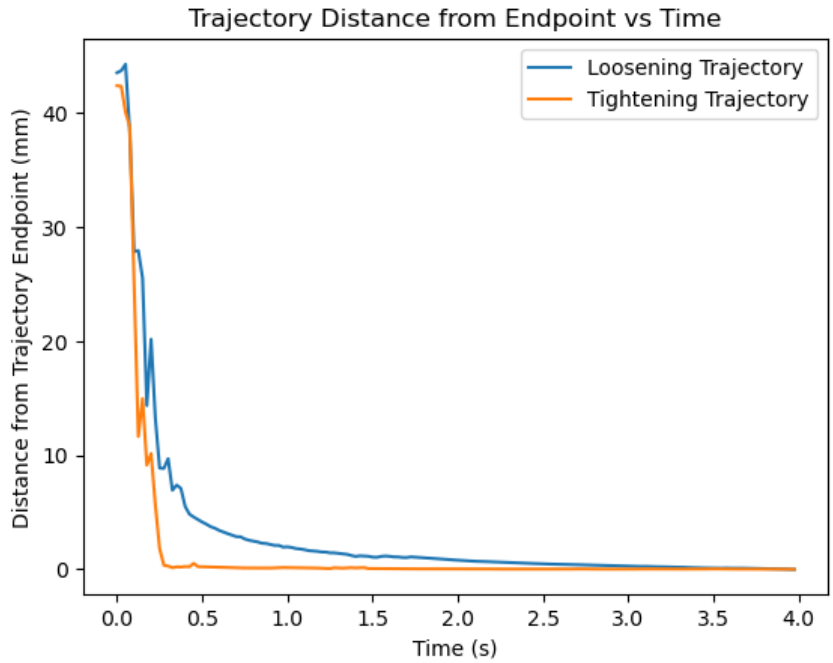
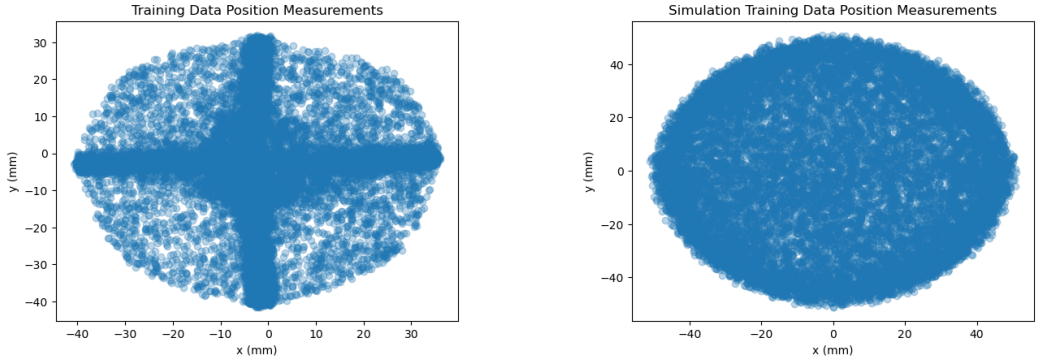


Figure 3.10: As can be seen, depending on the direction in which the spine moves, there can be a significant decay period before the spine tip reaches its final position. Specifically, on trajectories where the overall tension on the spine is loosened, the spine is responsible for springing into shape, which is a lot slower than the movement of the servos seen in the steep initial change in both trajectories.



(a) (b)

Figure 3.11: As can be seen, the distribution of points produced by the real spine (a) and the simulation (b) differ, with the real spine producing a non-uniform distribution of points collected about the cable axes.

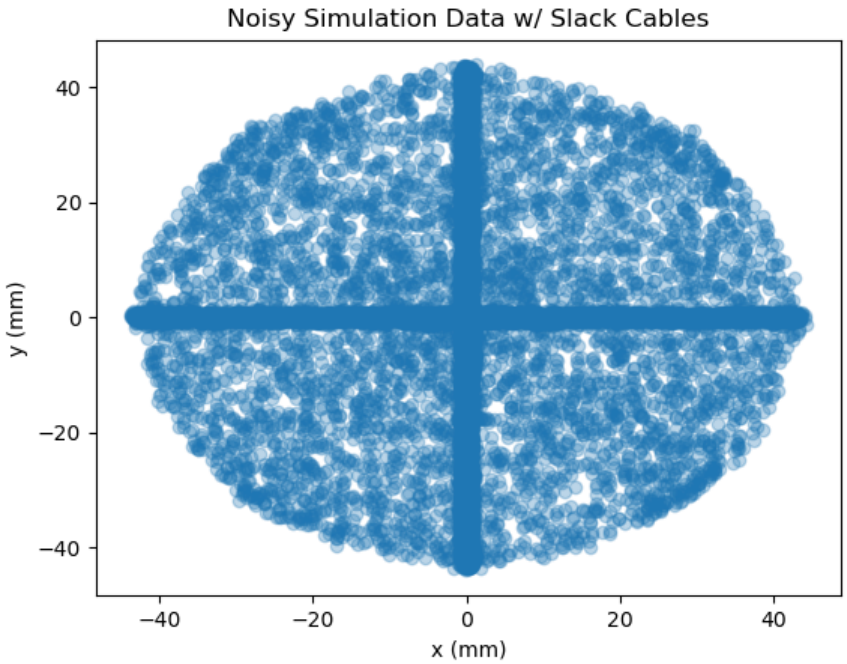


Figure 3.12: Distribution of points using the Camarillo constant curvature model with slack compensation. As can be seen, there is a clumping of points around the cable axes, much like what is seen in the measured points.

of the spine in addition to bending it. This compression of the spine means that a different cable attempting to pull the spine sideways must take up all of the slack caused by compression before it actually bends the spine sideways. Therefore, it is reasonable to expect a grouping of the cables around the cable axes, as shortening one cable requires a shortening of all of the other cables to reduce slack. To show this effect, using the Camarillo model with slack compensation applied, the simulated distribution of points matches more closely what is found in real life, with a clumping around the cable axes. This can be seen in Figure 3.12.

While other factors, such as spine geometry and hysteresis, likely contribute to this behavior of clumping around cable axes, it is largely caused by slack cables. Therefore, this highlights one of the potential inaccuracies of constant curvature models: slack tendons are not always considered.

Returning to model training, ten models were trained on the real-life motor-babble

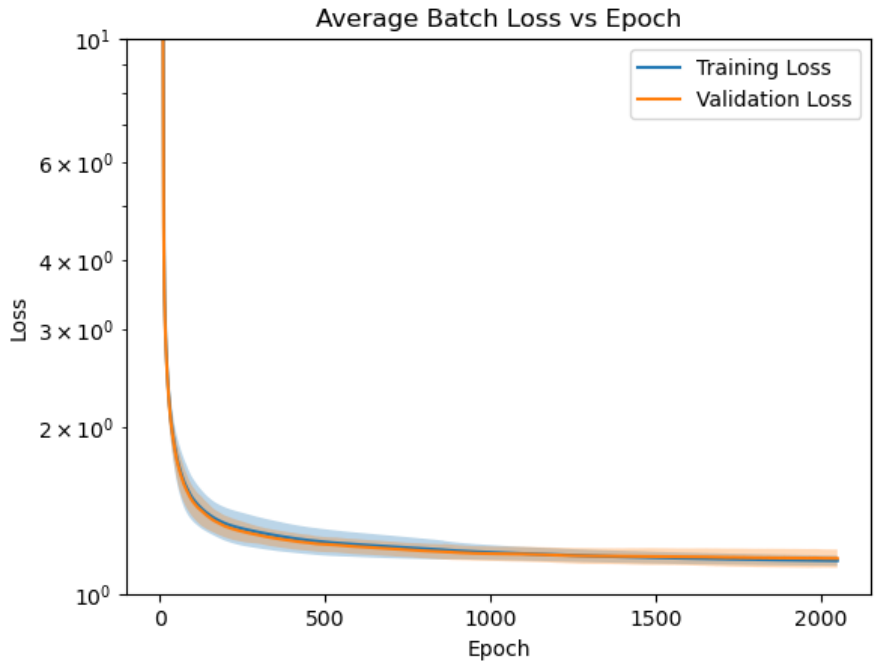


Figure 3.13: As can be seen, the training and validation errors decrease quickly before remaining roughly constant for most of training. The losses seen for the real models are on the same order as the models learned from simulated data.

dataset generated above with the same loss function and used a 75%-25% split between training data and validation data. The models were trained over 2048 epochs, and the epoch with the lowest validation error was selected for each model. The training and validation loss by epoch are shown below, and they highlight that the models consistently converged to a parameter set that produced a low loss.

To test the models, a dataset was generated using cable displacements gotten from the pure kinematics model discussed in Section 3.2.1. Using the angular constant curvature parameters (ℓ, θ, ϕ), the cable displacements needed for an angular mesh with $\ell = 64$mm, $\theta = [\frac{\pi}{24}, \frac{\pi}{12}, \frac{3\pi}{24}, \ldots, \frac{\pi}{2}]$, and $\phi = [\frac{[}{0}, \frac{\pi}{12}, \frac{\pi}{6}, \ldots 23\pi 12]$ were determined. Commanding these cable displacements, five measurements were captured for each point in the mesh. The learned models were then compared to the true position of the spine. Figure 3.14 shows the error for the ten learned models.

The model with the lowest test error (model 9) was selected. Figure 3.15 shows

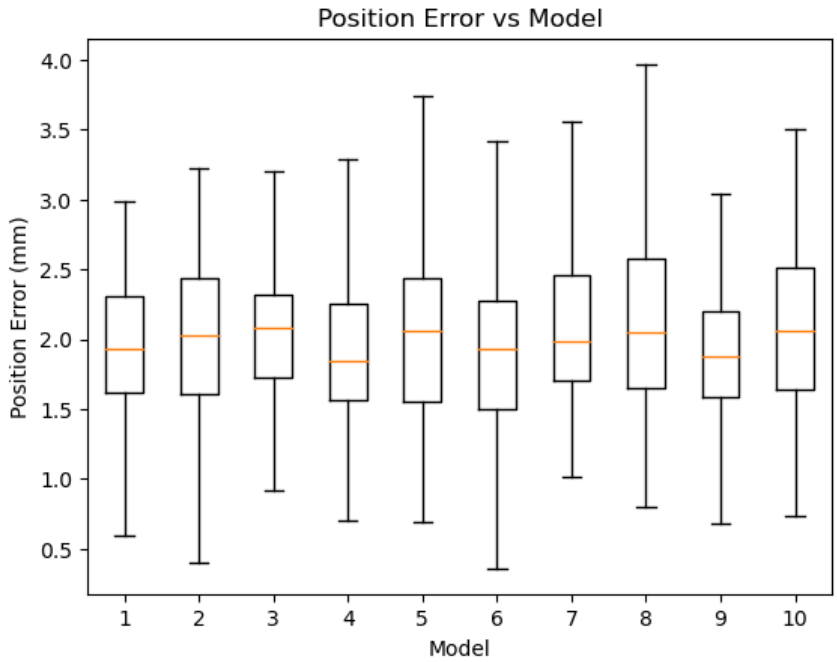


Figure 3.14: Position error of the ten different models trained. Although there is variation between them, they are all roughly similar.

the histogram of its position error.

With a learned kinematic model, it can now be compared to the constant curvature models outlined earlier.

3.2.3 Model Comparison

To compare the constant curvature models with the learned model, the same test that was used for the learned kinematic model was repeated for all three models—the pure kinematics model from Section 3.2.1, the Camarillo model from Section 3.2.1, and the learned model with the lowest test loss. After moving the robot and performing a new registration, the robot was moved through an angular test mesh of 12×24 points, collecting five measurements at each point. The robot was re-registered to better simulate how the model will perform, as the model will need to be used in various locations, requiring a new registration. The positions they output were then

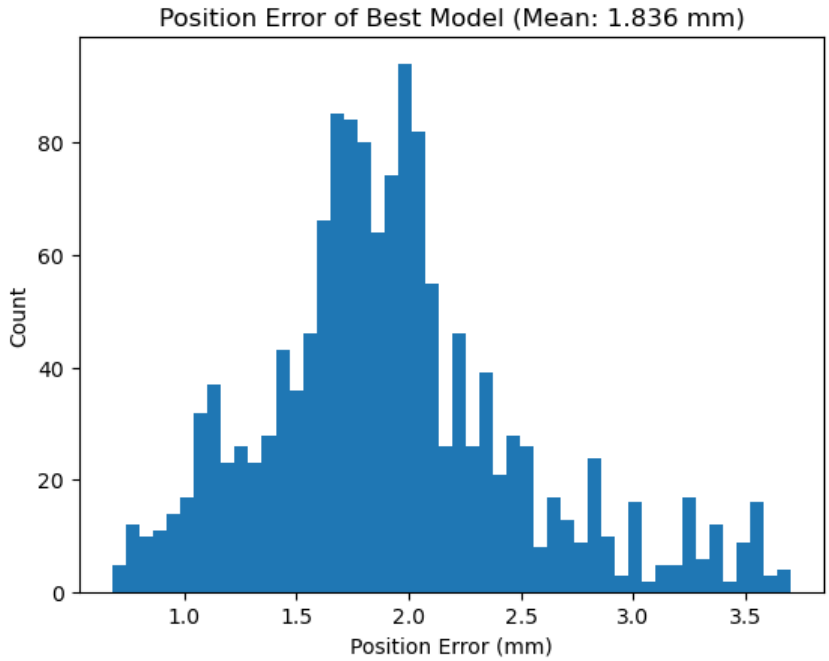


Figure 3.15: Position error histogram of learned kinematic model

compared to the measured positions.

Of all of the models, the learned model performed the best, with roughly equal performance out of the two constant curvature models. A projection of the positions onto the x-y plane, along with histograms of the position errors of the three models, are shown in Figure 3.16.

This result highlights the power of models learned directly from data, as the error in the constant curvature models is roughly four times greater than that of the learned model. This model can better handle the myriad complexities inherent in any real-world system. Additionally, the model is not dependent on the initial registration and can be re-registered in different locations.

3.2.4 Model Sources of Error

Although the learned kinematic model performs significantly better than the constant curvature models, sub-millimeter accuracy is generally considered the threshold

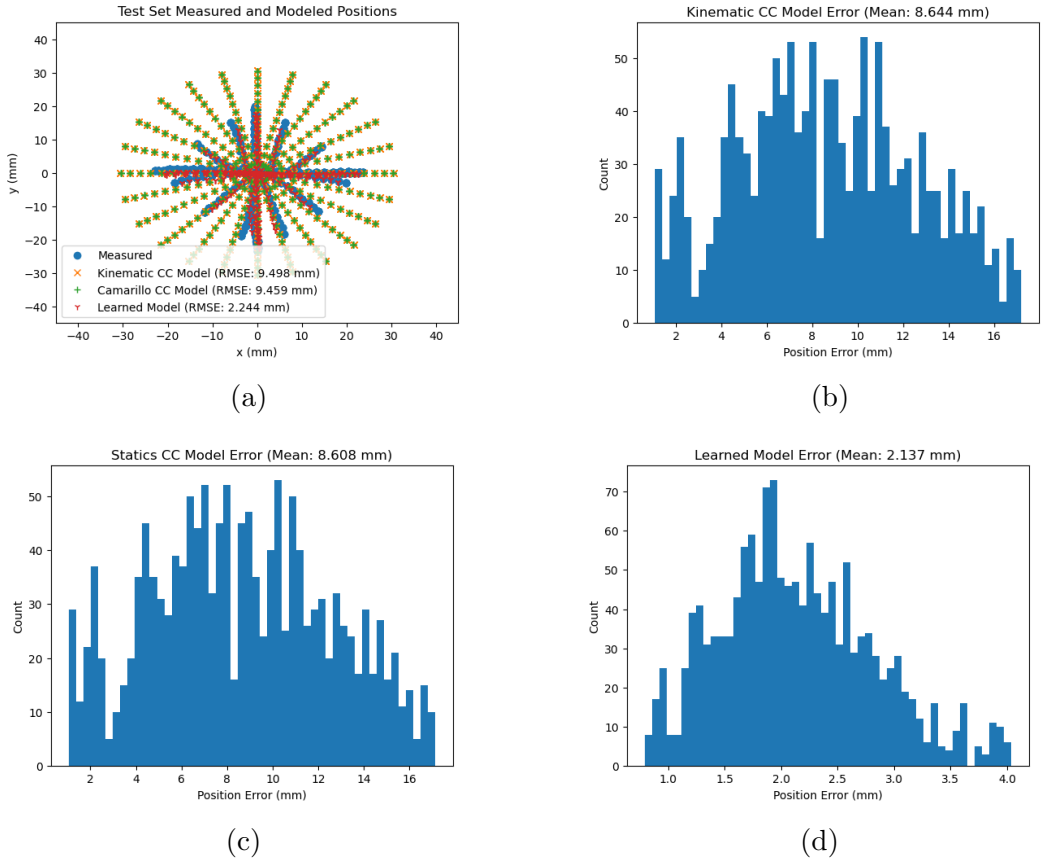


Figure 3.16: Measurements collected in model comparison test. (a) Projection of measured and model points onto x-y plane. Both constant curvature models produce roughly the same position. (b) Pure kinematics error histogram. (c) Camarillo statics-based error histogram. (d) Learned model error histogram, with by far the lowest average error.

required for high-precision surgical applications [77]. Additionally, this robot is much smaller than most surgical robot systems, and in its current one-segment configuration, it does not experience much external loading. Therefore, understanding where the error comes from is crucial to reaching this accuracy standard.

When performing robot registration, the aurora frame is registered to the model frame using twelve points repeated five times for a total of 60 points. Similarly, the aurora frame is registered to the tip frame using twelve points repeated five times. In performing many robot registrations, the Root Mean Square (RMS) Fiducial Registration Error (FRE) of both frames is roughly 0.4 mm. This registration error can then be used to estimate the Fiducial Localization Error (FLE) using Equation 3.63 [78].

$$\langle FRE^2 \rangle = \left(1 - \frac{2}{N}\right) \langle FLE^2 \rangle \tag{3.63}$$

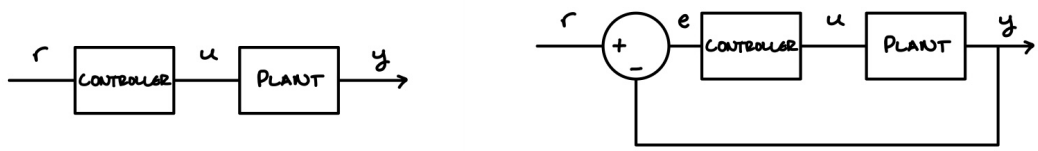
FLE measures how accurately points can be located in a given frame using a registration. Because N is large for both frames

$$\langle FRE^2 \rangle \approx \langle FLE^2 \rangle \tag{3.64}$$

Because the tip position in the model frame is calculated using both transformations, the FLE of the tip frame can be found as follows, assuming that FLE is normally distributed.

$$\langle FLE^2_{T2M} \rangle = \langle FLE^2_{A2M} \rangle + \langle FLE^2_{A2T} \rangle \tag{3.65}$$

Using this, one finds that the FLE of the tip-to-model frame transformation is approximately 0.57 mm. Because two registrations are being used, this error is on par with what is typically expected [77]. Therefore, the remaining error in the model is from the model and the robot itself.



(a) (b)

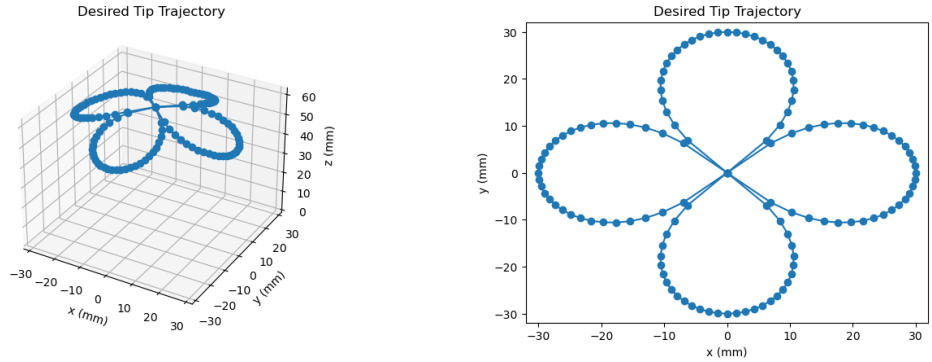
Figure 3.17: Different control architectures. (a) Open-loop control does not contain feedback, with information only traveling forwards in the system. (b) Closed-loop control introduces feedback and bases its control effort on the error signal.

3.3 Control

By developing a model of the system, controllers can be developed for it. There are two main types of controllers: open-loop and closed-loop. An open-loop controller relies solely on the model to determine the input to apply to the system. It is analogous to crossing the room by estimating how far away the other side is and then closing your eyes and walking to the other side of the room. While an open-loop controller is simple to implement, it requires precise knowledge of the system and its surroundings. Any inaccuracies in the model are present in the commanded action.

Closed-loop control, on the other hand, introduces feedback into the actions taken. Instead of closing one's eyes and walking across the room like with an open-loop controller, a closed-loop controller looks at how close it is to the other side of the room to determine how much further it should go. While closed-loop control leads to higher accuracy, it also requires that you measure information about the system's current state. In the case of our robot, this is done by the Aurora. However, measurements may not always be available, therefore one cannot always rely on closed-loop control. Block diagrams of open and closed-loop controllers can be seen in Figure 3.17.

Using the learned models developed both on simulated continuum robots and



(a) (b)

Figure 3.18: Desired trajectory for robot based off of lemniscate curve. (a) 3d view of trajectory showing its projection onto a constant curvature surface. (b) Overhead view of trajectory to better show lemniscate curve shape.

the real system, open-loop and closed-loop controllers are developed in the following sections. Their performance is tested by tracking a reference trajectory through the robot's workspace, shown in Figure 3.18.

The trajectory is composed of two lemniscates on the x-y plane defined by the following equation

$$r(\theta) = 30\sqrt{|\cos 2\theta|} \tag{3.66}$$

These x-y positions are then projected onto the surface defined by the pure kinematics constant curvature model of Section 3.2.1 with a spine length of $\ell = 60$ mm. This length was chosen to ensure that all of the points could be reached despite the compression that the spine experiences due to tension in the cables. With a trajectory established, controller development could proceed.

3.3.1 Open Loop Control

As stated above, open-loop control uses the system model to determine the best control input, which can then be applied to the system. This problem of determining

the correct inputs based on a desired position is also known as inverse kinematics. As the name suggests, inverse kinematics is the opposite of forward kinematics, which maps from control inputs to the system output, which, in our case, is the robot's tip pose.

To develop an optimal controller, a loss function is needed, which can then be optimized. The system's loss function takes the following form, where y^* is the desired tip position, u_0 is the starting cable displacements, $\hat{f(u)}$ is our learned model which approximates the true kinematics of the system $f(u)$, u is the proposed cable displacement, and Q and R are weighting matrices.

$$L(y^*, u_0, u) = (\hat{f}(u) - y^*)^T Q(\hat{f}(u) - y^*) + (u - u_0)^T R(u - u_0) \tag{3.67}$$

$$= e^T Q e + (\Delta u)^T R \Delta u \tag{3.68}$$

This loss function, which is optimized over u, penalizes position errors and large cable displacements. For the weighting matrices, the following diagonal matrices were used, which weight position errors five times more than cable displacements. In experimentation with various weightings, this was found to produce tip positions close to the reference positions while still penalizing unnecessary large cable displacements. However, one could adjust this value further depending on the requirements of their application.

$$Q = \begin{bmatrix} 5 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 5 \end{bmatrix} \tag{3.69}$$

$$R = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.70}$$

To optimize the loss function, SciPy's implementation of the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm was used [79]. This algorithm takes an initial value, which is set as the trajectory's previous cable displacement u_0. Additionally, this algorithm uses the loss function gradient, $\partial L(y^8, u_0, u)/\partial u$ in the optimization. While this can be approximated using finite differences, instead PyTorch's gradient calculation (AutoGrad) engine was used to calculate the gradient. While this is normally used for calculating the gradients of neural network parameters in training, it can be repurposed to calculate the gradient of the loss function with respect to the function's input. Algorithm 3.1 shows how this gradient is calculated based on the loss function.

```
def loss_fcn(
    u: np.ndarray,
    model: ANN.Model,
    y_star: torch.tensor,
    u_0: torch.tensor,
    weighting: Tuple[float, float],
) -> Tuple[float, np.ndarray]:
```

```
"""
Finds value of the loss function L(y) and its gradient.

Args:
    u: The cable displacements fed into the model
    model: The learned forward kinematics model
    y_star: The desired tip position
    u_0: The cable displacements at the previous timestep
    weighting: Contains the diagonal value of Q and R

Returns:
    The value of the loss function and its gradient
"""

# Convert numpy array to tensor
# (requires_grad is used to extract gradient)
u_tensor = torch.tensor(u, requires_grad=True)

# Evaluate model and extract position
model.zero_grad()
y_hat = model(dl_tensor)[0:3]

# Calculate error and change in cable length
e = y_hat - y_star
delta_u = u_tensor - u_0

# Calculate loss function
```

```
loss = torch.dot(e, e) * weighting[0] +
    torch.dot(delta_u, delta_u) * weighting[1]

# Obtain gradient of loss function based on inputs
loss.backward()
grad = u_tensor.grad.numpy()

# Return loss function and its gradient
return (loss.item(), grad)
```

Algorithm 3.1: Python implementation with PyTorch used to calculate the loss function and its gradient

This controller, which is simple to implement, is quickly able to generate the optimal cable displacements for a given kinematics model based on the loss function defined above. However, without bounds on the inputs to the optimization, the controller can produce faulty values. The performance of neural networks drops precipitously as soon as they go beyond the bounds of their training data. Therefore, any cable displacements outside the interval $(-12mm,\ 12mm)$ run through the model produce garbage values. To prevent this from occurring, a bounded version of the BFGS algorithm, L-BFGS-B, is used [80] [81].

3.3.2 Closed Loop Control

If one possesses a perfect forward kinematics model, then the controller implemented above is sufficient to follow any desired trajectory. However, neural networks are, by definition, function approximators. Therefore, it is expected that some level of error in the model will exist. Additionally, external disturbances and other factors lower the performance of an open-loop controller. For these reasons, closed-loop controllers

are developed that respond to position errors and attempt to reduce them.

The closed-loop controller starts by applying the cable displacements calculated from the open-loop controller. After applying these inputs, the controller uses an estimate of the Jacobian to calculate which direction to step the cables in to reduce the error in the tip position. This process is repeated for ten timesteps, and the robot moves on to the next position in the trajectory.

The Jacobian is an extension of the gradient to vector functions. For a given function $f = f(f_1(x_1, x_2, \ldots, x_n), f_2(x_1, x_2, \ldots, x_n), \ldots, f_m(x_1, x_2, \ldots, x_n))$, the Jacobian of f is calculated as follows

$$\frac{\partial f}{\partial x} = \begin{bmatrix} \nabla f_1(x) \\ \vdots \\ \nabla f_m(x) \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix} \tag{3.71}$$

The Jacobian relates a change in the input variables to the corresponding change in the function's output variables. In a continuum robot system, the Jacobian relates changes in cable lengths to changes in tip position. Using a finite difference approximation, this relationship for the robot becomes

$$\Delta y \approx \frac{\partial f}{\partial u} \Delta u \tag{3.72}$$

This relationship can be inverted using the Moore-Penrose pseudo-inverse [69] to determine the cable-length changes needed to cause a given change in position for small changes. The pseudo-inverse is used as the Jacobian matrix is often rectangular, meaning no true inverse exists.

$$\Delta u \approx \left[\frac{\partial f}{\partial u} \right]^{+} \Delta y \tag{3.73}$$

Given a position error, $e = y - y^*$, the Jacobian is used to calculate the direction

to move the cables to reduce the error by letting $\Delta y = -k_p e$, where k_p is a scaling constant $0 < k_p \leqslant 1$. The scaling constant is introduced to prevent overshooting and oscillations around the desired position.

$$\Delta u = -k \left[\frac{\partial f}{\partial u} \right]^{+} e \tag{3.74}$$

A Jacobian-based controller of this form, despite its simplicity, is guaranteed to achieve zero steady-state error given perfect knowledge of the Jacobian [82]. Therefore, for this controller architecture, one needs an accurate estimate of the Jacobian. There are many different ways to estimate the Jacobian of a continuum robot, including both model-free [83] and model-based methods [84]. In this work, a model-based method is used that leverages the previously developed learned kinematic model and PyTorch's AutoGrad engine to also provide an estimate of the Jacobian. The calculation of the Jacobian is shown in Algorithm 3.2.

```
def calc_jacobian(
u: np.ndarray,
model: ANN.model,
) -> np.ndarray:
    """
    Calculates the model Jacobian at a cable displacement u.

    Args:
        u: The cable displacements input to the model
        model: Neural network model

    Returns:
```

```
    The Jacobian of the model for cable displacement u.
"""
# Convert to tensor
u_tensor = torch.tensor(u.flatten(), requires_grad=True)

# Define unit vectors
unit_vectors = [
    torch.tensor(x)
    for x in [
        np.array([1.0, 0.0, 0.0, 0.0, 0.0, 0.0]),
        np.array([0.0, 1.0, 0.0, 0.0, 0.0, 0.0]),
        np.array([0.0, 0.0, 1.0, 0.0, 0.0, 0.0]),
    ]
]

# Calculate Jacobian
model.model.zero_grad()
jac = torch.stack(
    [
        torch.autograd.grad(
            model(u0_tensor),
            u0_tensor,
            vec,
        )[0]
        for vec in unit_vectors
    ]
).numpy()
```

```
    return jac
```

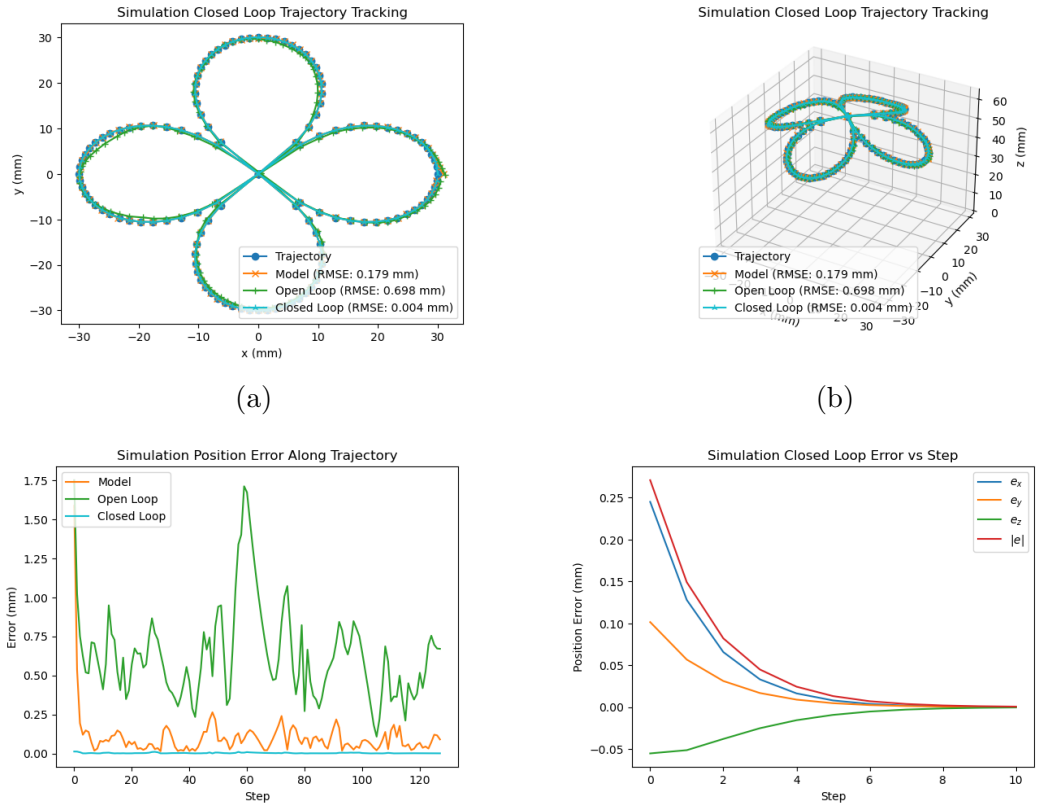
Algorithm 3.2: Python implementation with PyTorch used to calculate the continuum robot's Jacobian

This estimate of the Jacobian is then inverted and passed into Equation 3.74 to calculate the closed-loop adjustment of the spine position. While this method of estimating the Jacobian is easy to implement, it is ultimately dependent on the performance of the model. Therefore, further improvements to the kinematic model should be a primary focus of future development.

Additionally, more complex control architectures than a proportional controller, such as a PID or lead-lag controller, could have been used in conjunction with the Jacobian-based control in our system [82]. However, with a relatively low update frequency of around 4 Hz, any discrete-time controller implemented would need to have very few terms in its difference equation to introduce minimal lag into the system to prevent slowing down an already slow system. With both an open-loop and a closed-loop controller developed, their performance tracking the trajectory outlined above can be analyzed.

3.3.3 Controller Performance

The controllers were first implemented using the models trained on simulated continuum robot data. The best-performing model was plugged into the open-loop controller and used to generate the cable displacements needed for the simulated robot to track the trajectory. After moving to the position dictated by the open-loop controller, the closed-loop controller was run for ten steps at each trajectory point before moving on to the next one. The relative performance of the two controllers on the simulated system can be seen in Figure 3.19.



(a) (b)

(c) (d)

Figure 3.19: Open and closed-loop learned model controllers for the simulated system. (a) x-y plane projection of tracking error, all systems can match reference trajectory. (b) A 3D view of the trajectory tracking to show that height is also tracked. (c) The closed loop system has by far the lowest error of the model and the open loop controller, showing the power of feedback. (d) The position error converges to zero as the closed-loop controller acts

.

Both the open-loop and closed-loop controllers can track the trajectory very accurately, with an RMS error of 0.004 mm. While obviously, this is a simulation, it shows the proficiency of the learned kinematic model applied to control of the system. With this performance, the same experiment was run on the real system.

In implementing the controller on a real-world system, the effects of the system dynamics, namely the transient response, needed to be accounted for. To ensure that movement did not affect the performance of either control policy, a delay of four seconds was introduced between each time step. Lowering delay, however, was found to have relatively little impact on the tip position and control policies, as the current setup of the Aurora can only operate at around four Hz. The performance of the real robot can be seen in Figure 3.20.

As is shown in Figure 3.20, the model is used to find positions that it believes are as close to the trajectory as possible, with an RMS error of 0.948 mm over the trajectory. This error comes mostly from the regions directly around the cable axes, meaning there is uncertainty in the model in these regions, which causes a degradation in performance. While Figure 3.11 shows a clumping of points around the cable axes in the training data, and Figure 3.12 shows that slack cables cause this clustering around the axes, cable slack would not necessarily cause the models to incorrectly model points in this region. Additionally, the real training data distribution has a much wider spread around the cable axes than the simulated data with slack cables. Therefore, there must be another cause for model inaccuracy in these regions.

This other cause is most likely hysteresis. By being biased towards the previous positions of the spine, one would expect to see a larger spread in the points than what is just the result of noise. This is exactly what is seen when comparing Figures 3.11 and 3.12. By adding hysteresis to the Camarillo model with slack compensation, one sees a larger spread of points around the cable axes. To add hysteresis to the Camarillo model, the output of the model $f_c(u(k))$ was biased towards the previous

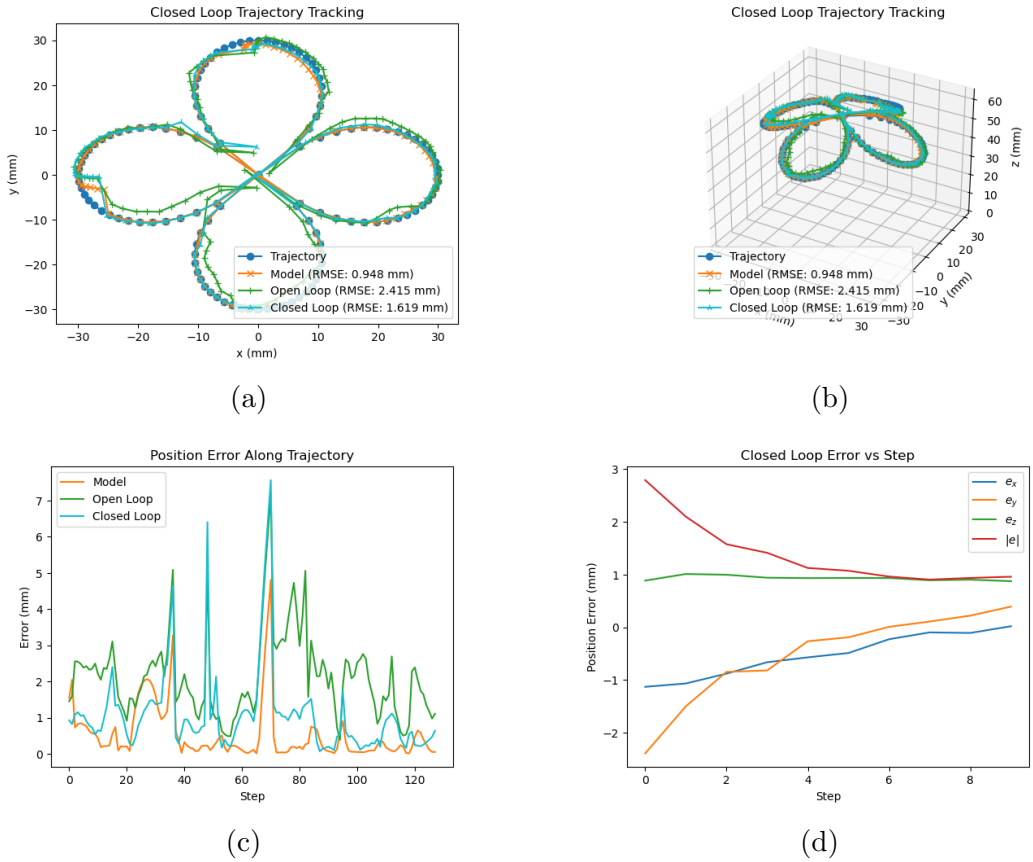


Figure 3.20: Open and closed-loop learned model controllers for the real robot. (a) x-y plane projection of tracking error, all systems roughly match the reference trajectory apart from two short sections of the trajectory where the model loses accuracy. (b) A 3D view of the trajectory tracking to show that height is also tracked. (c) The closed loop system has the lowest tracking error of the group. However, the change is less dramatic than in the simulation. (d) The position error decreases as the closed-loop controller acts

.

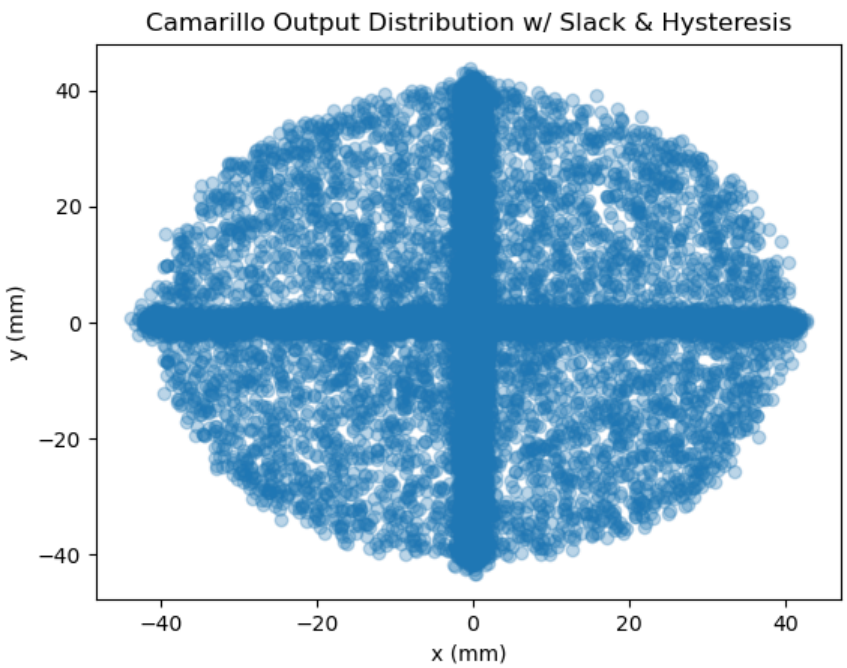


Figure 3.21: Caption

tip position $y(k-1)$ by the coefficient k_h to produce the current tip position $y(k)$ using Equation 3.75.

$$y(k) = (1 - k_h) \cdot f_c(u(k)) + k_h \cdot y(k-1) \tag{3.75}$$

The effects of this hysteresis on the output of the Camarillo model, with $k_h = 0.05$, can be seen in Figure 3.21. Hysteresis acts to spread the clumps out points around the cable axes.

However, the hysteresis in the real system only seems to occur around the cable axes and is much less pronounced in the areas between cable axes. To test this, 17 points in the robot's workspace were chosen: one with no cable displacements, eight on the cable axes, and eight between the cable axes. The robot was then commanded to one of these points after visiting one of the other 16 points. This was repeated for all 16 other points to determine whether visiting that point from different parts of the workspace effected the final position. The tip position clusters from this experiment

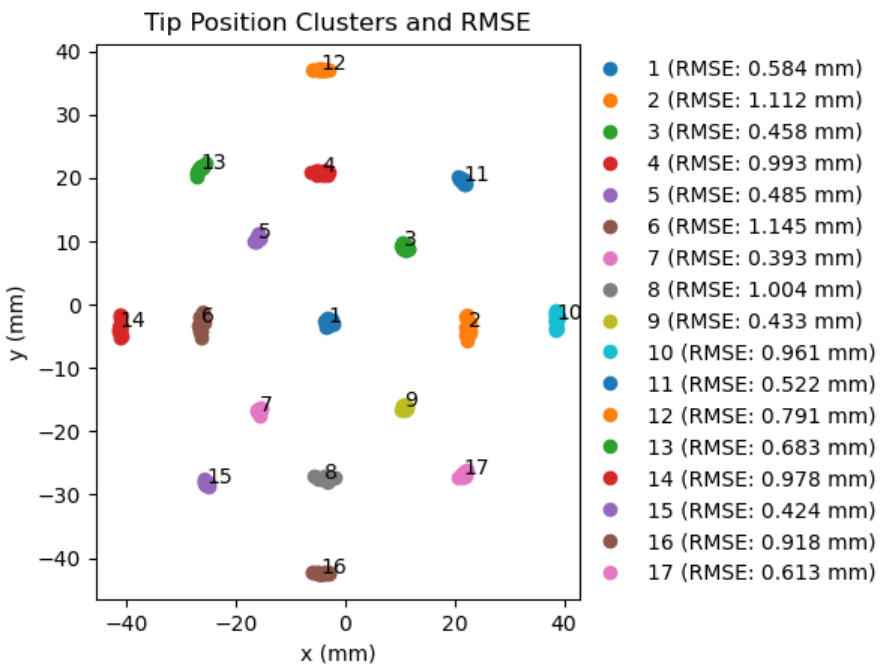


Figure 3.22: Caption

can be seen in Figure 3.22. The Aurora electromagnetic coils claim an RMS position error of 0.48mm over the entire trackable volume. As the robot is positioned such that it is in the center of the tracking volume, this RMS error is likely lower. Therefore, it is clear that hysteresis has an effect on the system, as the RMS position error is as high as 1.181 mm for a cluster of points.

Additionally, there is a statistically significant difference between the RMS position error between points on the cable axes and between the cable axes. The mean RMS position error of the positions on the cable axes is 0.988 mm, compared to 0.501 mm between the axes. A one-way ANOVA test comparing the RMS error on the cable axes vs between the axes produces a P-value of 4.85×10^{-7}, highlighting that there is a significant difference between the two groups.

Because this system experiences hysteresis, specifically around the cable axes, it would make sense that the performance of both the model and the controller would degrade in these areas. For a given cable input, the system can assume many different

tip positions based on where in the workspace the spine is coming from. However, this also provides a potential solution: adding previous inputs to the learned model. By having the learned kinematics model $\hat{f}(u(k), u(k-1))$ be a function of the current and previous input, the effect of hysteresis in the system could also be learned. This is an area for potential future development.

3.4 Conclusion

This chapter covered all of the steps required to develop a learned kinematic model for a continuum robot and how to use it for control. This began with registration and locating where the system is located in 3D space. This was achieved using a method developed to determine both the tip position and orientation of the robot instead of just the tip. By getting an accurate estimate of the tip pose of the robot, the performance of existing constant curvature models could be investigated.

While the two constant curvature models covered differed in implementation, they had, at their core, the same assumption: the system bends in a series of arcs. While this assumption made modeling more tractable, it also significantly hindered the accuracy of these models. This was highlighted in the comparison with the robot's learned kinematics model.

By learning a model directly from data, one sacrifices much of the ability to look under the hood and understand how the model works; the system instead must be treated as a black box. While this is certainly a downside of learned models, in this case, it was worth it as the learned model significantly outperformed the constant curvature alternatives with an RMS error of 2.24 mm vs 9.50 mm for the pure kinematics constant curvature model and 9.46 mm for the statics-based Camarillo model. However, there are still improvements that can be made to this model.

In examining the training dataset for the model, it was observed that despite

providing the system with a uniform distribution of inputs, the output distribution was non-uniform. Looking at the distribution of points of the simulation training data, shown in Figure 3.11, the tip positions are uniformly distributed. The main factor behind this non-uniform distribution was determined to be slack in the cables, as the distribution produced by the slack compensating Camarillo model seen in Figure 3.12 produces similar behavior, clumping the points around the axes.

However, even with noise, the width of these clumps around the cable axes indicates that another factor is impacting the output distribution of the points: hysteresis. This was shown to occur in Figure 3.22, where the tip position errors of points around the workspace were significantly higher than the measurement error expected from the Aurora. Another way to highlight the effects of hysteresis was by adding it to the Camarillo model with slack, shown in Figure 3.21. The presence of hysteresis in the system also presents a way to improve the learned models: using the current input and past inputs to predict the output pose.

However, kinematics models are not the only models that can be learned. When collecting training and test data for the robot, delays were applied to the system to ensure that all movement had ceased before data collection had occurred. This movement would have negatively impacted the performance of the kinematics model. However, it also neglects the system's dynamics, which can also be learned [25], [85]. This raises the question: is it beneficial to learn system dynamics? While this undoubtedly provides the opportunity to develop controllers that operate at higher rates with better performance in relation to the transient response, it also significantly increases the difficulty and for relatively little benefit.

First, to capture the system's dynamics, the learned model would have to include information about the current state of the robot, such as its pose, and the inputs applied to the system [25]. This precludes creating a purely open-loop controller based on the learned model. As a result, the controller would most likely either

have to use Model Predictive Control (MPC) [25] or be learned using Reinforcement Learning (RL) [65]. Second, considering that the robot is intended for surgical use, a very fast transient response is not necesarilly that desireable. Generally, in surgery, all movements are made slowly and carefully. Therefore, having the best transient response is not necessarily a significant benefit. As long as the system works, and it does not endanger their patients, surgeons most likely will not care if the settling time is two seconds as opposed to one second. Because of these factors, unless it can be shown that a dynamic controller can produce significantly higher accuracy than the kinematic model, then it is unlikely to be worth the effort to pursue it.

Despite some of the relative shortcomings of the learned kinematic model, it was used to develop open and closed-loop controllers built entirely upon the learned model. While the controllers developed were relatively simple, they highlight the potential developing controllers based on learned models represents for continuum robot systems. As seen with the control of the simulated continuum robot, by implementing simple open and closed loop controllers, one is able to achieve sub-millimetric position accuracy using these learned models. Even on the real robot, a simple control architecture based on a learned kinematics model was able to track a trajectory accurately. There is obvious room for improvement, but this chapter has shown the potential of learned kinematic models for use with continuum robots.

Chapter 4

Conclusion

Undergoing surgery is a significant stressor on the human body, often requiring long recovery with the potential of morbidity and mortality. By lowering the general level of trauma that the body experiences during the surgery, patients experience better outcomes, spend less time in the hospital, and report higher quality of life [3]. Surgeons, in an effort to reduce trauma, have adopted MIS techniques, including laparoscopy and NOTES, which rely on small incisions or existing orifices to reduce the amount of cutting the surgeon needs to do. These procedures have provide significant improvements to patients in hospital stay durations, complication rates, and recovery time [2] [4]. However, due to limited visual feedback and loss of tactile feedback, surgeons have looked to robots to improve their accuracy in MIS procedures [86]. Robots provide movement filtering and other measures to improve surgeon dexterity and reduce fatigue [5] [87].

While these robots provide greater dexterity in some areas, their large size causes issues when in tight, constricted spaces [87]. Continuum robots provide a potential solution to this issue, as they are designed to be compliant so that they can navigate tight spaces [88]. Continuum robots like the Ion bronchoscope by Intuitive Surgical rely on active and passive deflection to snake their way through a patient's bronchi

[89]. These systems can potentially revolutionize RAS as they can do many of the procedures standard surgical robots cannot. However, there are some obstacles to reaching that point. Due to their infinite degrees of freedom, continuum robots are significantly more challenging to model and control than standard robots [12]. This is forced myself and others to turn to more esoteric models of their behavior.

As part of this thesis, the work done in [34] [35] was built upon to develop a low-cost continuum robot based on the open source ENDO continuum robot proposed in [36]. The contributions in this area can be largely broken into two categories: improvements to the physical system and the development of models and control policies. An issue that plagued the Leigh and Maia designs was cable tear-through, in which the cable would tear through the spine under high tensions. This was remedied by splitting their continuous spines into shorter segments with rigid discs between them. This change preserved the spine's flexibility while using the discs to absorb most of the tensile load in the system. Additionally, the system was made much smaller and more portable, reducing the overall volume of the system from 5.865 L to 0.661 L while retaining a modular design to facilitate the easy development of multisegment continuum robots.

Along with improvements to the spine and frame, improvements were also made to the system's low-level software. This involved improving the reliability of communication with the system through implementing a system based on the BSC communication protocol. A cable tensioning routine that can reliably tension all cables equally and repeatably was also implemented, which removes the need for the cable tensioning pulleys in the Leigh design. These improvements provide better ease of use to future users and developers of the robot.

As alluded to earlier, challenges in modeling continuum robots has been one of the factors limiting their application. Existing models are often derived analytically. While this provides a good understanding of the mechanics of the system, it often

leads to departures from real world behavior as simplifying assumptions must be made to keep the model tractable. A common assumption made is the constant curvature assumption, which assumes that the robot bends in piecewise continuous arcs [10]. To establish a baseline, two constant curvature models were implemented, one based purely on the kinematics of cable displacements and one derived from static analysis of a continuum robot system [17]. These models were tested and found that they produce RMS errors of 9.50 mm and 9.46 mm for a set of points that sweeps the workspace. With this baseline established, other models could be pursued.

Using random input data, a training set of input-output kinematic data was generated and used to train models. In testing these models, an RMS error of 2.24 mm was recorded, which is much lower than the constant curvature models. There is still room to improve, but this highlights the potential learning a model based on input-output data.

This learned model was then applied to develop an optimal open-loop controller and a Jacobian-based closed-loop controller. The model was used to calculate the loss function for the open-loop controller and provide the Jacobian in the closed-loop controller. This was simple due to the existing gradient calculation algorithms built into machine learning libraries (PyTorch). These two controllers were then tasked with following a trajectory through the robot's workspace. The RMS error of the open-loop controller was 2.415 mm and 1.619 mm for the closed-loop controller, showing that these controllers can track a trajectory in their workspace.

4.1 Potential Future Areas of Inquiry

What follows is a discussion of potential areas of inquiry on the different facets of developing a continuum robot, namely the physical robot, modeling, and control.

4.1.1 Physical Improvements

An obvious improvement would be to pursue an additional segment. Adding an additional segment gives the robot additional degrees of freedom, allowing it to explore a larger workspace and have more freedom in terms of orientation. It would also provide an opportunity to explore more complicated behaviors, such as following the leader or snaking through a passageway. In its one-segment configuration, the robot is limited roughly to a surface, whereas the two-segment configuration opens up a volume of potential positions.

Another area for inquiry in the development of the spine would involve more rigorous material testing. Achieving the right balance between flexibility and rigidity in the spine could come from molded silicone parts, or metal flextures. Additionally, intentional variation in the material properties, such as higher stiffness closer to the base of the robot could lead to improved performance. However, this is unachievable using our current 3D printing manufacturing process, therefore this would have to be altered.

4.1.2 Modeling

The successful creation of a learned kinematic model provides a starting point for further development of system models. Developing an understanding of the different sources of error in the model will provide insights on how to better improve models going forward. For example, the presence of hysteresis points to the potential benefit of including previous inputs into the model. Additionally, studying the model's sensitivity to external factors, such as different orientations and attaching loads to the robot's tip, may provide valuable insights into how widely applicable the model is. This would be valuable as the system is used more broadly in various scenarios. This could also lead to the the initial training of a base model, which could then be fine-tuned on a small sample of data collected in a given scenario, allowing the model to correct for

external factors, similar to the encoder structure developed in [68].

The accuracy of the model can be improved in other ways. The most straightforward approach is to combine predictions from multiple technologies, including kinematics and shape sensing, into one larger model. Neural networks are able to learn the shape of continuum robots based on measurements from sensors [26]. Measurements from strain gauges or other sensors could be included in the model's training data to give it more power to discriminate the different poses the robot can take, and improve the accuracy over a standard learned kinematics model. This could be used to remove the reliance on measuring the tip position of the robot for feedback control, instead using shape sensing to estimate the tip positon and use that in a control loop.

Another factor that could be investigated is the dynamics of the robot. All of the models developed were kinematic and assumed stationarity. This was enforced by waiting for the robot to settle before collecting measurements. However, this system has dynamics that. Therefore, a more accurate model would include these factors. The Aurora can produce measurements at a rate of 40 Hz, which is significantly faster than the settling time of the robot, meaning that the system's transient would need to be accounted for. A recurrent model, such as a Neural AutoRegressive eXogenous (NARX) or other recurrent neural network architectures could be used to capture and predict the dynamics of the spine [25]. These models could then be applied to more complicated control policies. However, because this adds additional complexity to both modeling and control, it should be approached only if there is clear benefit to be found in these more complicated models and control policies.

4.1.3 Control

Depending on the desired application, a multitude of different control policies could be investigated. For example, if the robot is engaged in repetitive tasks, such as collecting a series of measurements that follow the same trajectory, techniques such as

Iterative Learning Control (ILC) could be employed to decrease tracking error along the trajectory [90]. This could allow robots to perform tedious yet important actions and free up surgeons to tackle more complicated surgeries. Conversely, if a robot is tasked with more exploratory tasks, techniques such as Reinforcement Learning (RL) could be applied to develop higher level control policies [65]. Using both model free and model based techniques, these systems could develop optimal control policies without the input of a human.

Taking the human out of the loop, however, may not be the end goal. Instead, developing a control policy that is best able to interact with the people operating the robot to form a symbiotic team. This development of human-computer interaction-focused policies would most likely require interaction with human feedback to shape an optimal policy, thus requiring the application of RL to this nebulous, high-level objective function [91]. Overall, there are endless potential control applications for this system. If one can define an objective function to optimize, developing a controller to minimize it is most likely possible.

Developing an accurate, low-cost continuum robot opens up the possibility of unforeseen applications and advances based on the technology. The compliant nature of continuum robots allows them to interact more easily with people than standard rigid robots, which have to take safety concerns very seriously. Ultimately, continuum robots have the ability to work alongside humans, instead of being separate from them.

Appendix A

Bill of Materials

The bill of materials is found on the following pages

Part	Quantity	Material	Sub-Assembly	Vendor	Description
Flexible Spine Segment	4	Elastic 50A Resin	Spine	N/A	The flexible, bending sections of the continuum robot spine.
Rigid Spine Segment	3	Grey or Clear Resin	Spine	N/A	The thin, rigid sections of the continuum robot spine, used to support cable load in spine.
Spine Base Segment	1	Grey or Clear Resin	Spine	N/A	The resin base of the spine that connects to the frame.
Spine Tip Segment	1	Grey or Clear Resin	Spine	N/A	At the tip of the spine, terminates cables, secures Aurora probe, and couples with registration hat.
Registration Hat	1	Grey or Clear Resin	Registration	N/A	Couples with frame top and spine tip to provide known tip registration points.
Frame Top	1	3mm Acryllic	Frame	N/A	Top of frame connects to spine base segment and provides known points for model frame registration.
Frame Wall	4	3mm Acryllic	Frame	N/A	Connects servos to frame, as well as connecting top and base of frame together.
Frame Base	1	3mm Acryllic	Frame	N/A	Provides foundation for frame. Connects to frame walls and vertical cable guides.
Cable Guide	1	3mm Acryllic	Frame	N/A	Routes cables from pulleys to spine. Connects frame walls together.

Part	Quantity	Material	Sub-Assembly	Vendor	Description
Vertical Cable Guide	1	Grey or Clear Resin	Frame	N/A	Prevents cables from falling off of pulleys. Connects to frame base.
Berkley FireLine Super Line Crystal 30lb	2 Ft	N/A	Actuation	Tackle Warehouse	Used as the cables in the robot. Connect the servos to the spine. 0.38mm OD
Servo Pulleys	4	PLA	Actuation	N/A	30mm OD pulleys that the servos use to pull on the cables. 25T Spline matches servo output.
Futaba Universal U-305 Servo	4	N/A	Actuation	Futaba USA	Servos used to actuate robot. +- 60 degree digital servos. Rated 8 kg
Molex Electrical Crimp Connectors	4	Aluminum			Used to terminate cables on pulley sides
Arduino Due	1	N/A	Electronics	Arduino	Microcontroller used to communicate with laptop and send commands to servo driver.
6-Channel 12-bit PWM/Servo Driver	1	N/A	Electronics	Adafruit	PWM driver connected to microcontroller via I2C used to drive servos.
NDI Aurora Field Generator	1	N/A	Electronics	NDI	Field generator that interacts with coils to track robot tip pose.
NDI Aurora 6-DOF Coil	1	N/A	Electronics	NDI	Coil that interacts with field and provides tip pose information.
$4-40 \times 1/2$" Socket Head Cap Screw	4	Steel	Fasteners	McMaster Carr	Bolt used to secure spine base to frame top.
4-40 Nut	4	Steel	Fasteners	McMaster Carr	Nut used to secure spine base to frame top.

Part	Quantity	Material	Sub-Assembly	Vendor	Description
$8-32 \times 1/2$" Socket Head Cap Screw	16	Steel	Fasteners	McMaster Carr	Bolt used to secure servos to frame walls.
8-32 Nut	16	Steel	Fasteners	McMaster Carr	Nut used to secure servos to frame walls.

Table A.1: Bill of materials for continuum robot system.

Appendix B

Software and Code

All code for this project can be found in the project's GitHub repository: `https://github.com/Camty1/continuum_cam`.

Appendix C

Spine Parameters

Flexible Section Properties	
Number of Cables	4
Section Length (A)	14 mm
Outer Diameter (B)	12 mm
Cable Sleeve Diameter (C)	2 mm
Cable Sleeve Height (D)	2 mm
Central Hole Diameter (E)	4 mm
Column Thickness (F)	2 mm
Straight Length (G)	1 mm
Bend Angle (H)	25°

Table C.1: Parametric design parameters of flexible sections.

Appendix D

Two Segment Training

Due to the poor performance of the learned kinematic model for the two-segment simulated robot in Section 3.2.2, further investigation into improving the performance of these models follows in this Appendix. To achieve this, two factors were chosen: changing the training dataset's size and network size. For testing the effect of dataset size, models with two hidden layers of 32 neurons were trained on datasets of $2^13 = 8,192$, $2^14 = 16,348$, or $2^15 = 32,786$ points. For testing the effect of model size, models with two hidden layers of 32, 64, or 128 neurons were trained on datasets of $2^14 = 16,348$ points. These were tested by training five models for each configuration to determine the effect on training and performance.

For all of the models, training was performed on a dataset of uniformly distributed random inputs over the interval (-12 mm, 12 mm). The training data was then used with a 75%-25% split into a training set and a validation set to prevent overfitting. To tune the parameters of the model, the ADAM optimizer was used with a learning rate of 1e-3. After training for 2048 epochs, the parameters of the model during the epoch of lowest validation loss were saved for testing. The training and validation loss for all six of the different training scenarios are in Figure D.1.

As can be seen, changing the size of the training dataset does not improve the

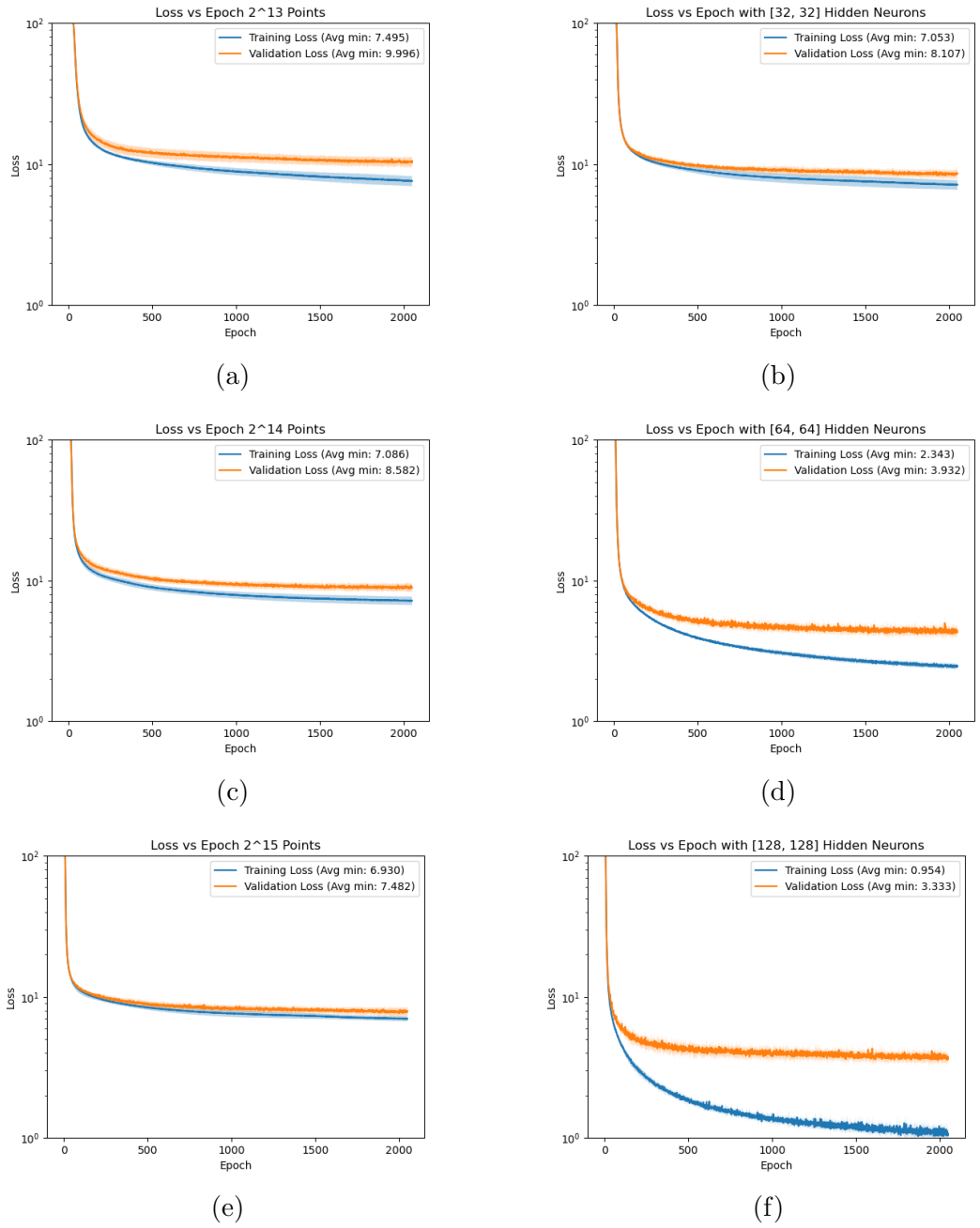


Figure D.1: Training and validation losses for the six training configurations. In the left column, the size of the training dataset is varied, in the right column, the size of the model is varied.

.

accuracy of the model substantially. The average minimum training loss of the three different dataset sizes are roughly the same. However, increasing the size of the dataset does reduce the difference between the minimum training loss and the minimum validation loss (2.501, 1.492, 0.550 for 2^{13}, 2^{14}, and 2^{15} points respectively). Therefore, by increasing the size of the dataset, the models produced are better able to generalize beyond the training data. Increasing the size of the model, however, improves the overall performance of the model. This improved performance, however, can also lead to overfitting, as can be seen in Figure D.1f, where the training and validation losses are relatively much further apart. Therefore, increased model size needs to be coupled with larger datasets to prevent this from occurring.

To test the models, an additional random input dataset of 512 points was generated. The models' predictions were then compared with the true positions and orientations. The results of this test can be seen in Figure D.2.

As can be seen, changing the number of points in the dataset improves the performance of the models, but only slightly. This is most likely caused by the better generalization seen with larger datasets (much like how the validation loss was slightly lower). However, the size of the model ultimately determines the ceiling on the performance of the model, regardless of the number of data points. This is clearly shown in the significant improvements to both position error and orientation error when increasing the size of the model.

To combine these two effects and hopefully produce a model that has the improved generalization of a larger dataset and the improved performance of a larger model, a final five models with two hidden layers of 128 neurons were trained on a dataset containing 2^{15} points. Their training losses and test performances can be seen in Figures D.3 and D.4. These models have significantly improved performance over the original models trained in Section 3.2.2 on two segment models, showing that even if the number of segments in the robot is increased, by increasing the size of the training

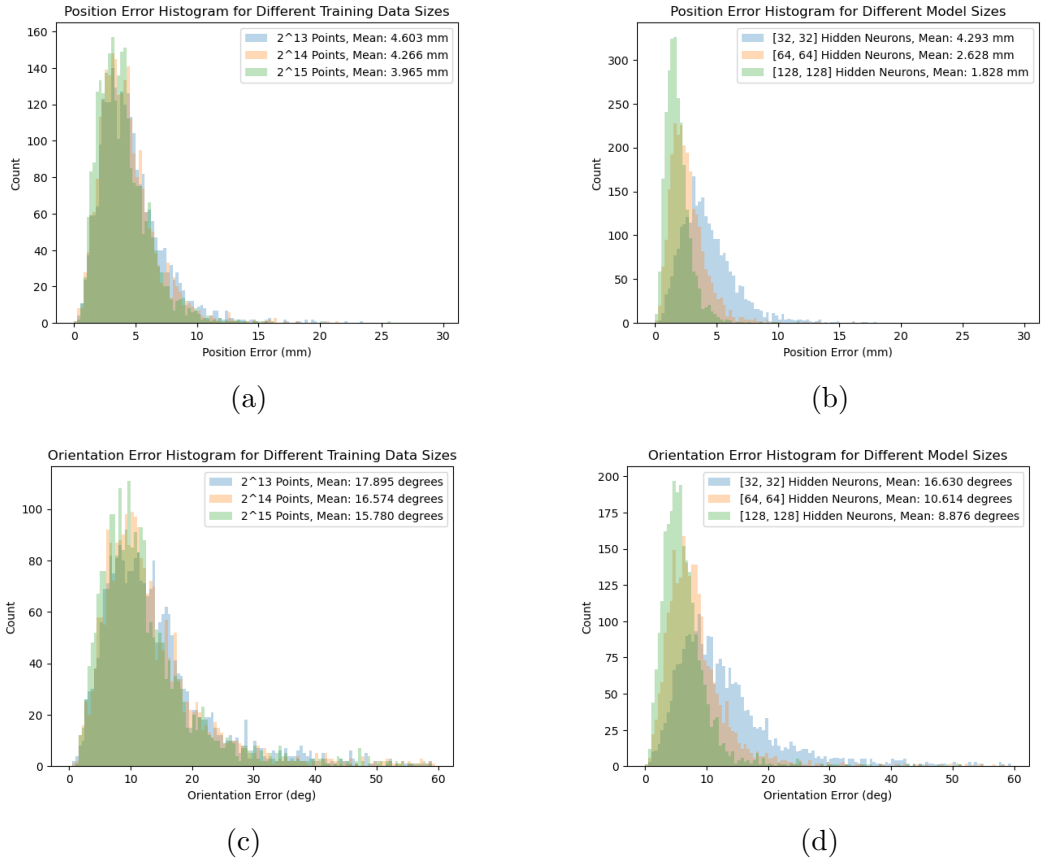


Figure D.2: Position and orientation errors for the various data sizes and model sizes. In the left column, the size of the training dataset is varied, in the right column, the size of the model is varied.

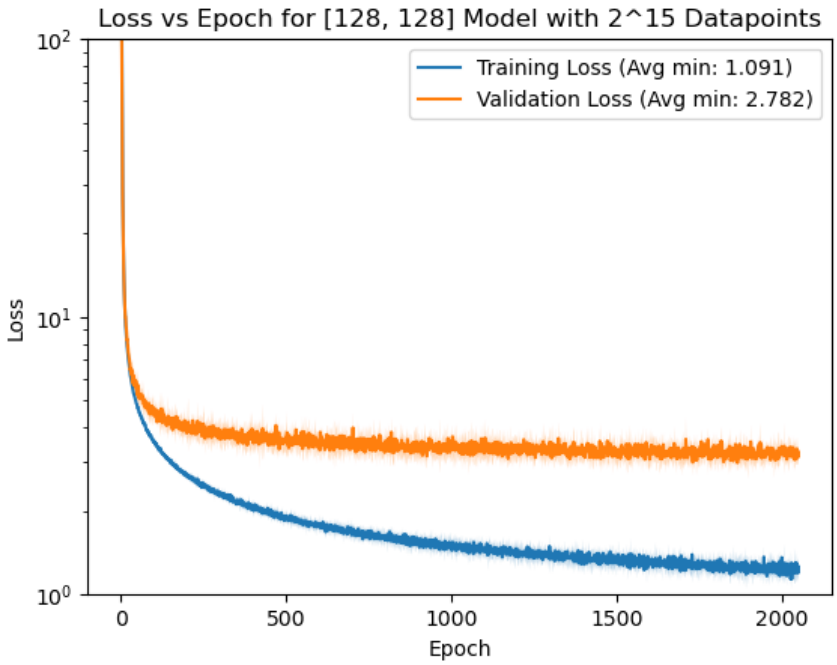
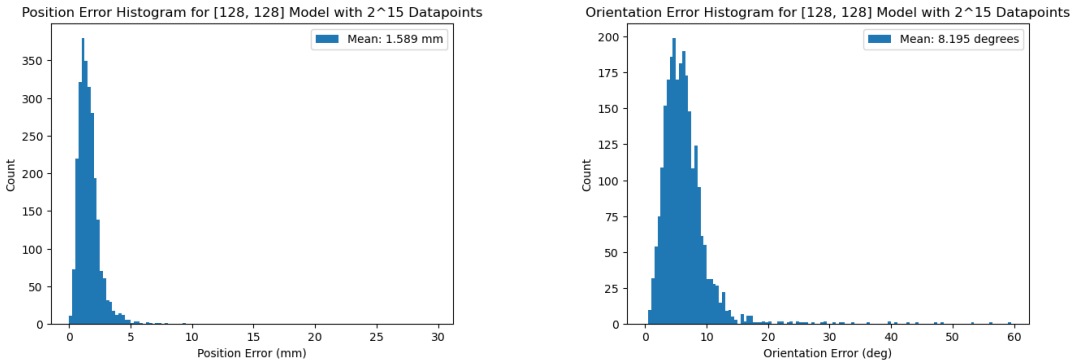


Figure D.3: Using both a larger model and a larger dataset, the training loss closely matches that of the training loss in the big model with less data, however, the validation loss in this model is lower, meaning that the model is better able to generalize to new data.

data and the model, one can still achieve solid performance using a learned model.



(a) (b)

Figure D.4: This larger model, trained on a larger dataset, performs the best of the two-segment learned models, with a comparable position and orientation error compared with the one-segment models.

Bibliography

[1] G. P. Dobson, "Trauma of major surgery: A global problem that is not going away", *International Journal of Surgery (London, England)*, vol. 81, pp. 47–54, Sep. 2020, ISSN: 1743-9191. DOI: 10.1016/j.ijsu.2020.07.017. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7388795/ (visited on 03/18/2024).

[2] B. S. Peters, P. R. Armijo, C. Krause, S. A. Choudhury, and D. Oleynikov, "Review of emerging surgical robotic technology", en, *Surgical Endoscopy*, vol. 32, no. 4, pp. 1636–1655, Apr. 2018, ISSN: 1432-2218. DOI: 10.1007/s00464-018-6079-2. [Online]. Available: https://doi.org/10.1007/s00464-018-6079-2 (visited on 03/18/2024).

[3] H. Li, J. Zheng, J.-Y. Cai, *et al.*, "Laparoscopic VS open hepatectomy for hepatolithiasis: An updated systematic review and meta-analysis", *World Journal of Gastroenterology*, vol. 23, no. 43, pp. 7791–7806, Nov. 2017, ISSN: 1007-9327. DOI: 10.3748/wjg.v23.i43.7791. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5703939/ (visited on 03/18/2024).

[4] R. Agha and G. Muir, "Does laparoscopic surgery spell the end of the open surgeon?", *Journal of the Royal Society of Medicine*, vol. 96, no. 11, pp. 544–546, Nov. 2003, ISSN: 0141-0768. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC539626/ (visited on 03/18/2024).

[5] M. Orady, A. Hrynewych, A. K. Nawfal, and G. Wegienka, "Comparison of Robotic-Assisted Hysterectomy to Other Minimally Invasive Approaches", *JSLS : Journal of the Society of Laparoendoscopic Surgeons*, vol. 16, no. 4, pp. 542–548, 2012, ISSN: 1086-8089. DOI: 10.4293/108680812X13462882736899. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3558889/ (visited on 03/18/2024).

[6] E. Ritchie and E. A. Landis, "Industrial Robotics in Manufacturing", en, *Journal of Leadership, Accountability and Ethics*, vol. 18, no. 2, Jun. 2021, Number: 2. [Online]. Available: https://articlearchives.co/index.php/JLAE/article/view/4000 (visited on 03/19/2024).

[7] S. Yee, "Transoral Robotic Surgery", en, *AORN Journal*, vol. 105, no. 1, pp. 73–84, 2017, ISSN: 1878-0369. DOI: 10.1016/j.aorn.2016.11.011. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1016/j.aorn.2016.11.011 (visited on 06/07/2024).

[8] Z. N. Maan, N. Gibbins, T. Al-Jabri, and A. R. D'Souza, "The use of robotics in otolaryngology–head and neck surgery: A systematic review", *American Journal of Otolaryngology*, vol. 33, no. 1, pp. 137–146, Jan. 2012, ISSN: 0196-0709. DOI: 10.1016/j.amjoto.2011.04.003. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0196070911000779 (visited on 05/14/2024).

[9] A. Agrawal, D. K. Hogarth, and S. Murgu, "Robotic bronchoscopy for pulmonary lesions: A review of existing technologies and clinical data", *Journal of Thoracic Disease*, vol. 12, no. 6, pp. 3279–3286, Jun. 2020, ISSN: 2072-1439. DOI: 10.21037/jtd.2020.03.35. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7330790/ (visited on 05/14/2024).

[10] R. J. Webster and B. A. Jones, "Design and Kinematic Modeling of Constant Curvature Continuum Robots: A Review", en, *The International Journal of Robotics Research*, vol. 29, no. 13, pp. 1661–1683, Nov. 2010, Publisher: SAGE Publications Ltd STM, ISSN: 0278-3649. DOI: 10.1177/0278364910368147. [Online]. Available: https://doi.org/10.1177/0278364910368147 (visited on 07/31/2023).

[11] I. D. Walker, "Continuous Backbone "Continuum" Robot Manipulators", en, *International Scholarly Research Notices*, vol. 2013, e726506, Jul. 2013, Publisher: Hindawi. DOI: 10.5402/2013/726506. [Online]. Available: https://www.hindawi.com/journals/isrn/2013/726506/ (visited on 05/14/2024).

[12] T. d. Veiga, J. H. Chandler, P. Lloyd, *et al.*, "Challenges of continuum robots in clinical context: A review", en, *Progress in Biomedical Engineering*, vol. 2, no. 3, p. 032 003, Jul. 2020, Publisher: IOP Publishing, ISSN: 2516-1091. DOI: 10.1088/2516-1091/ab9f41. [Online]. Available: https://dx.doi.org/10.1088/2516-1091/ab9f41 (visited on 07/31/2023).

[13] T. M. Bieze, "Contribution to the kinematic modeling and control of soft manipulators using computational mechanics", en, Ph.D. dissertation, Université de Lille, Oct. 2017. [Online]. Available: https://hal.science/tel-03516545 (visited on 05/14/2024).

[14] E. Harsono, J. Yang, S. Bhattacharya, and H. Yu, "Design and analysis of a novel hybrid-driven continuum robot with variable stiffness", *Mechanism and Machine Theory*, vol. 177, p. 105 067, Nov. 2022, ISSN: 0094-114X. DOI: 10.1016/j.mechmachtheory.2022.105067. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0094114X22003147 (visited on 03/29/2024).

[15] R. Hartenberg and J. Danavit, *Kinematic synthesis of linkages*. New York: McGraw-Hill, 1964.

[16] R. J. Roesthuis and S. Misra, "Steering of Multisegment Continuum Manipulators Using Rigid-Link Modeling and FBG-Based Shape Sensing", *IEEE Transactions on Robotics*, vol. 32, no. 2, pp. 372–382, Apr. 2016, Conference Name: IEEE Transactions on Robotics, ISSN: 1941-0468. DOI: 10.1109/TRO.2016.2527047. [Online]. Available: https://ieeexplore.ieee.org/document/7437492 (visited on 05/15/2024).

[17] D. B. Camarillo, C. R. Carlson, and J. K. Salisbury, "Configuration Tracking for Continuum Manipulators With Coupled Tendon Drive", *IEEE Transactions on Robotics*, vol. 25, no. 4, pp. 798–808, Aug. 2009, Conference Name: IEEE Transactions on Robotics, ISSN: 1941-0468. DOI: 10.1109/TRO.2009.2022426.

[18] M. W. Hannan and I. D. Walker, "Kinematics and the Implementation of an Elephant's Trunk Manipulator and Other Continuum Style Robots", en, *Journal of Robotic Systems*, vol. 20, no. 2, pp. 45–63, 2003, ISSN: 1097-4563. DOI: 10.1002/rob.10070. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.10070 (visited on 10/19/2023).

[19] K. Oliver-Butler, J. Till, and C. Rucker, "Continuum Robot Stiffness Under External Loads and Prescribed Tendon Displacements", *IEEE Transactions on Robotics*, vol. 35, no. 2, pp. 403–419, Apr. 2019, Conference Name: IEEE Transactions on Robotics, ISSN: 1941-0468. DOI: 10.1109/TRO.2018.2885923. [Online]. Available: https://ieeexplore.ieee.org/document/8606257 (visited on 09/24/2023).

[20] D. C. Rucker and R. J. Webster, "Deflection-based force sensing for continuum robots: A probabilistic approach", in *2011 IEEE/RSJ International Conference*

on Intelligent Robots and Systems, ISSN: 2153-0866, Sep. 2011, pp. 3764–3769. DOI: 10.1109/IROS.2011.6094526.

[21] F. Renda, F. Boyer, J. Dias, and L. Seneviratne, "Discrete Cosserat Approach for Multisection Soft Manipulator Dynamics", *IEEE Transactions on Robotics*, vol. 34, no. 6, pp. 1518–1533, Dec. 2018, Conference Name: IEEE Transactions on Robotics, ISSN: 1941-0468. DOI: 10.1109/TRO.2018.2868815. [Online]. Available: https://ieeexplore.ieee.org/document/8500341 (visited on 12/28/2023).

[22] T. M. Bieze, F. Largilliere, A. Kruszewski, Z. Zhang, R. Merzouki, and C. Duriez, "Finite Element Method-Based Kinematics and Closed-Loop Control of Soft, Continuum Manipulators", *Soft Robotics*, vol. 5, no. 3, pp. 348–364, Jun. 2018, Publisher: Mary Ann Liebert, Inc., publishers, ISSN: 2169-5172. DOI: 10.1089/soro.2017.0079. [Online]. Available: https://www.liebertpub.com/doi/abs/10.1089/soro.2017.0079 (visited on 04/15/2024).

[23] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning", en, *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015, Number: 7553 Publisher: Nature Publishing Group, ISSN: 1476-4687. DOI: 10.1038/nature14539. [Online]. Available: https://www.nature.com/articles/nature14539 (visited on 09/29/2023).

[24] G. Cybenko, "Approximation by superpositions of a sigmoidal function", en, *Mathematics of Control, Signals and Systems*, vol. 2, no. 4, pp. 303–314, Dec. 1989, ISSN: 1435-568X. DOI: 10.1007/BF02551274. [Online]. Available: https://doi.org/10.1007/BF02551274 (visited on 09/29/2023).

[25] T. G. Thuruthel, E. Falotico, F. Renda, and C. Laschi, "Learning dynamic models for open loop predictive control of soft robotic manipulators", en, *Bioinspiration & Biomimetics*, vol. 12, no. 6, p. 066 003, Oct. 2017, Publisher: IOP Publishing,

ISSN: 1748-3190. DOI: 10.1088/1748-3190/aa839f. [Online]. Available: https://dx.doi.org/10.1088/1748-3190/aa839f (visited on 12/30/2023).

[26] A. Kuntz, A. Sethi, R. J. Webster, and R. Alterovitz, "Learning the Complete Shape of Concentric Tube Robots", *IEEE transactions on medical robotics and bionics*, vol. 2, no. 2, pp. 140–147, May 2020, ISSN: 2576-3202. DOI: 10.1109/tmrb.2020.2974523. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7243456/ (visited on 04/29/2024).

[27] K. D. Miller, R. L. Siegel, C. C. Lin, *et al.*, "Cancer treatment and survivorship statistics, 2016", en, *CA: A Cancer Journal for Clinicians*, vol. 66, no. 4, pp. 271–289, 2016, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.3322/caac.21349, ISSN: 1542-4863. DOI: 10.3322/caac.21349. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.3322/caac.21349 (visited on 06/26/2024).

[28] Catalona William J., Smith Deborah S., Ratliff Timothy L., *et al.*, "Measurement of Prostate-Specific Antigen in Serum as a Screening Test for Prostate Cancer", *New England Journal of Medicine*, vol. 324, no. 17, pp. 1156–1161, Apr. 1991. DOI: 10.1056/NEJM199104253241702. [Online]. Available: https://www.nejm.org/doi/full/10.1056/NEJM199104253241702 (visited on 06/26/2024).

[29] C. J. Harvey, J. Pilcher, J. Richenberg, U. Patel, and F. Frauscher, "Applications of transrectal ultrasound in prostate cancer", *British Journal of Radiology*, vol. 85, no. special_issue_1, S3–S17, Nov. 2012, ISSN: 0007-1285. DOI: 10.1259/bjr/56357549. [Online]. Available: https://doi.org/10.1259/bjr/56357549 (visited on 06/26/2024).

[30] A. J. Wein, L. R. Kavoussi, A. C. Novick, A. W. Partin, and C. A. Peters, *Campbell-Walsh Urology*, en. Elsevier Health Sciences, Sep. 2011, Google-Books-ID: W1aeyJD46kIC, ISBN: 978-1-4557-2298-3.

[31] M. Bodenstein, M. David, and K. Markstaller, "Principles of electrical impedance tomography and its clinical application", en-US, *Critical Care Medicine*, vol. 37, no. 2, p. 713, Feb. 2009, ISSN: 0090=3493. DOI: 10.1097/CCM.0b013e3181958d2f. [Online]. Available: https://journals.lww.com/ccmjournal/fulltext/2009/02000/Regional_lung_volume_during_high_frequency.00043.aspx (visited on 06/26/2024).

[32] R. J. Halter, A. Hartov, J. A. Heaney, K. D. Paulsen, and A. R. Schned, "Electrical Impedance Spectroscopy of the Human Prostate", *IEEE Transactions on Biomedical Engineering*, vol. 54, no. 7, pp. 1321–1327, Jul. 2007, Conference Name: IEEE Transactions on Biomedical Engineering, ISSN: 1558-2531. DOI: 10.1109/TBME.2007.897331. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/4237338 (visited on 06/26/2024).

[33] E. K. Murphy, X. Wu, and R. J. Halter, "Fused-data transrectal EIT for prostate cancer imaging", en, *Physiological Measurement*, vol. 39, no. 5, p. 054 005, May 2018, Publisher: IOP Publishing, ISSN: 0967-3334. DOI: 10.1088/1361-6579/aabf22. [Online]. Available: https://dx.doi.org/10.1088/1361-6579/aabf22 (visited on 06/26/2024).

[34] B. Leigh, "Control of a Continuum Robot for Applications in Minimally Invasive Surgery", eng, Undergraduate, Dartmouth College, Hanover, New Hampshire, Jun. 2023.

[35] C. Lago Pena Maia, "Introducing the Design and Inverse Model of a Continuum Manipulator for Surgical Use", English, Undergraduate, Dartmouth College, Hanover, New Hampshire, Jun. 2022.

[36] A. B. Clark, V. Mathivannan, and N. Rojas, "A Continuum Manipulator for Open-Source Surgical Robotics Research and Shared Development", *IEEE Transactions on Medical Robotics and Bionics*, vol. 3, no. 1, pp. 277–280, Feb.

2021, arXiv:2101.01080 [cs], ISSN: 2576-3202. DOI: 10.1109/TMRB.2020.3041391. [Online]. Available: http://arxiv.org/abs/2101.01080 (visited on 08/20/2023).

[37] *Form 3+: Industrial-Quality Desktop Resin 3D Printer*, en. [Online]. Available: https://formlabs.com/3d-printers/form-3/ (visited on 03/20/2024).

[38] *Resin Family: Flexible and Elastic*, en. [Online]. Available: https://formlabs.com/materials/flexible-elastic/ (visited on 05/14/2024).

[39] R. Kang, Y. Guo, L. Chen, D. T. Branson, and J. S. Dai, "Design of a Pneumatic Muscle Based Continuum Robot With Embedded Tendons", *IEEE/ASME Transactions on Mechatronics*, vol. 22, no. 2, pp. 751–761, Apr. 2017, Conference Name: IEEE/ASME Transactions on Mechatronics, ISSN: 1941-014X. DOI: 10.1109/TMECH.2016.2636199. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7775004?casa_token=LVLapwYWEb8AAAAA:zwCDehzymrzpMwY00JcgqzSZlCdDuNVIbN36EiKXyKVft0Jg2SwDnD7Z0Uv6HiEwe0TnOh428w (visited on 03/29/2024).

[40] G. Wu and G. Shi, "Design, modeling, and workspace analysis of an extensible rod-driven Parallel Continuum Robot", *Mechanism and Machine Theory*, vol. 172, p. 104 798, Jun. 2022, ISSN: 0094-114X. DOI: 10.1016/j.mechmachtheory.2022.104798. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0094114X22000702 (visited on 03/29/2024).

[41] D. B. Camarillo, C. F. Milne, C. R. Carlson, M. R. Zinn, and J. K. Salisbury, "Mechanics Modeling of Tendon-Driven Continuum Manipulators", *IEEE Transactions on Robotics*, vol. 24, no. 6, pp. 1262–1273, Dec. 2008, Conference Name: IEEE Transactions on Robotics, ISSN: 1941-0468. DOI: 10.1109/TRO.2008.2002311.

[42] R. C. Ramanathan, W. Gourash, and P. R. Schauer, "EQUIPMENT AND INSTRUMENTATION FOR LAPAROSCOPIC BARIATRIC SURGERY", en,

[43] Alguém, *Disposable Trocar systems comprised of an Obturator, Seal and Cannula*, Nov. 2008. [Online]. Available: `https://commons.wikimedia.org/wiki/File:Disposable_Trocars.jpg` (visited on 03/27/2024).

[44] A. Su and S. J. Al'Aref, "Chapter 1 - History of 3D Printing", in *3D Printing Applications in Cardiovascular Medicine*, S. J. Al'Aref, B. Mosadegh, S. Dunham, and J. K. Min, Eds., Boston: Academic Press, Jan. 2018, pp. 1–10, ISBN: 978-0-12-803917-5. DOI: `10.1016/B978-0-12-803917-5.00001-8`. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/B9780128039175000018` (visited on 04/05/2024).

[45] *How Stereolithography Or SLA 3D Printing Works? - Manufactur3D*, en-US, Section: EDUCATION, Jan. 2018. [Online]. Available: `https://manufactur3dmag.com/stereolithography-sla-3d-printing-works/` (visited on 04/01/2024).

[46] K. Szykiedans and W. Credo, "Mechanical Properties of FDM and SLA Low-cost 3-D Prints", *Procedia Engineering*, The 20th International Conference Machine Modeling and Simulations, MMS 2015 September 7-9, 2015, Terchová, Slovak Republic organized by: Faculty of Industrial Technologies in Púcho Alexander Dubček University of Trenčín, vol. 136, pp. 257–262, Jan. 2016, ISSN: 1877-7058. DOI: `10.1016/j.proeng.2016.01.207`. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S1877705816002113` (visited on 04/05/2024).

[47] S. R. Gaikwad, N. H. Pawar, and S. U. Sapkal, "Comparative evaluation of 3D printed components for deviations in dimensional and geometrical features", *Materials Today: Proceedings*, Third International Conference on Recent Advances

in Materials and Manufacturing 2021, vol. 59, pp. 297–304, Jan. 2022, ISSN: 2214-7853. DOI: 10.1016/j.matpr.2021.11.157. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214785321071649 (visited on 04/05/2024).

[48] *Elastic 50A Resin V2*, en. [Online]. Available: https://formlabs.com/store/materials/elastic-50a-resin-v2/ (visited on 03/20/2024).

[49] A. Yamada, W. Yonemichi, O. Inatomi, A. Andoh, and T. Tani, "Steerable catheter based on wire-driven seamless artificial blood vessel tube for endoscopic retrograde transpapillary interventions", eng, *International Journal of Computer Assisted Radiology and Surgery*, vol. 18, no. 3, pp. 433–447, Mar. 2023, ISSN: 1861-6429. DOI: 10.1007/s11548-022-02805-x.

[50] *Grey Resin*, en. [Online]. Available: https://formlabs.com/store/materials/grey-resin/ (visited on 04/02/2024).

[51] *Best-in-class 3D Printers*, en. [Online]. Available: https://formlabs.com/store/materials/clear-resin/ (visited on 05/14/2024).

[52] J. Cullen, "Binary Synchronous Communications", *IEEE Transactions on Communication Technology*, vol. 17, no. 6, pp. 654–658, Dec. 1969, Conference Name: IEEE Transactions on Communication Technology, ISSN: 2162-2175. DOI: 10.1109/TCOM.1969.1090148. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/1090148?casa_token=YkNTju3lGOsAAAAA:s7C6UK_jCE5WqaylX0NIc3vWTarErBHGPocL4GI2R41eFtAjc35DaHCGoSDO27dJ50t8D9xWJw (visited on 04/05/2024).

[53] J. S. Sobolewski, "Cyclic redundancy check", in *Encyclopedia of Computer Science*, GBR: John Wiley and Sons Ltd., Jan. 2003, pp. 476–479, ISBN: 978-0-470-86412-8. (visited on 04/08/2024).

[54] W. W. Peterson and D. T. Brown, "Cyclic Codes for Error Detection", *Proceedings of the IRE*, vol. 49, no. 1, pp. 228–235, Jan. 1961, Conference Name: Proceedings of the IRE, ISSN: 2162-6634. DOI: 10.1109/JRPROC.1961.287814. [Online]. Available: https://ieeexplore.ieee.org/document/4066263 (visited on 06/28/2024).

[55] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-Squares Fitting of Two 3-D Point Sets", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-9, no. 5, pp. 698–700, Sep. 1987, Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence, ISSN: 1939-3539. DOI: 10.1109/TPAMI.1987.4767965. [Online]. Available: https://ieeexplore.ieee.org/document/4767965 (visited on 04/16/2024).

[56] O. Sorkine-Hornung and M. Rabinovich, "Least-squares rigid motion using svd", *Computing*, vol. 1, no. 1, pp. 1–5, 2017.

[57] Georg-Johann, *English: Visual representation of a singular value decomposition (SVD) of the 2-dimensional real shearing*, Aug. 2010. [Online]. Available: https://commons.wikimedia.org/wiki/File:Singular-Value-Decomposition.svg (visited on 04/16/2024).

[58] F. L. Markley, Y. Cheng, J. L. Crassidis, and Y. Oshman, "Averaging Quaternions", *Journal of Guidance, Control, and Dynamics*, vol. 30, no. 4, pp. 1193–1197, 2007, Publisher: American Institute of Aeronautics and Astronautics _eprint: https://doi.org/10.2514/1.28949, ISSN: 0731-5090. DOI: 10.2514/1.28949. [Online]. Available: https://doi.org/10.2514/1.28949 (visited on 04/17/2024).

[59] B. A. Jones and I. D. Walker, "Practical Kinematics for Real-Time Implementation of Continuum Robots", *IEEE Transactions on Robotics*, vol. 22, no. 6,

pp. 1087–1099, Dec. 2006, Conference Name: IEEE Transactions on Robotics, ISSN: 1941-0468. DOI: 10.1109/TRO.2006.886268.

[60] D. B. Camarillo, C. R. Carlson, and J. K. Salisbury, "Task-Space Control of Continuum Manipulators with Coupled Tendon Drive", en, in *Experimental Robotics*, O. Khatib, V. Kumar, and G. J. Pappas, Eds., ser. Springer Tracts in Advanced Robotics, Berlin, Heidelberg: Springer, 2009, pp. 271–280, ISBN: 978-3-642-00196-3. DOI: 10.1007/978-3-642-00196-3_32.

[61] D. C. Rucker and R. J. Webster III, "Statics and Dynamics of Continuum Robots With General Tendon Routing and External Loading", *IEEE Transactions on Robotics*, vol. 27, no. 6, pp. 1033–1044, Dec. 2011, Conference Name: IEEE Transactions on Robotics, ISSN: 1941-0468. DOI: 10.1109/TRO.2011.2160469.

[62] H. Yuan, L. Zhou, and W. Xu, "A comprehensive static model of cable-driven multi-section continuum robots considering friction effect", *Mechanism and Machine Theory*, vol. 135, pp. 130–149, May 2019, ISSN: 0094-114X. DOI: 10.1016/j.mechmachtheory.2019.02.005. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0094114X1831468X (visited on 09/24/2023).

[63] J. Bosman, T. M. Bieze, O. Lakhal, M. Sanz, R. Merzouki, and C. Duriez, "Domain decomposition approach for FEM quasistatic modeling and control of Continuum Robots with rigid vertebras", in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, ISSN: 1050-4729, May 2015, pp. 4373–4378. DOI: 10.1109/ICRA.2015.7139803. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7139803?casa_token=7JWALXbLr28AAAAA:7JqD1QqXGvWAlf54iyS0Ra_WALmGrmi5VddDqyRpmLya_kN-WI-0C7iSkseqXH7h_s-baY9kew (visited on 04/15/2024).

[64] R. Grassmann, V. Modes, and J. Burgner-Kahrs, "Learning the Forward and Inverse Kinematics of a 6-DOF Concentric Tube Continuum Robot in SE(3)", in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, ISSN: 2153-0866, Oct. 2018, pp. 5125–5132. DOI: 10.1109/IROS.2018.8594451. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8594451?casa_token=smrfGv3Mvc8AAAAA:L4_8eEVGWhNbVQQUEjUB3KDk6mccOTk8P1khhGRsMZXuKnZH_lArGIcQFjL2VbHtjfcZTRAp3A (visited on 04/15/2024).

[65] T. G. Thuruthel, E. Falotico, F. Renda, and C. Laschi, "Model-Based Reinforcement Learning for Closed-Loop Dynamic Control of Soft Robotic Manipulators", *IEEE Transactions on Robotics*, vol. 35, no. 1, pp. 124–134, Feb. 2019, Conference Name: IEEE Transactions on Robotics, ISSN: 1941-0468. DOI: 10.1109/TRO.2018.2878318. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8531756?casa_token=h36PW_7ATzcAAAAA:iRQ84aFwS1RSuEX6vOPGcUW3P5RxI6zYSY4IjC5hXh0LnH26eBAd103BKINDnyQJTVEI2nGTfw0 (visited on 12/14/2023).

[66] D. Kubus, R. Rayyes, and J. J. Steil, "Learning Forward and Inverse Kinematics Maps Efficiently", in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, ISSN: 2153-0866, Oct. 2018, pp. 5133–5140. DOI: 10.1109/IROS.2018.8593833. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8593833?casa_token=w4vmMtTY4AgAAAAA:HAXj1C3afzPSdBSS_VvIjzFbxmIbokoEL_8m6lGURcyPQ89caBc7Cq7Qf_K4NYEeO6d4EcrURQ (visited on 04/15/2024).

[67] B. Baker, I. Akkaya, P. Zhokhov, *et al.*, *Video PreTraining (VPT): Learning to Act by Watching Unlabeled Online Videos*, arXiv:2206.11795 [cs], Jun. 2022. DOI: 10.48550/arXiv.2206.11795. [Online]. Available: http://arxiv.org/abs/2206.11795 (visited on 08/06/2023).

[68] M. Schwarzer, N. Rajkumar, M. Noukhovitch, *et al.*, "Pretraining Representations for Data-Efficient Reinforcement Learning", in *Advances in Neural Information Processing Systems*, vol. 34, Curran Associates, Inc., 2021, pp. 12 686–12 699. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2021/hash/69eba34671b3ef1ef38ee85caae6b2a1-Abstract.html (visited on 04/15/2024).

[69] J. C. A. Barata and M. S. Hussein, "The Moore-Penrose Pseudoinverse. A Tutorial Review of the Theory", *Brazilian Journal of Physics*, vol. 42, no. 1-2, pp. 146–165, Apr. 2012, arXiv:1110.6882 [math-ph], ISSN: 0103-9733, 1678-4448. DOI: 10.1007/s13538-011-0052-z. [Online]. Available: http://arxiv.org/abs/1110.6882 (visited on 04/28/2024).

[70] J. Starke, E. Amanov, M. T. Chikhaoui, and J. Burgner-Kahrs, "On the merits of helical tendon routing in continuum robots", in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, ISSN: 2153-0866, Sep. 2017, pp. 6470–6476. DOI: 10.1109/IROS.2017.8206554.

[71] G. P. Tolstov, *Fourier Series*, en. Courier Corporation, Mar. 2012, Google-Books-ID: _amjAQAAQBAJ, ISBN: 978-0-486-14174-9.

[72] S. R. Dubey, S. K. Singh, and B. B. Chaudhuri, "Activation functions in deep learning: A comprehensive survey and benchmark", *Neurocomputing*, vol. 503, pp. 92–108, Sep. 2022, ISSN: 0925-2312. DOI: 10.1016/j.neucom.2022.06.111. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231222008426 (visited on 05/10/2024).

[73] S.-i. Amari, "Backpropagation and stochastic gradient descent method", *Neurocomputing*, vol. 5, no. 4, pp. 185–196, Jun. 1993, ISSN: 0925-2312. DOI: 10.1016/0925-2312(93)90006-O. [Online]. Available: https://www.sciencedirect.com/science/article/pii/092523129390006O (visited on 05/10/2024).

[74] H. Schaub, "Rigid Body Kinematics", in *Analytical Mechanics Of Space Systems*, ser. AIAA Education Series, American Institute of Aeronautics and Astronautics, Jan. 2003, pp. 71–126, ISBN: 978-1-56347-563-4. DOI: 10.2514/5.9781600861550.0071.0126. [Online]. Available: https://arc.aiaa.org/doi/10.2514/5.9781600861550.0071.0126 (visited on 05/11/2024).

[75] E. Eade, "Lie Groups for 2D and 3D Transformations", en, May 2017.

[76] D. P. Kingma and J. Ba, *Adam: A Method for Stochastic Optimization*, arXiv:1412.6980 [cs], Jan. 2017. DOI: 10.48550/arXiv.1412.6980. [Online]. Available: http://arxiv.org/abs/1412.6980 (visited on 05/12/2024).

[77] T. Haidegger, L. Kovacs, B. Benyo, and Z. Benyo, "Spatial accuracy of surgical robots", in *2009 5th International Symposium on Applied Computational Intelligence and Informatics*, May 2009, pp. 133–138. DOI: 10.1109/SACI.2009.5136227. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/5136227 (visited on 05/14/2024).

[78] D. M. Kwartowitz, S. D. Herrell, and R. L. Galloway, "Toward image-guided robotic surgery: Determining intrinsic accuracy of the da Vinci robot", en, *International Journal of Computer Assisted Radiology and Surgery*, vol. 1, no. 3, pp. 157–165, Nov. 2006, ISSN: 1861-6429. DOI: 10.1007/s11548-006-0047-3. [Online]. Available: https://doi.org/10.1007/s11548-006-0047-3 (visited on 05/14/2024).

[79] "Quasi-Newton Methods", en, in *Numerical Optimization*, J. Nocedal and S. J. Wright, Eds., New York, NY: Springer, 2006, pp. 135–163, ISBN: 978-0-387-40065-5. DOI: 10.1007/978-0-387-40065-5_6. [Online]. Available: https://doi.org/10.1007/978-0-387-40065-5_6 (visited on 05/14/2024).

[80] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, "A Limited Memory Algorithm for Bound Constrained Optimization", *SIAM Journal on Scientific Computing*,

vol. 16, no. 5, pp. 1190–1208, Sep. 1995, Publisher: Society for Industrial and Applied Mathematics, ISSN: 1064-8275. DOI: 10.1137/0916069. [Online]. Available: https://epubs.siam.org/doi/10.1137/0916069 (visited on 05/14/2024).

[81] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal, "Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization", *ACM Transactions on Mathematical Software*, vol. 23, no. 4, pp. 550–560, Dec. 1997, ISSN: 0098-3500. DOI: 10.1145/279232.279236. [Online]. Available: https://dl.acm.org/doi/10.1145/279232.279236 (visited on 05/14/2024).

[82] B. Siciliano, L. Sciavicco, S. Chiaverini, P. Chiacchio, L. Villani, F. Caccavale, *et al.*, "Jacobian-based algorithms: A bridge between kinematics and control", in *Proceedings of the Special Celebratory Symposium In the honor of Professor Bernie Roth's 70th Birthday*, Citeseer, 2003, pp. 4–35.

[83] M. C. Yip and D. B. Camarillo, "Model-Less Feedback Control of Continuum Manipulators in Constrained Environments", *IEEE Transactions on Robotics*, vol. 30, no. 4, pp. 880–889, Aug. 2014, Conference Name: IEEE Transactions on Robotics, ISSN: 1941-0468. DOI: 10.1109/TRO.2014.2309194.

[84] A. Sarma, T. A. Brumfiel, Y. Chitalia, and J. P. Desai, "Kinematic Modeling and Jacobian-Based Control of the COAST Guidewire Robot", *IEEE Transactions on Medical Robotics and Bionics*, vol. 4, no. 4, pp. 967–975, Nov. 2022, Conference Name: IEEE Transactions on Medical Robotics and Bionics, ISSN: 2576-3202. DOI: 10.1109/TMRB.2022.3216026. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9925615 (visited on 05/14/2024).

[85] M. Da Lio, D. Bortoluzzi, and G. P. Rosati Papini, "Modelling longitudinal vehicle dynamics with neural networks", *Vehicle System Dynamics*, vol. 58, no. 11, pp. 1675–1693, Nov. 2020, Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/00423114.2019.1638947, ISSN: 0042-3114. DOI:

10.1080/00423114.2019.1638947. [Online]. Available: https://doi.org/10.1080/00423114.2019.1638947 (visited on 05/14/2024).

[86] M. Tonutti, D. S. Elson, G.-Z. Yang, A. W. Darzi, and M. H. Sodergren, "The role of technology in minimally invasive surgery: State of the art, recent developments and future directions", *Postgraduate Medical Journal*, vol. 93, no. 1097, pp. 159–167, Mar. 2017, ISSN: 0032-5473. DOI: 10.1136/postgradmedj-2016-134311. [Online]. Available: https://doi.org/10.1136/postgradmedj-2016-134311 (visited on 04/17/2024).

[87] S. U. Bae, W. K. Jeong, and S. K. Baek, "Current status of robotic single-port colonic surgery", en, *The International Journal of Medical Robotics and Computer Assisted Surgery*, vol. 13, no. 1, e1735, 2017, _eprint: https://onlinelibrary.wiley.com/doi/p ISSN: 1478-596X. DOI: 10.1002/rcs.1735. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rcs.1735 (visited on 04/18/2024).

[88] A. L. Orekhov, C. Abah, and N. Simaan, "Snake-Like Robots for Minimally Invasive, Single Port, and Intraluminal Surgeries", in arXiv:1906.04852 [cs], Oct. 2018, pp. 203–243. DOI: 10.1142/9789813232266_0008. [Online]. Available: http://arxiv.org/abs/1906.04852 (visited on 08/27/2023).

[89] J. Reisenauer, M. J. Simoff, M. A. Pritchett, *et al.*, "Ion: Technology and Techniques for Shape-sensing Robotic-assisted Bronchoscopy", *The Annals of Thoracic Surgery*, vol. 113, no. 1, pp. 308–315, Jan. 2022, ISSN: 0003-4975. DOI: 10.1016/j.athoracsur.2021.06.086. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0003497521013837 (visited on 05/14/2024).

[90] D. Bristow, M. Tharayil, and A. Alleyne, "A survey of iterative learning control", *IEEE Control Systems Magazine*, vol. 26, no. 3, pp. 96–114, Jun. 2006, Conference

Name: IEEE Control Systems Magazine, ISSN: 1941-000X. DOI: 10.1109/MCS.2006.1636313. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/1636313 (visited on 05/14/2024).

[91] A. Hiranaka, M. Hwang, S. Lee, *et al.*, "Primitive Skill-Based Robot Learning from Human Evaluative Feedback", in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, ISSN: 2153-0866, Oct. 2023, pp. 7817–7824. DOI: 10.1109/IROS55552.2023.10341912. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10341912 (visited on 05/14/2024).