May 2021

Continuum Robots: Interfacing, Modeling, and Automation

Chase Gilbert Frazelle
Clemson University, cfrazel@g.clemson.edu

Continuum Robots: Interfacing, modeling, and automation

A Dissertation
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy
Electrical Engineering

by
Chase Gilbert Frazelle
May 2021

Abstract

The work in this dissertation explores interfacing, modeling, and automation of continuum manipulators. In particular, we work to build a bridge between continuum robots and new areas of research through haptic interfacing, and explore how established and popular methods in the broader robotic community apply to continuum robots and their applications.

First, we describe a novel haptic interface designed specifically for the teleoperation of extensible continuum manipulators. The proposed device is based off of, and extends to the haptic domain, a kinematically similar input device for continuum manipulators called the MiniOct. The work in this dissertation describes the physical design of the new device, the method of creating impedance-type haptic feedback to users, and some of the requirements for implementing this device in a bilateral teleoperation scheme for continuum robots. We report a series of experiments to validate the operation of the system, including simulated and real-time conditions. The experimental results show that a user can identify the direction of planar obstacles from the feedback for both virtual and physical environments. Further, we discuss the challenges for providing feedback to an operator about the state of a teleoperated continuum manipulator.

Next, we explore methods for approximating continuum robot dynamics through rigid bodies connected together via compliant joints. The nature of the joints are designed to capture the material properties of continuum robot backbones, while the rigid links approximate the inertia and shape of the actual continuum robot's body. In evaluating the models, we formulate a state estimator based on energy minimization methods and compare the estimated state of a physical robot to ground truth values provided by external sensors. Finally we evaluate the model through a series of experiments testing the step response of the model and an experiment to observe the model throughout dynamic motion. The results show a good match between predicted motion of the simulated robot and the physical robot's state estimator.

Continuum robots have long held a great potential for applications in inspection of remote, hard-to-reach environments. In future environments such as the Deep Space Gateway, remote deployment of robotic solutions will require a high level of autonomy due to communication delays and unavailability of human crews. Further work in this dissertation explores the application of policy optimization methods through Actor-Critic gradient descent in order to optimize a continuum manipulator's search method for an unknown object. We show that we can deploy a continuum robot without prior knowledge of a goal object location and converge to a policy that finds the goal and can be reused in future deployments. We also show that the method can be quickly extended for multiple Degrees-of-Freedom and that we can restrict the policy with virtual and physical obstacles. These two scenarios are highlighted using a simulation environment with 15 and 135 unique states, respectively.

Finally, given the challenging nature of continuum robot modeling in comparison with traditional rigid-link robots, we consider the Kinematic-Model-Free (KMF) robot control method, previously shown to be extremely effective in permitting a rigid-link robot to learn approximations of local kinematics and dynamics ("kinodynamics") at various points in the robot's task space. These approximations enable the robot to follow various trajectories and even adapt to changes in the robot's kinematic structure. In a final work in this dissertation, we present the adaptation of the KMF method to a three-section, nine degrees-of-freedom continuum manipulator for both planar and spatial task spaces. Using only an external 3D camera, we show that the KMF method allows the continuum robot to converge to various desired set points in the robot's task space, avoiding the complexities inherent in solving this problem using traditional inverse kinematics. The success of the method shows that a continuum robot can "learn" enough information from an external camera to reach and track desired points and trajectories, without needing knowledge of exact shape or position of the robot.

Dedication

To my sister, for being an exemplar of pursuing one's dreams and to my parents and family for their endless support and encouragement.

Acknowledgments

In reflecting on the journey I have experienced in pursuing the work in this dissertation, and my collegiate career as a whole, there is a seemingly innumerable number of people to whom I feel I owe thanks. I would like to take this moment to express my gratitude to some of the many individuals and groups that made this all possible.

First, to my advisor and committee chair, Dr. Ian Walker, for all of the support and encouragement throughout my time in the lab. From giving me my first research opportunity as an undergraduate senior to supporting my various project ideas and travels to present those ideas, Dr. Walker's seemingly endless supply of patience and insight has been invaluable and irreplaceable in my time in Clemson and my development as a researcher. I also owe a huge debt of gratitude to Dr. Apoorva Kapadia for both his role in getting me started in the lab and learning the ropes of research and writing, but also for the encouragement and reassurances throughout the years that I'm on the right path.

I also wish to thank my remaining committee members, Dr. Richard Groff, Dr. Adam Hoover, and Dr. Ioannis Karamouzas, for their role in refining this work and providing feedback on my research. I would also thank them for the knowledge and wisdom they imparted to me through various graduate courses, hallway conversations, and collaborations. Finally, I would like to thank them for being equally excellent advisors to their own students, many of whom I had the fortune to befriend and exchange knowledge that we gained through our respective works.

One of the more unique experiences of my graduate studies came from my time at Johnson Space Center through the NSTRF program. I greatly appreciate and say thanks for all of the mentoring and advice I received from Jonathan Rogers and the many members of ER4 during my time at JSC and the unique opportunities afforded to me, as well as the advice and collaboration that has extended beyond those periods. I also am forever thankful to the NASA Space Technology Fellowship program which funded the majority of my graduate studies through the NSTRF17 fellowship, contract 80NSSC17K0173.

I have had the fortune of making a great number of friends and esteemed peers during my graduate studies. I feel I could write an entirely new document of similar length to this one just reflecting on the friends I've made and the fun and stressful times we've shared and still not do justice to everyone I call friend. While I cannot possibly name everyone fitting this title, I do wish to give a special thanks to Ryan Scott, Irfan Kil, Michael Wooten, and Aaron Shepard for being an ever present source of stress relief and camaraderie since the very beginning of this journey, not to mention being inspiring and unflappable individuals when facing adversity in all walks of life.

I would also take the time to thank every individual that has passed through the Continuum Robotics Lab (CuRL) in my time here, either as student under Dr. Walker, as collaborator from another group, or just a visitor with a curiosity for robotics. You have all afforded me forever valuable opportunities to hear new ideas, challenge my understanding of the world, and form friendships and professional relationships that are irreplaceable. I also appreciate the patience I have been shown when I have started off down rabbit holes that do not always answer the questions of the problem at hand.

Two final groups I wish to thank for their unfailing support are my family and the many friends I made during my undergraduate studies here at Clemson, which at this point are nearly indistinguishable from family. To my family, thank you not only for your support during this journey, but also for always encouraging me in all walks of life. To my many friends, especially the Berkeley Boys and our many close friends, thank you for being such amazing people and living your lives to the absolute fullest, it is truly an unending source of inspiration and strength to have such amazing and truly good people in my life. Go Tigers!

Finally, I owe an immeasurable gratitude to Di Nguyen for being by my side through all of the hurdles of graduate school. Her support made the periods of time I had to spend away from Clemson possible and her support and presence during the many late nights, when we both felt the stress of upcoming deadlines, made the fruits of those efforts possible. I also cannot express enough gratitude for her patience during the countless times that our conversations have resulted in my brain somehow turning to research and thinking up new ideas to test in the lab. I look forward to continuing to support you in kind as you finish your doctorate and we move on together to the next chapter of our lives.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Continuum Robots

Continuum, or continuous backbone, robots [1] manifest as a cross between traditional rigid-link robots (given their ability to maneuver in, sense, and interact with their environment) and applied materials and compliant mechanism design, as the choice of material and actuation modality can prove to be as influential to the function of a continuum robot as the arrangement of joints is to a traditional rigid-link robot. The "continuum" element of this class of robots draws many parallels to the biological world, ranging from vertebrates with continuum appendages such elephant trunks and prehensile tails to invertebrates such as cephalopods whose entire body is made up of compliant, soft material capable of extreme dexterity and manipulation [2–5]. Continuum robots also draw inspiration from plants, with works exploring plant locomotion [6, 7], nutrient and environment driven growth [8, 9], and mimicking of unique plant structures to aid in task execution [10, 11]. The diversity of biological inspiration for this class of robots also creates one of the core subclass divisions, that of extensible and non-extensible continuum robots. As the name suggests, extensible continuum manipulators, such as in [12, 13], have the ability to extend and retract their length as desired, where non-extensible continuum robots have a fixed length from the time of construction, though the effective length of some continuum robots built from inextensible materials can be adjusted using concentric, constraining tubes [14, 15].

Along with drawing inspiration from a different biological source than the standard anthropomorphic robots, continuum robots have a number of function related characteristics that distinguish them from traditional rigid-link robots. One such characteristic is their ability to change shape (bend) at any point along their structure, allowing them to manifest a theoretically infinite number of Degrees-of-Freedom (DoF) and exhibit hyper-redundancy within their work-space [16]. This ability allows continuum robots to conform to obstacles and perform maneuvers such as whole-arm manipulation where they use a portion of their length to manipulate an object directly instead of just the end-effector or tool-end [5, 17, 18]. The hyper-redundancy also presents a high dimension of self-motion [19], in which the robot can alter its backbone shape without changing the location of the end-effector, a useful task for conforming to changes in the environment while performing a task.

Another distinguishing characteristic of continuum robots is fact that the source of actuation is generally located away from the core structure of the manipulator [20], transferring actuation through a variety of actuation types, including tendons, pneumatic or hydraulic pressure, or synthetic muscles that rely on external power sources to drive local locomotion, among others. This absence of the actuation source in the length of the manipulator makes continuum robots very scalable, with solutions ranging from robots that are a couple centimeters in length [21] for non-invasive surgery to multiple meters [22] for inspection and manipulation in hard-to-reach environments.

1.1.0.1 Applications for Continuum Robotics

The scalability and remote actuation aspects of continuum manipulators together bring a plethora of application spaces to these systems. One popular field for application is the use of continuum robots in the medical field [21], where we can find several examples of medical procedures [23–26] that benefit from small tool arms that are well suited for the curving paths of various structures in human anatomy. The applicability of continuum robots in medicine draws in part on the ability of continuum robots to perform follow-the-leader maneuvers [27, 28], ideal for small insertion points.

Also benefiting from the follow-the-leader aspect of continuum robots is the exploration and inspection of hard-to-reach environments [29–32], both structured and unstructured. Combined with the natural compliance of these robots, this also lends to continuum robots being ideal for cluttered environments [33, 34] that could cause rigid-link robots to become trapped. In particular, the introduction of growing robots, as related in [9, 35] and which share many similarities with traditional continuum robots, showcases the ability of soft growing structures to explore highly restricted environments and take advantage of the soft structure to conform and react to the environment, giving even greater access restricted, traditionally non-navigable spaces.

One application area of particular interest in this work is the use of continuum robots in assisting Space exploration and extending mission life. In particular, their suitability for inspection and movement in cluttered environments has shown potential for application in Space applications [36, 37] aboard the International Space Station, and could similarly be useful the future Deep Space Gateway [31]. Other application spaces for continuum robots include underwater exploration [38], search and rescue [39], and hazardous waste material handling [10, 40, 41].

1.1.1 Modeling and Control Literature

1.1.1.1 Continuum Robot Kinematics

Kinematic modeling of continuum robots has been the subject of extensive research, producing everything from practical but approximate models to complex but geometrically exact solutions. A modal approach was presented in [16], where the authors describe the robot shape using easily formulable modal functions specified as the product of a Bessel function with sines and cosines. The work presented in [42] employed a similar approach but instead used wavelet decomposition. The discrepancy between the shapes achievable by the finite number of proposed model curves and those of the continuum robot's backbone curve limited the applicability of this approach. Perhaps the most commonly adopted approach to modeling constant curvature continuum robot kinematics to date is that of [43], which introduced models for forward and inverse kinematics. The result, by relating backbone shape to actuator variables, extends that obtained in [44] which used virtual rigid-link kinematics and conventional D-H parameters to relate backbone shape to task coordinates [20, 45–47], which in turn can be shown to produce the same results as the approach taken in [48–50] which treats the robot backbone as a curve in space, and utilizes Frenet-Serret frame floating along the curve to characterize it. However, all of these models suffer from numerical (algorithmic, i.e. numerical instabilities at configurations where there is no corresponding physical limitation) singularities and demand special numerical treatment when close to straight (zero curvature) section configurations. In response to the above limitations, an alternative stable yet computationally intensive approach was presented by Godage et al. [51, 52] using mode shape functions, wherein the configuration space variables are approximated using multivariate Taylor series.

This approach has until recently been established as the most numerically stable currently in the literature. However, recently a new singularity-free analytical approach for modeling constant curvature kinematics has been introduced [53] and further detailed in [54], which derives inspiration from screw theory.

In other work in continuum kinematic modeling, the use of exponential coordinates to define kinematics has also appeared in the literature [46, 55, 56], and is shown to lead to essentially the same results as other constant curvature kinematic models. When assuming non-constant curvature, the theory of Cosserat Rods has been particularly useful in deriving geometrically exact kinematic models which account for external and gravitational loading, but are harder to implement and complex in nature [57–59].

1.1.1.2 Continuum Robot Dynamics

Considering the diversity of methods used to model continuum robot kinematics, it is no surprise that continuum robot dynamics has also received notable attention, especially of late as computers become more powerful and able to better aid in calculating the complex dynamics of these systems. Early continuum dynamics characterized continuum manipulators as infinite serial chains projected onto established robotic structures [60], where later expansions of this idea characterized the chains using Frenet-Serret frames [49] and added extensibility [61].

More recent dynamics work for continuum robots has approached the problem from the material and structure view, using continuum mechanics to relate continuum robots to beams undergoing large deflection [62] or a combination of beam mechanics and the Euler-Lagrange method [63]. Similarly, a popular foundation theory for continuum dynamics of late has been Cosserat rod theory [64–66], which provide for large deflections and can provide geometrically exact solutions (though not fit for extensible continuum robots).

Other dynamic models related to continuum robots have utilized approximations coupled with traditional dynamics formulations. In [67] and [68], the models used simplified, small mechanical elements combined with the Newton-Euler method to produce approximate simulations of continuum bodies without creating closed-form equations that could be used for control formulation. Conversely, [69] uses the Euler-Lagrange formulation to produce closed form dynamics for a single section, non-extensible continuum robot based on ideal, constant curvature kinematics. In [70], the Euler-Lagrange method is also used to produce a planar dynamic model for a multi-section continuum robot. Finally, the principal of virtual power is used in [71] to derive dynamics of a continuum rod with high accuracy, but it is not clear how usable this model is in real-time and is not accompanied with inverse mechanics model.

Among these various models the complexity of the inherent mechanics, especially when applied to multiple sections, often proves to be unwieldy without making numerous simplifying assumptions. Indeed, our own recent approximate models [72, 73] are still computationally intensive even after removing the complexity of continuum mechanics and settling for approximate rigid-link models. Coupled with the effects of gravity, achieving motion along a desired path is dependent on reliable modeling and sufficient feedback. With respect to continuum robots in practical applications, the states of such solutions are often inherently unobservable, under-actuated, or both, without the implementation of computationally expensive sensor arrays or a suite of external cameras to extract full robot shape while avoiding occlusion. Given these challenges, it is no surprise that the development of continuum dynamic models and control using those models remains active and open areas of interest and research.

1.1.2 Literature on Continuum Interfacing

Teleoperation, the idea of remotely operating a deployed system through some form of human to machine interface, is an area of robotics that is fundamental for the development and use of robotic systems in remote environments. The ability to interact with a remote environment through a robotic system remains a major advantage when dealing with hazardous situations and unstable surroundings [74]. Despite extensive research in the topic of teleoperation [75], the subject is still open to innovation, especially when concerning less conventional continuum robots.

In the literature, there have been various investigations into the teleoperation of extensible continuum manipulators [76–78]. These particular works demonstrated useful capabilities in teleoperating continuum robots but each had shortcomings in various aspects, especially concerning intuitive interfacing. However, these works also relied on complex mappings and control schemes in order to bridge a mismatch in kinematics and degrees-of-freedom which do not necessarily present clearly to users. There have been three recent works particularly focused at developing kinematically similar interfaces for this class of robots. In [79], a 3-section, 9 Degree-of-Freedom (DoF) continuum interface was introduced specifically for the teleoperation of continuum manipulators. The high DoF allowed for 1:1 mappings between the device, termed the MiniOct, and 3-section extensible continuum robots. The MiniOct was also shown to be a versatile tool that could serve as an interface for a variety of continuum robots that varied in both scale and actuation type. The continuum interface presented in [80] is a 4 DoF interface capable of mapping to non-extensible 4 DoF continuum robots. The works shows an intuitive correlation between the shape of the master device and the slave device. In [81], a continuum interface geared towards the teleoperation of growing-vine robots is introduced. In addition to the soft nature of the interface, a unique feature of this device is the use of IMUs to measure the curvature of the device, independent of orientation. The interface itself has 2 DoF, with an additional linear potentiometer that controls the growth rate of vine robots. While unidirectional teleoperation has clearly been explored for continuum manipulators, there appears to be no literature on the design or use of haptic interfaces for teleoperating extensible continuum manipulators.

A natural step forward for the teleoperation of continuum robots would be the introduction of bilateral teleoperative modes, one example of which would be to provide information from the deployed secondary system to the user through haptic feedback. Haptic feedback is the method of providing sensory feedback through touch [82]. Haptic feedback has been repeatedly used in teleoperated surgery in order to provide surgeons with additional insight about tissues and patient vitals [83], [84]. Other work has studied human perception and lessons that can be learned from psychology in order to maximize the information that can be perceived through auxiliary senses [85]. While many haptic interfaces exist and have found practical applications [86], these solutions exhibit a mismatch in DoF and kinematics to multi-section extensible continuum manipulators.

1.1.3 Simulating Continuum Interaction with the World

Given the complexity of continuum robot modeling, especially any attempts to capture the nature of the compliant materials comprising these devices, it is not surprising that the options for standardized simulators is limited or non-existent. In designing systems for future deployment and integration, the ability to plan and simulate deployment scenarios can be crucial to mission success and save both time and resources prior to implementation. There are few examples of software simulations of continuum robots that present both visual simulations and incorporate physical properties of these systems.

In [87], a virtual user interface is created to drive the OctArm continuum robot and provide a visual approximation of the ideal robot shape based on continuum kinematics. While the interface is customizable and able to potentially connect to various continuum systems, it does not relay information about the actual robot or simulate the physics of the real-world. Likewise, various simulation models have been presented across numerous works that convey continuum robot configuration throughout execution of a dynamic response, but are largely one-off simulations [88, 89] or are merely visible representations of previously executed/solved dynamic model simulations (i.e. the visual elements are only defined for the duration of the experiment).

Not reported directly in the literature, but relevant to the simulation work introduced herein, are examples of continuum robot simulation models that fill some of the gaps in literature. For example, the work in [90] creates and presents a continuum library that establishes a visual model of a continuum robot using with multiple sections based on the Jones kinematic model. This model is notable also because of its compatibility with the popular ROS architecture. Where this model stops short is in providing a viable model for physics simulation. Conversely, the package in [91] introduces a custom physics simulation of continuum robots that includes a basic visualizer and customizable robot parameters. However, the package does not include simulations with other models or a direct way to connect with other platforms or tools. Similarly, [92] introduces a custom physics simulation for testing and modeling continuum robots in Matlab based on beam mechanics, but without the ability to simulate complex motions and interactions with the environment and other systems.

In considering a general purpose package for simulating continuum robots in various environments, communicating with off-the-shelf tools and systems, the aforementioned ROS architecture presents a widely utilized framework that provides for relatively simple system integration. There are many systems already supported with custom ROS packages such as KUKA, Kinova, and ABS, as well as more custom and unique systems such as Robonaut [93]. The work in this dissertation will explore integrating continuum robots into the ROS architecture in a generalized form, focusing on the physics simulator Gazebo and the visual tool RVIZ. Basic tools that highlight the functionality of the package will also be discussed, before some use cases of the package are presented.

1.1.4 Motion planning and automation Literature

There have been a number of explorations into motion planning methods for continuum robots, which often can help avoid the challenges that redundancy and complex configuration spaces bring to solving problems like inverse kinematics. Configuration space exploration and exploration through sample-based methods such as Rapidly-exploring Random Tree (RRT) methods have been deployed for maneuvering continuum manipulators through various task spaces [94–98], even adaptively updating the knowledge and trajectory during execution [34]. Once configuration space driven methods are implemented to locate points of interest in the continuum manipulator's task space, kinematics driven methods such as visual servoing [33] can be used to refine the manipulator's interaction with the environment.

Reinforcement learning (RL) provides an attractive alternative where a robot/agent learns to take actions that maximize its cumulative reward through interactions with the environment. Many early success stories exist, from training robots to compete in RoboCup competitions to enabling robots to acquire advanced manipulation skills [99–102]. More recently with the rise of deep learning, impressive results have been obtained on physical articulated robots for a wide range of motor and manipulation tasks [103–106]. In the continuum robotics domain, a number of works have explored the use of reinforcement learning to improve motion planning methods and improve upon various control schemes [107, 108]. Most relevant to this work is the use of a Soft Actor-Critic (SAC) method in [89] to optimize a continuum manipulators policy for reaching a point in space with the robot's end-effector. In that work, the authors employ a Random Network Distillation (RND) method to train a series of neural networks and then use the SAC algorithm to maximize the return of the policy designed to capture a known object in space. They report a dependency on sparse reward and a need for the RND method in order to promote adequate exploration. Of these works, many rely on a need of *a priori* knowledge of the environment or of a specific goal state.

Herein, we investigate the feasibility of using a RL framework to train policies on continuum manipulators. In particular, this dissertation explores the application of reinforcement learning for continuum manipulators with the aim of automating continuum robots being used for inspection. An actor-critic policy gradient method is applied with the purpose of locating a goal object and creating a global policy that determines how the robot behaves when deployed with the task of observing various points of interest in its task space. The method can be expanded for encompassing extra DoF, as well as be used to develop policies simultaneously for different points of interest within the robot's workspace.

Rigid-link robots are, as the name suggests, traditionally comprised of rigid bodies connected through a finite series of joints. While this makes for tractable dynamics and permits reasonable controllability, this form of structure does not lend itself well to cluttered environments or scenarios in which collisions, even minor, can result in damage to the robot and environment. Conversely, continuum robots, which can bend at any point along their continuous backbones, are designed to be compliant, and in cluttered environments are able to experience contacts without causing damage. The compliant nature of continuum robots makes them ideal for inspection and sensing in restricted environments, such as cluttered cargo areas and hard-to-reach areas of interest. Unfortunately, this increase in compliance also carries an increase in the complexity of the robot's dynamics and inverse kinematics, especially in the case of a redundant continuum manipulator. In this paper, we propose the use of a model learning algorithm to replace the use of dynamics and kinematics in guiding a continuum robot's end-effector through space.

Continuum manipulators manifest a theoretically infinite number of Degrees-of-Freedom (DoF), distinctly different from even hyper-redundant rigid-link robots, such as snake robots [16]. There is by now a fairly extensive literature on continuum robot kinematics, many of which constrain these infinite DoF to ideal assumptions about the robot and its environment. Given these ideal conditions, the constant curvature based kinematic models [43, 53] have proven effective for approximating the shape of continuum systems and the location of end-effectors in open space. Further works have expanded into modeling non-constant curvature bending of continuum robots that are subject to internal and external loads [16, 109, 110], or collisions with the environment [57, 111]. These more realistic models, while better at approximating shape and predicting output under load, require significantly more information about the continuum system and do not lend themselves well to invertibility or the added complexity of implementation on multi-section continuum manipulators. Neither of these classes of kinematic models have proven trivial for the execution of task space path planning and following.

In considering the complexity of kinematic and dynamic models for continuum robots, it is easy to see why the relatively few works that explore motion planning with continuum manipulators have relied on simplification of these models or reduction in the number of DoF. This simplification allows implementation of some of the popular motion planning methods used for rigid-link robots, such as RRT [96, 97] and reinforcement learning methods [107, 108].

1.2 Dissertation Preview

This section previews the content and contributions of the remaining chapters, which will explore various aspects of continuum robots and robotics research.

In Chapter 2, Section 2.1, we briefly review continuum kinematics as they will be used to relate the work presented throughout this dissertation. In Section 2.2, we introduce the first haptic continuum interface, as reported in [112], designed for the bilateral teleoperation of extensible continuum robots. The novelty of this work expands on a previous novel interface for continuum robots that is kinematically similar to their intended secondary devices. The work introduces for the first time an ability to achieve bilateral teleoperation of continuum robots in kinematically identical interfacing and the use of haptics to relate error between actual and desired states in deployed continuum systems. Section 2.3 then explores continuum robot simulation using physics simulators and the applicability of ROS to continuum robots, a useful tool for the practical testing and use of the haptic continuum interface. Previous simulation of continuum robots has largely used custom physics simulations, not universally compatible packages, or in the case of previous works with ROS, has ignored physics altogether and only simulated ideal kinematics. The work in this section therefore presents a simulation model that is compatible with simulators in industry and with ROS, to capture a functional approximation of continuum robots without the need for custom physics engines.

In Chapter 3, we introduce a piece-wise constant curvature dynamic model for continuum manipulators based on a rigid-link approximation of continuum robot sections. The novelty of this model is to provide real-time dynamics approximation of continuum robots while also capturing an approximation of the compliance, and subsequent manifestation of non-constant curvature, of continuum robots. Section 3.3 presents a method for approximating the piece-wise constant curvature of the model with the physical robot without the need for measuring the full state of the robot. Finally, Section 3.4 compares the response of the model to that of a real continuum robot to various stimuli, followed by discussion and suggestions for model expansion.

Chapter 4 explores two novel contributions investigating the automation of continuum robot tasks. The first work, in Section 4.1 and first reported in [113], applies reinforcement learning methods to the task of searching a workspace for points of interest. The method improves upon previous state-of-the-art visual servoing for continuum robots by creating a global policy that functions when the robot is in aliased states. The second automation related work, presented in Section 4.2 and first reported in [114], explores using a model-free control method to automate the motion of a continuum robot's end-effector about its workspace while knowing no information about the structure or state of the robot itself. The work enables convergence to points in the workspace with error equivalent to the state-of-the-art in continuum modeling and control, while greatly reducing the complexity of information known about the robot and the amount of calculations necessary to choose the appropriate inputs to drive the system.

Final conclusions and predictions for the future direction of these collective works is presented in Chapter 5.

Chapter 2

Interfacing with the Continuum World

In this chapter, we give a brief review and discussion of the kinematic models that appear throughout this dissertation and briefly review the intuitions different parameterizations of continuum kinematics can provide. Following this, we introduce a novel continuum interface with the capability of providing haptic feedback to users and enable bilateral teleoperation between the interface and a deployed secondary system. Later in the chapter we introduce a ROS compatible continuum robot simulation model that is useful for simulating continuum robot deployment and provides a path towards system integration and communication with the broader robotic community. Finally, we discuss considerations for both the haptic interface and ROS package.

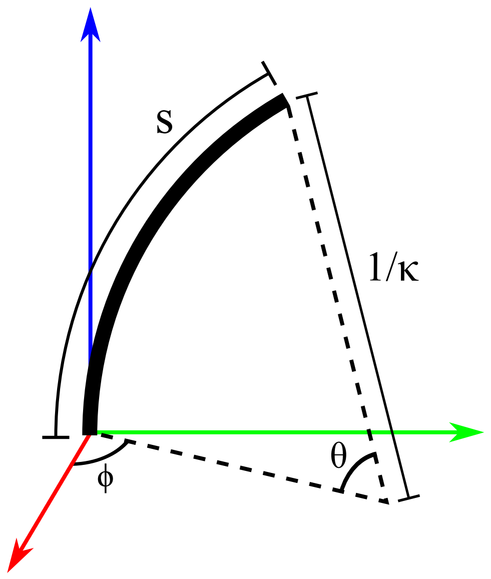
2.1 Kinematic Preliminaries

This section serves to review continuum robot kinematic models as later referenced throughout this work. Two models, found in the literature and reproduced here, offer equivalent mathematical descriptions of continuum robots but present unique physical interpretations and intuitions for continuum robot motion. Both of the following models are presented as constant curvature kinematic models, in which continuum sections are assumed to bend at a constant rate along their backbone, forming perfect circular segments.

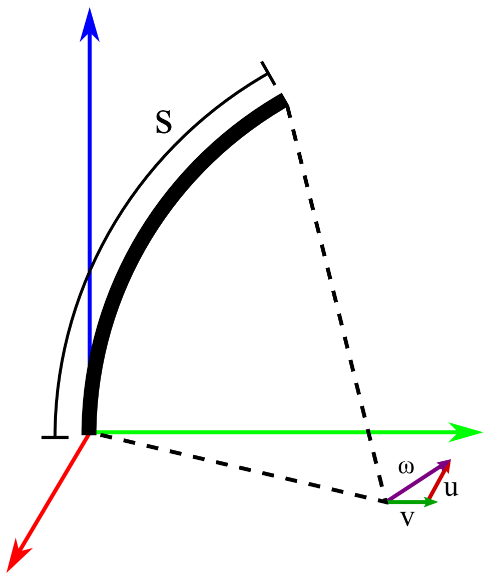
The first model, introduced in [43] and henceforth denoted in this work as the “Jones” model, describes a 3 DoF continuum robot section via a section’s backbone arc-length (s), curvature (κ), and direction of bending relative to the base frame (ϕ), as visually presented in Figure 2.1a. This model originally draws inspiration from rigid-link robot kinematics, whereby the orientation of the plane of bending is related by the rotation of a revolute joint at the base of the section (ϕ) and both curvature and arc-length are related through a RPR (Revolute-Prismatic-Revolute) joint configuration. The two revolute joints of the virtual RPR configuration are constrained to represent the rotation between the base and end-point of the section and the prismatic joint is constrained to match the chord length of the arc formed by the physical continuum section.

The second model heavily referenced throughout this thesis is first introduced in [53] and later expanded in [54], and in this thesis is denoted as the ”Allen” model. Similar to the Jones model, the Allen model uses three parameters to describe a single continuum section in three dimensional space, represented visually in Figure 2.1b. The first two parameters, denoted u and v, are part of an angular velocity vector $\omega(t) = [u(t), v(t), 0]^T$ that describes the movement to the section’s end frame relative to the section’s base frame. The value u can be inferred to be the rotation around the X-axis in the base frame and v describes the rotation about the Y-axis. The rotation component about the Z-axis is always zero. These values equate to a total rotation $\theta = \sqrt{u^2 + v^2}$ about the unit vector $\omega/\|\omega\|$. The third parameter is the section length, or arc-length, similarly defined as in the Jones model but originally denoted h. For simplicity, we continue to use the variable s to denote the central length of the section. The Allen model draws inspiration from Screw theory [115], in which the configuration and motion of the continuum arm can be described through the rotation of a point about a defined axis in space. The Allen parameterization is unique from the Jones model in that it provides for stability and continuity of the kinematic parameters when a section is straight or exhibits zero curvature, a discontinuity that plagues most other ”constant-curvature” kinematic models.

While the two kinematic models present different parameterizations of a continuum section, they are geometrically identical, allowing one to convert between the models at will. Both models also provide for methods of calculating the kinematic parameters through the measurement of tendons placed at intervals of 120° around the outside of a continuum section, as illustrated in Figure 2.2. These equations, and the equations that allow one to convert between the two kinematic models, are presented in Table 2.1. When comparing the equations side-by-side, it can appear as though the two



(a) Continuum section described through arc-length (s), curvature (κ), and direction of bending (ϕ)



(b) Continuum section described through arc-length (s), and orthogonal rotation elements (u and v)

Figure 2.1: A comparison of kinematic parameters describing the same geometric configuration of a continuum robot section with constant curvature.

models do not align perfectly when observing the relating equation for ϕ, but this is explained by the fact that the initial reporting of the Jones model assumes tendon 1 is located along the Y-axis of the base plane as opposed to along the X-axis as in the Allen model and depicted in Figure 2.2. This creates an artificial 90° phase shift between the two models, but is easily remedied by introducing the proper shift in either model when converting.

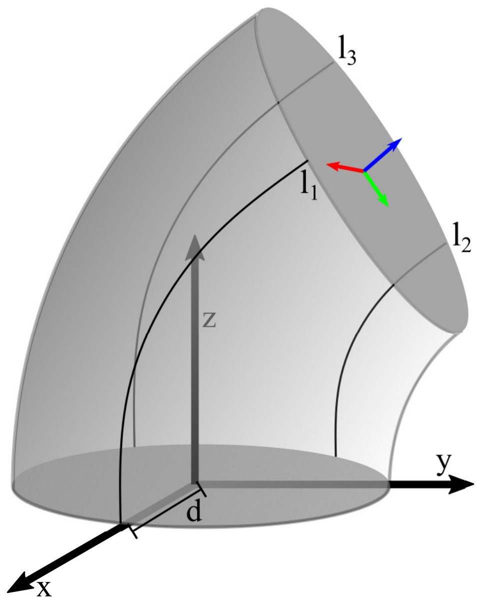


Figure 2.2: Demonstration of tendon placement at 120deg intervals around a continuum section, with the first tendon placed along the X-axis. The value d is the radial distance of the tendons from the central backbone of the continuum section.

Jones Model	Allen Model
$^1 s = \frac{(l_1+l_2+l_3)nd}{\sqrt{l_1^2+l_2^2+l_3^2-l_1l_2-l_2l_3-l_1l_3}} \cdot \sin\left(\frac{\sqrt{l^2{}_1+l^2{}_2+l^2{}_3-l_1l_2-l_2l_3-l_1l_3}}{3nd}\right)$	$u = \frac{l_2-l_3}{d\sqrt{3}}$ (2.1.1)
$\kappa = 2\frac{\sqrt{l_1^2+l_2^2+l_3^2-l_1l_2-l_2l_3-l_1l_3}}{d(l_1+l_2+l_3)}$	$v = \frac{s(t)-l_1}{d}$ (2.1.2)
$\phi = \tan^{-1}\left(\frac{\sqrt{3}}{3}\frac{l_3+l_2-2l_1}{l_2-l_3}\right)$	$s = \frac{l_1+l_2+l_3}{3}$ (2.1.3)
Relating Models	
$\theta = \sqrt{u^2+v^2} = s\kappa$	
$\phi = \tan^{-1}\left(\frac{-u}{v}\right)$	
$[u, v] = [-\theta\sin\phi, \theta\cos\phi]$	

Table 2.1: Comparison of Kinematic Model Parameters

The relative transformation matrix associated with each model is given below. It is left as

[1]One can also use equation 2.1.3 to approximate arc-length for the Jones model.

an exercise for the reader to note that the relating equations in Table 2.1 can be used to convert from each transformation matrix to the other and back without loss of information when accounting for the phase shift of ϕ. The transformation matrix for the Jones kinematic model is:

$$H(s,\kappa,\phi) = \begin{bmatrix} c^2(\phi)(c(s\kappa)-1)+1 & s(\phi)c(\phi)(c(s\kappa)-1) & -c(\phi)s(s\kappa) & \frac{c(\phi)(1-c(s\kappa))}{\kappa} \\ s(\phi)c(\phi)(c(s\kappa)-1) & c^2(\phi)(1-c(s\kappa))+c(s\kappa) & -s(\phi)s(s\kappa) & \frac{s(\phi)(1-c(s\kappa))}{\kappa} \\ c(\phi)s(s\kappa) & s(\phi)s(s\kappa) & c(s\kappa) & \frac{s(s\kappa)}{\kappa} \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (2.2)$$

where $c(\cdot)$ and $s(\cdot)$ represent the functions $\cos(\cdot)$ and $\sin(\cdot)$, respectively, and $c^2(\cdot)$ denotes $\cos^2(\cdot)$. The transformation matrix for the Allen model is given as:

$$H(u,v,s) = \begin{bmatrix} \gamma v^2+1 & -\gamma uv & \zeta v & -\gamma sv \\ -\gamma uv & \gamma u^2+1 & -\zeta u & \gamma su \\ -\zeta v & \zeta u & \cos(\theta) & \zeta s \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (2.3)$$

where $\zeta(\theta) = \sin(\theta)/\theta$ and $\gamma(\theta) = (\cos(\theta)-1)/\theta^2$. As noted in Table 2.1, the value θ in the case of the Allen model is defined to be $\theta = \sqrt{u^2+v^2}$. It is noted in the original reporting of the Allen model that the functions $\zeta(\theta)$ and $\gamma(\theta)$ can be shown to still be defined when θ is zero using the Taylor series expansions of sine and cosine.

2.2 Haptics in the World of Continuum Interfaces: The HaptOct

In this section we introduce the HaptOct, a haptic continuum interface capable of providing a one-to-one kinematic mapping between the user interface and a 9 DoF extensible continuum manipulator, the first known haptic display targeted directly at continuum manipulators. The core of this design is based on a previously reported interface called the MiniOct, but with the added ability to oppose user input based on the state of a deployed secondary system located in a remote environment.

The novelty of this work is found in using error between the primary continuum interface and the deployed secondary system to convey somatosensory input to the user, in addition to the visual information conveyed by the kinematically similar interface. The use of such systems is widely available in rigid-link robot teleoperation, where it is a relatively simple matter to calculate motor torques on the secondary system and apply similar torques to an interface that is identical or similar to the remote robot. With respect to continuum robots, it is not a trivial matter to calculate exerted torques, interactive forces from the environment, or even sense internal deviation from an ideal state without relatively large amounts of information and detailed, computationally expensive models.

The following subsections will explore the construction of the HaptOct, the underlying process for generating the virtual impedance display, and experiments using virtual environments and using a physical remote robot to test the efficacy of the system.

2.2.1 The MiniOct: A Kinematically Similar Interface

In [79], a novel continuum robot interface was introduced in the form of the MiniOct, a 9 DoF interface capable of the same motions and configurations as a 9 DoF extensible continuum robot. The device, pictured in Figure 2.3a, consisted of a passive continuum interface (labeled as region 'A' in the image) and a tendon storage and measurement portion (labeled region 'B'). The continuum interface comprised of three continuum sections, each of which a user could manually extend and bend in any direction, independent of the other sections. Each individual section is comprised of a parallel set of extension springs and steel cables that allow an operator to extend and curve each section into a desired configuration. Plastic spacers (9 per section) are distributed evenly along the length of a section and keep the springs and cables uniformly, radially, spaced.

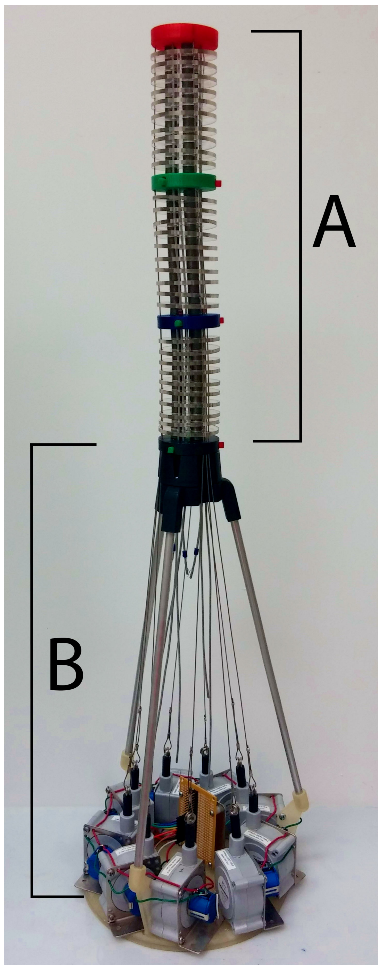
The spacers cause any imposed curvature to divide evenly along the length of the section. A spring loaded tab connected to each structural steel cable and housed within the section divider at the base of each section holds the section shape until the operator changes the shape by pressing one of the tabs, releasing the cable, and then extending or shortening the length of the cable. When a tab is released by the operator, the shape is held by friction. An illustration of the operation of this mechanism is given in Figure 2.4 The continuum interface is supported via 3 aluminum rods. These rods can be extended or shortened to allow for the storage of longer structural cables.

The MiniOct detected shape of the continuum section through a passive series of spring loaded string potentiometers. The changes in tendon length measured by the string pots were converted to kinematic values using constant curvature kinematics such as found in Section 2.1. The continuum sections of the input device measured 10cm in length at rest and were capable of extending approximately 100% of the default length. The overall device was 71.59 cm in height (including the cable measurement region), with the continuum section able to extend an additional 30 cm when at maximum extension for all 3 sections. The range of bending angle θ was approximately π, or 180°, forming a perfect semi-circle. Demonstrations of this device teleoperating various continuum robots displayed good shape matching between the interface and the secondary system in open-loop schemes. The shape match could also be improved by a simple closed-loop controller on the secondary system.

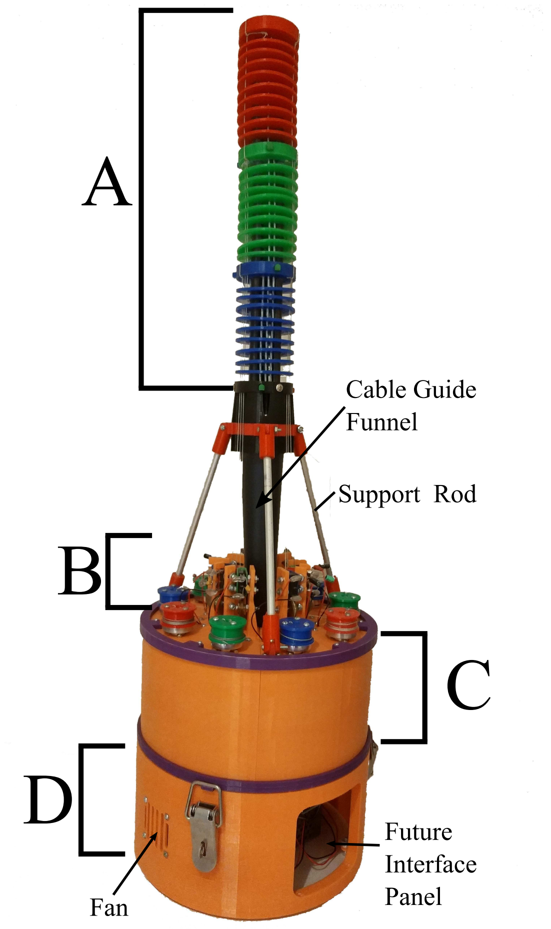
2.2.2 HaptOct Continuum Interface (A)

The continuum section of this device–the part handled by the user–was selected to be a continuation of the continuum interface described for the MiniOct. The MiniOct's design was chosen as the basis for this new device because of the kinematic and visual similarity between the master device and potential secondary systems. Additional design details of the original continuum interface are described in [79].

An example of multi-section teleoperation using the continuum interface as it is implemented for the HaptOct can be seen in Figure 2.5. One addition made to the continuum interface for this new device was to include a funnel running through the center of the device directly below the continuum element (labeled in Figure 2.3b). This funnel serves to protect the electronics and device internals in the lower sections of the HaptOct by guiding the excess structural cable away from potential snag points or positions that could short electrical components. The guide funnel also



(a) MiniOct: a 9 DoF extensible continuum interface



(b) HaptOct: a haptic continuum interface

Figure 2.3: Upgrading the passive MiniOct interface (left) to the haptic feedback capable HaptOct (right).

prevents the user from being injured by the moving steel cables.

2.2.3 Configuration Sensing (B)

As with the original MiniOct, we use the measured length of three cables (per section) to determine the length and shape of each section. As an upgrade to the passive and noisy string potentiometers in the original device, we have replaced each string pot with a combination of an HEDS-5540 series optical encoder and a 25.4mm diameter spool that stores excess cable. The encoders are capable of providing 2000 counts per rotation of the spool, giving a granularity of 39.9μm for each cable. In addition to the spools, a pulley located near each spool helps route the

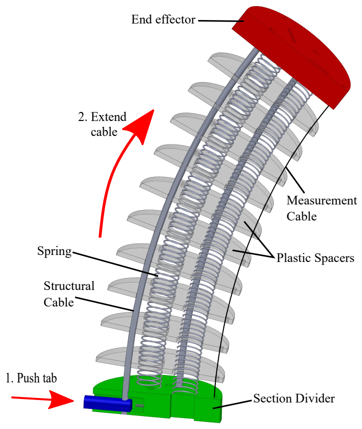


Figure 2.4: Basic Operation of Continuum Section

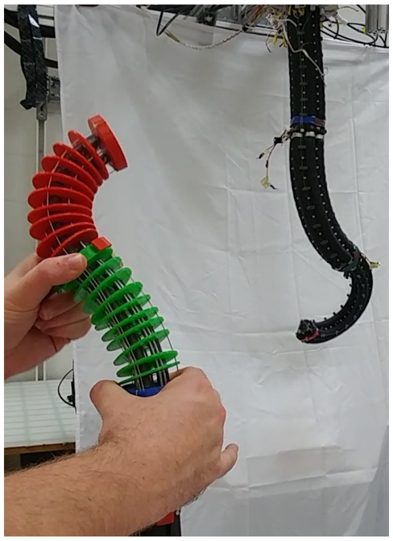


Figure 2.5: Teleoperation Between HaptOct (left) and OctArm Manipulator (right)

cables from the horizontal orientation of the spool to the extending vertical motion of each section. One advantage of this overall arrangement is the elimination of a maximum extension restriction for the overall device (previously 30cm).

The use of the optical encoder and spool also removes the passive, but increasingly opposing, force of the original spring loaded potentiometers that beneficially removed slack on the measurement cables but also made it harder to extend the MiniOct to full length (a noted issue in [79]). In order to replace the slack collection, we used a solution inspired by slack line detection in industry [116]. As seen in Figure 2.6, the solution involves placing a micro-limit switch with a roller in line with each measurement tendon just above the routing pulleys. The cover on the switch keeps the tendon pressed against the roller when there is excessive slack and prevents the tendon from falling away from the pulley or out of alignment. When the switch opens, the resulting signal causes the device to take in slack, using the motors described in Section 2.2.4, until the switch closes. Important to the development was the release force of the switches (8.15 grams-force), as a higher release force could potentially turn the spool and open the switch when no actual slack existed on the line. This solution to use limit switches to detect and avoid slack is similarly implemented in [117]. Other recent solutions in continuum robotics to avoid slack such as [118] and [119] require considerably more complex hardware and foot print than the implemented solution herein.

The changes in length recorded by the encoders are then converted via kinematics [120] to shape values that can be used as input for a slave device as with the MiniOct.

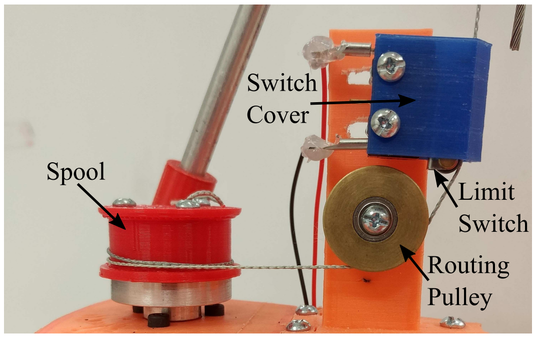


Figure 2.6: Limit Switch and Cover for Slack Detection

2.2.4 Actuation (C)

The HaptOct acts as an impedance-type haptic device by reflecting opposing forces to an operator in response to extensions of the continuum interface. The medium for these opposing forces is the individual measurement cables in the interface, but the forces themselves are generated by DC motors connected to the measurement cable spools. Each spool is connected to the output shaft of a Maxon Brushless DC Motor (series 273753) by a 6mm Pololu mounting hub. Connected to the opposite end of the motor shafts are the encoders for the measurement cables. All nine motors lack a gearbox connected to the output of the motor, thereby allowing the cable spools to easily turn by hand when unpowered, which is critical for allowing the operator to easily manipulate the continuum interface without unintended resistance. These motors are readily available and have a desirable current-to-torque relationship for our needs. The DC motors are driven by a series of

Pololu G2 high power motor drivers, which require a Pulse Width Modulated (PWM) signal and a binary direction signal. Each Maxon motor is mounted to a custom motor mount that encases the entire motor and provides an outlet for the shaft, power cables, and the encoder port.

2.2.5 Processing (D)

The core processing of the HaptOct is executed by an Arduino Due which has an Atmel SAM3X8E ARM Cortex-M3 CPU [121]. The board features an 84MHz clock, 12 PWM pins, 42 GPIO pins, and 12 analog input pins. The Due is capable of communication via serial USB, Inter-integrated Circuit (I^2C) protocol, and Serial Peripheral Interface (SPI). We primarily utilize the serial communication and Bluetooth communication through an external Bluetooth module. The Due was ultimately chosen for use because it enabled the HaptOct to achieve an on-board refresh rate of approximately 2.5kHz when interacting with simple virtual obstacles such as virtual surfaces. The refresh rate for bilateral teleoperation between the HaptOct and a slave device is largely limited by the communication protocol, primarily serial USB communication in this case, and the size of communication packets. The refresh rate for the experiments here averaged around 26Hz.

The device is powered by a 40W AC-to-DC converter that supplies an output voltage of 5V. The 5V output directly powers the Due and 2 DC computer fans that help ventilate the lower sections of the HaptOct. In order to power the motors and motor drivers, the 5V is stepped up to 7.5V using a DC-to-DC converter.

2.2.6 Impedance Display

The logic and theory that drive the HaptOct are detailed in this section. The device is designed to be an impedance display, which measures movement provided by the operator and exerts a force that can be sensed by the operator [82]. Figure 2.7 contains the logic flow for the bilateral teleoperation of a three-section continuum manipulator. The vector value l_m represents the lengths of measurement cables of the master device and l_s defines the actuator lengths of the slave device as measured locally by the slave system. These measurements are used to approximate the kinematic values ($q_{m,s}$) for both the master and slave device, respectively, by each system's local controller. The slave controller sends the vector input τ_s to the slave device as calculated by the controller in order to achieve the designated configuration. The vector value F_m is the opposing force experienced by the operator corresponding to an error vector, e_s, produced by the slave system. The error between the master device and tracking slave device is only one component of the total opposing force felt by an operator. The central block, T_{delay}, represents the lag in time experienced between the master and slave device due to transmission over various networks and physical distance between the two devices. In this work, we do not explore the impact of time lag on user experience and assume a negligible round trip delay in our network.

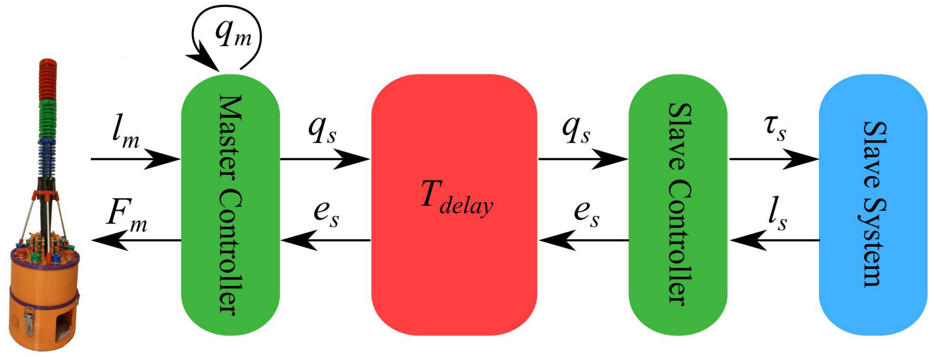


Figure 2.7: Teleoperation Scheme for HaptOct

2.2.7 Master System

As described in Section 2.2.3, the HaptOct is capable of sensing the lengths of the measuring cables through optical encoders. These lengths are converted to the forward kinematic values (as in Section 2.1) used for the slave system and scaled to the kinematic range of the slave. Due to the master device having the same kinematic structure as the slave, we are able to map one-to-one the kinematics of the HaptOct to a slave device during this process. The scaled values are then relayed to the slave system controller.

As previously mentioned, the master system receives the current error seen by the slave system controller. The error values are then used to calculate an error force component which, in addition to other force components, determines the final force experienced by the operator. The reported error can be with respect to either the kinematic values or the actuator level errors (i.e. lengths of tendons/actuators), but the computed output force to the operator is calculated with respect to each individual measurement cable. An example of the relationship between the error, additional force values, and final output force is given by the equation:

$$F_l = C_e\mathbf{e} + C_{\Delta e}\mathbf{\Delta e} + C_{\substack{max \\ length}} f_{\substack{max \\ length}}(l) + ... \tag{2.4}$$

$$f_{\substack{max \\ length}} = (l - l_{max}) \cdot H(l - l_{max}) \tag{2.5}$$

where the scalars C_e and $C_{\Delta e}$ are the force coefficients for the slave state error ($\mathbf{e} \in \Re^{9x1}$) and change in error ($\mathbf{\Delta e} \in \Re^{9x1}$), respectively. The equation for the overall opposing force, F_l, for a cable l can be expanded to incorporate additional force terms. For example, $f_{\substack{max \\ length}}$ defined in Eq. 2.5, and its corresponding coefficient $C_{\substack{max \\ length}}$, exists to prevent the extension of a tendon beyond a predetermined maximum length and thus adds to the overall opposing force F_l only when a cable on the HaptOct is extended beyond a set maximum (as conveyed by the Heaviside step function $H(\cdot)$). This maximum length enforcing term is also smoothed by scaling the force by the extension beyond the maximum length. Additional terms could include forces exerted by a known environment (i.e. an a priori simulated obstacle), or internal forces/dynamics known about the slave device.

In order to provide an example for how the error in configuration between the master and slave device contributes to individual tendon forces, we exploit the Allen kinematic model as reviewed in Section 2.1. We have reported previously in [120] the 3-actuator based equations that relate $s(t)$, $u(t)$, and $v(t)$ to tendon or actuator lengths, reproduced here:

$$l_1 = s(t) - d \cdot v(t) \tag{2.6}$$

$$l_2 = s(t) + \frac{d \cdot v(t)}{2} + \frac{\sqrt{3}d \cdot u(t)}{2} \tag{2.7}$$

$$l_3 = s(t) + \frac{d \cdot v(t)}{2} - \frac{\sqrt{3}d \cdot u(t)}{2} \tag{2.8}$$

In observing these equations, we find a unique relationship between each of the three length values and the values $s(t)$, $u(t)$, and $v(t)$, as expected. This can serve as the basis for forces due to arc-length and bending in our system. By extension of these equations, we can calculate unique error values for each individual tendon by comparing the master configuration with that of the slave. An example of the error term for tendon l_2 is:

$$e_{l_2} = (s_m(t) - s_s(t)) + \frac{v_m(t) - v_s(t)}{2} + \frac{\sqrt{3}(u_m(t) - u_s(t))}{2} \tag{2.9}$$

where (s_m,u_m,v_m) is the master configuration and (s_s,u_s,v_s) is the slave configuration for a single section. The change in error term ($\mathbf{\Delta e}$) can be computed as the time derivative of this error, relating whether or not the slave is converging to the configuration of the master device. The output force computed from combining the various terms is then converted to a torque exerted by each DC motor as a function of current supplied by an individual motor driver.

2.2.8 Slave System

The flow of logic shown in Figure 2.7 assumes the slave system in this scheme has the ability to converge to a desired configuration through closed-loop control and can monitor and communicate error as the system converges or diverges. As in [79], potential slave systems for this scheme can have a variety of actuation methods (pneumatic, tendon, etc.) and different proportions or range-of-motion. This flexibility in types of hardware capable of being teleoperated represents an advantage of the proposed device.

2.2.9 Experimental Validation

We completed a series of experiments to validate both the hardware and the teleoperation scheme. The following subsections describe the experiments and results. The OctArm manipulator [12], illustrated in Figure 2.8, serves as an excellent example slave device in a bilateral teleoperation scheme. The OctArm is a 3-section, 9 DoF continuum robot, actuated by pneumatically driven McKibben actuators. Pressuring the McKibben muscle creates extension along the length of an individual muscle. By connecting three of these muscles (or sets of them at equal pressures) in parallel to form a continuum section, we can extend said section by pressurizing all muscles simultaneously by the same amount or create bending by differentially pressurizing the muscles. The OctArm is then comprised of three of these serially connected sections, with each section actuated by three independently controlled pressures, and capable of independent extension and two DoF of spatial bending, providing the OctArm 9 DoF overall.

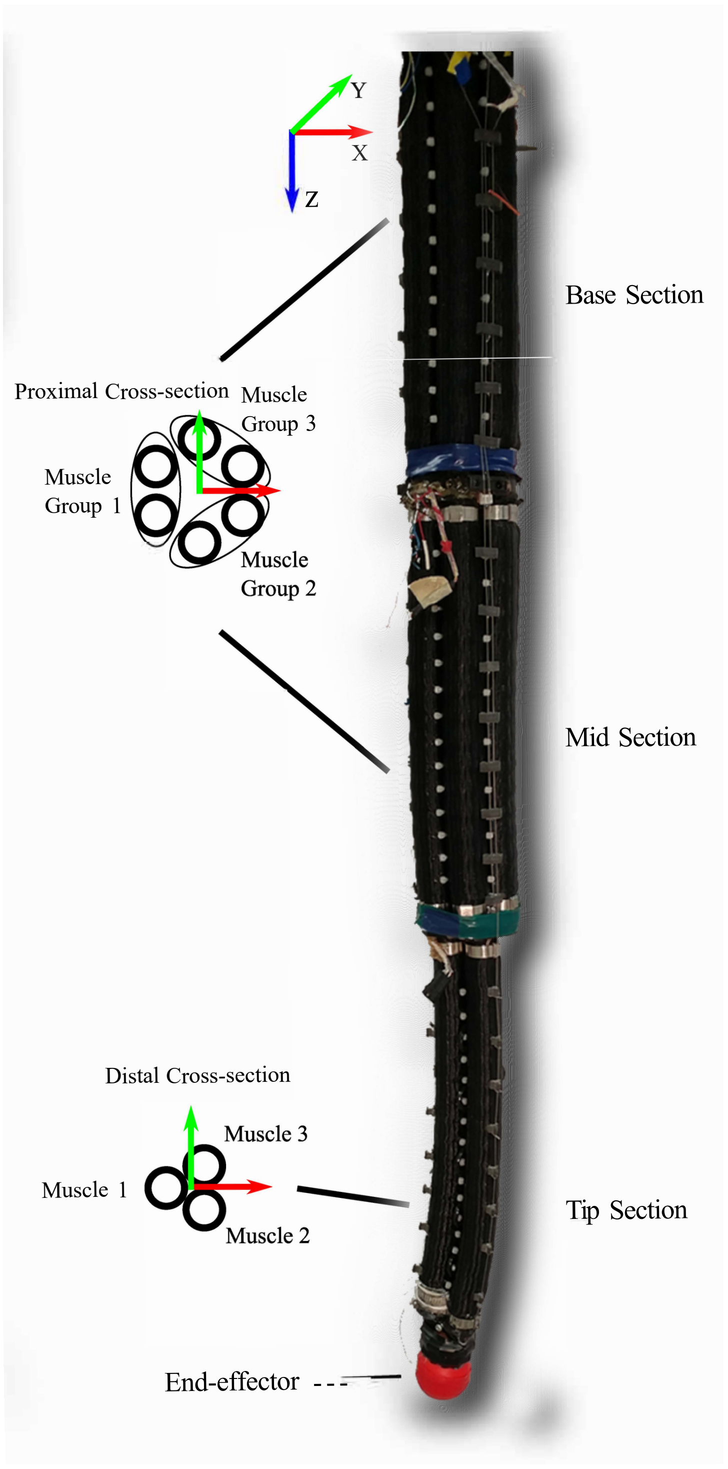


Figure 2.8: OctArm Continuum Manipulator

The OctArm can also be seen in Figure 2.5 where it is being teleoperated using the HaptOct. In the following experiments, the OctArm is laying on top of a horizontal surface.

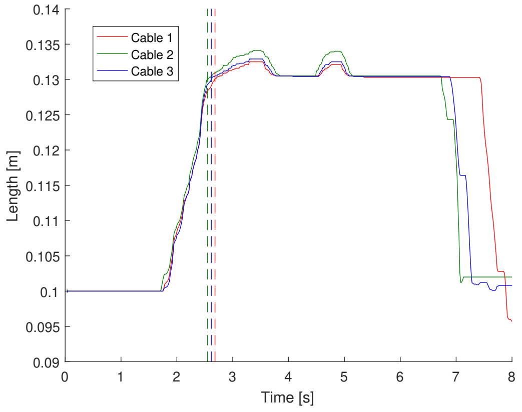
2.2.9.1 Maximum Length Enforcement

The first experiment tests the ability of the system to limit the operator from over extending a section (i.e. the impact of $f_{\frac{max}{length}}$ in Eq. 2.4). As a default, the function prevents the master device from extending beyond 100% of the initial length, a feat that most continuum manipulators are incapable of. When designated to drive a particular slave device, the max extension can be scaled to match the range of motion of the slave. For example, OctArm sections can nominally extend an additional 30% beyond their initial lengths.

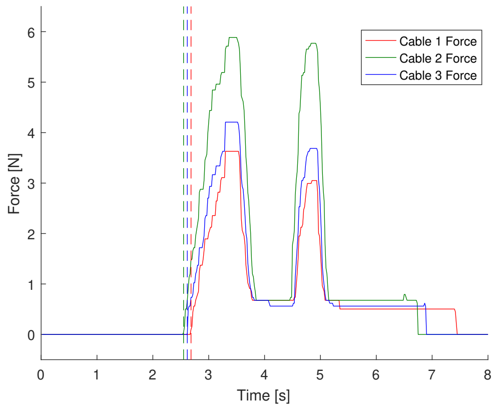
In this experiment, we constrained the distal section of the master device to match the proportional maximum extension of the OctArm. Given the initial section length of 10cm for the HaptOct (as described in [79]), we set a virtual maximum length of 13cm for each individual measurement cable. We then attempted to extend the section to this threshold and beyond while recording the resulting force exerted by the interface. Figure 2.9a shows the lengths of the three measurement cables and is marked with a vertical dashed line when the lengths first pass the maximum length threshold. Figure 2.9b shows the force exerted, calculated in Newtons, for each of the cables. It can be seen that all three cables react proportionately to their individual displacements beyond the maximum and produce sufficient force to be noticed by the operator (>3N per cable). The difference in time for each cable exceeding that maximum and the difference final maximum between cable tensions can be contributed to the motion by the operator not being pure extension (i.e. the user is lengthening at a slight bending angle). This results in cable 2 in this particular experiment being extended slightly more (between 1 and 2mm) than cables 1 and 3, also resulting in a greater force from the HaptOct.

2.2.9.2 Known Obstacle in Slave Task Space

In the second experiment, we evaluated the ability of the HaptOct to oppose invalid configurations given a priori knowledge of the slave system's environment. In particular, we simulate a scenario in which the slave system is known to be laying on a planar surface, restricting the valid motions of the robot to configurations curving or extending away from the plane or moving parallel to the surface.



(a) Measurement cable lengths during distal section extension.



(b) Generated forces corresponding to overextension of distal section.

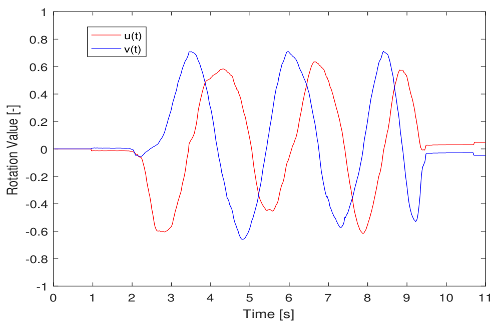
Figure 2.9: Experiment 1, Force output due to extension.

For this experiment, we introduced a new term to the force equation that describes our virtual obstacle. As with Eq. 2.5, we use the Heaviside step function ($H(\cdot)$) to define simple boundaries and use more complex functions for harder to define boundaries. An example equation for describing a virtual obstacle seen by l_2 is:

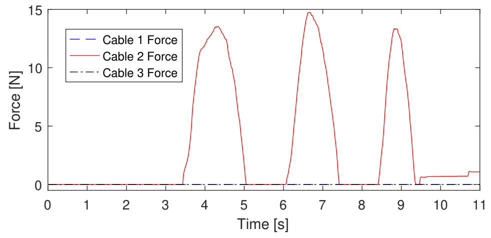
$$f_{2_{virt}} = C_u \cdot u(t) \cdot H(u(t) - u_b) + C_v \cdot v(t) \cdot H(v(t) - v_b) \tag{2.10}$$

where C_u and C_v are the coefficients for the bending displacements $u(t)$ and $v(t)$, respectively, and scalars u_b and v_b serve as simple bounds for $u(t)$ and $v(t)$. The final term $f_{2_{virt}}$ can be added to the total force value defined in Eq 2.4 for tendon 2. Similar terms are also added to the force equations for tendons 1 and 3.

Now that we have a force expression for bending, we can simulate a planar obstacle for the slave device. We set the bound to be the yz-plane, which correlates to setting a bound on $u(t)$ such that $u(t) \geq 0$. Figure 2.10a shows the values of $u(t)$ and $v(t)$ as the operator attempts to bend and rotate the distal section of the HaptOct in all directions. Figure 2.10b shows the three force values corresponding to the changes in $u(t)$ and $v(t)$. As predicted in Eq. 2.10, we see a noticeable increase in force response as the operator tries to bend in the positive $u(t)$ direction. Cable 1 does not apply an opposing force to the operator because it is independent of $u(t)$ and there is no restriction on $v(t)$. Cable 3 does have dependency on $u(t)$, but is not permitted to apply negative forces. This shows the HaptOct is capable of opposing operator motion for basic virtual obstacles.



(a) Kinematic values u(t) and v(t) during rotation of distal section.



(b) Generated forces corresponding to bending of distal section with known planar obstacle.

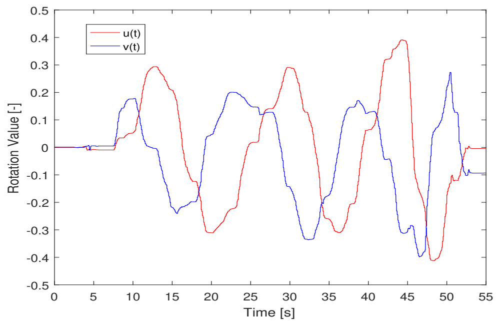
Figure 2.10: Experiment 2, Force output due to bending.

2.2.9.3 Real-time Unknown Obstacle in Slave Task Space

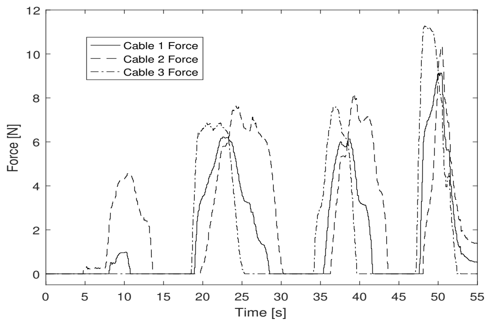
We repeated the previous experiment concerning a slave system placed against a planar surface but removed the a priori knowledge of the surface from our scheme. In this case, we perform real-time bilateral teleoperation and rely on the error between the slave device and master device to convey the planar obstacle to the operator.

As seen in Figure 2.11a, the operator in this experiment rotates the HaptOct section in all directions over 4 distinct periods. We can see the relationship between $u(t)$ and $v(t)$ given this plot. In Figure 2.11b, we see the corresponding forces experienced by the operator during this repeated motion. At an initial glance, it is difficult to determine the obstacle by the reaction forces alone. However, in Figure 2.11c, we see the original motion plot overlaid by the scaled generated forces.

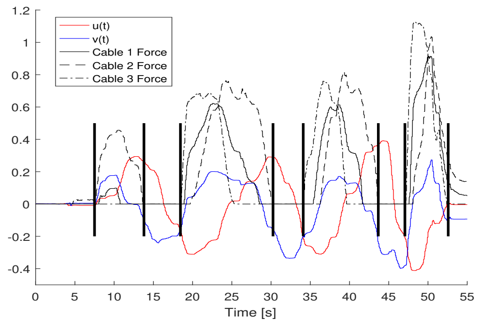
As marked by the vertical bars in the plot, it is easy to now see that the periods of time when we have the largest force exerted by each tendon align with the periods when the magnitude of $v(t)$ is positive or nearly positive. The value of $u(t)$ during these same periods spans the entire range of $u(t)$ used by the operator. The operator can conclude from this result that the slave robot is laying in the xz-plane and oriented such that values of $v(t) \geq 0$ result in collision with the plane.



(a) Kinematic values u(t) and v(t) during rotation of distal section.



(b) Generated forces corresponding to bending of distal section with unknown planar obstacle.



(c) Combined scaled plot of positions and forces.

Figure 2.11: Experiment 3 Results, Observing unknown obstacles.

2.3 Virtual Environments for Exploration and Deployment of Continuum Solutions

When considering remote teleoperation and deployment of continuum manipulators, the ability to plan and simulate potential interactions and tasks can greatly assist the process of creating effective solutions. It is also useful to give consideration to collaboration between continuum robots and other robotic platforms. This section explores the establishment and use of a universal continuum robot package designed to be compatible with the Robot Operating System (ROS), a framework for communicating between robotic platforms and various tools, sensors, and software solutions.

2.3.1 Simulating the physics of Continuum Robots in Gazebo

SDF model description language used in order to incorporate the use of universal joints in the physics enabled body. The trade-off here is a more physically accurate model for software like Gazebo, but the need for a second, kinematically similar model for tools such as RVIZ, which do not recognize universal joint types.

Thankfully, it is a simple matter to map between the two types of models using the set of equations in Table 2.1. For convenience, the equations of note are reproduced here:

$$\theta = s\kappa = \sqrt{u^2 + v^2} \tag{2.11}$$

$$\phi = \arctan \frac{-u}{v} \tag{2.12}$$

$$u = -s\kappa \sin(\phi) \tag{2.13}$$

$$v = s\kappa \cos(\phi) \tag{2.14}$$

The following elements form the base components of the physics enabled model for a continuum manipulator in Gazebo. First is the universal joint, defined to have two orthogonal axes of rotation, one about the local x-axis and the second about the local y-axis, corresponding to the kinematic values u and v, respectively.

```
<joint name='RJB0' type='universal'>
  <parent>base_link</parent>
  <pose frame=''>0 0 -0.0435 0 0 0</pose>
  <child>bSeg0</child>
  <axis>
    <xyz>1 0 0</xyz>
    <limit>
        <lower>-0.314</lower>
        <upper>0.314</upper>
        <effort>10</effort>
        <velocity>0.5</velocity>
    </limit>
    <dynamics>
      <damping>0.002</damping>
    </dynamics>
  </axis>
  <axis2>
    <xyz>0 1 0</xyz>
    <limit>
      <lower>-0.314</lower>
      <upper>0.314</upper>
      <effort>10</effort>
      <velocity>0.5</velocity>
    </limit>
    <dynamics>
      <damping>0.002</damping>
    </dynamics>
  </axis2>
  <physics>
    <ode>
      <limit>
        <cfm>0</cfm>
        <erp>0.2</erp>
      </limit>
      <suspension>
        <cfm>0</cfm>
        <erp>0.2</erp>
      </suspension>
    </ode>
  </physics>
</joint>
```

Following each universal joint is at least one "rigid" element chosen to represent the backbone of the continuum element. Fitting with the typical appearance of continuum robots, the backbone is approximated via cylinders, linked serially. The following SDF excerpt approximates the backbone to appear as a incremental element of a carbon fiber tube, inspired by the backbone of the Tendril robot. Segment elements are defined to have visual, collision, and inertia volumes. In the case of the cylindrical element, all three volumes are chosen to have the same dimensions.

```
<link name='bSeg0'>
  <pose frame=''>0 0 0.1685 0 0 0</pose>
  <visual name='visual'>
    <geometry>
      <cylinder>
        <length>0.087</length>
        <radius>0.002</radius>
      </cylinder>
    </geometry>
    <pose frame=''>0 0 0 0 0 0</pose>
    <material>
      <ambient>0.1 0.1 0.1 1.0</ambient>
      <diffuse>0.1 0.1 0.1 1.0</diffuse>
      <specular>0.01 0.01 0.01 1</specular>
      <emissive>0 0 0 1</emissive>
    </material>
  </visual>
  <collision name='collision'>
    <geometry>
      <cylinder>
        <length>0.087</length>
        <radius>0.002</radius>
      </cylinder>
    </geometry>
    <pose>0 0 0 0 0 0</pose>
  </collision>
  <inertial name='inertial'>
    <pose frame=''>0 0 0 0 0 0</pose>
    <mass>0.0005</mass>
      <inertia>
        <ixx>3.6e-4</ixx>
        <ixy>0</ixy>
        <ixz>0</ixz>
        <iyy>3.6e-4</iyy>
        <iyz>0</iyz>
        <izz>1.9e-5</izz>
      </inertia>
    </inertial>
  </link>
```

Finally, in this model, we add an second rigid element to the mode for each sub-segment to approximate the spacers placed along the backbone of tendon driven continuum robots. In anticipation of model inclusions, the spacer element is connected to the backbone segment via a "fixed" joint, rigidly connecting the two elements. Future work will explore replacing the fixed joint with a prismatic joint in order to enable extension of continuum sections.

```
<joint name='FJB0' type='fixed'>
  <parent>bSeg0</parent>
  <child>bDisk0</child>
  <pose>0 0 -0.0015 0 0 0</pose>
</joint>

<link name='Spacer0'>
    <pose frame=''>0 0 0.2135 0 0 0</pose>
  <visual name='visual'>
    <geometry>
      <cylinder>
        <length>0.003</length>
        <radius>0.008</radius>
      </cylinder>
    </geometry>
    <pose frame=''>0 0 0 0 0 0</pose>
    <material>
      <ambient>0 1.0 1.0 1.0</ambient>
      <diffuse>0 1.0 1.0 1.0</diffuse>
      <specular>0 0.01 0.01 1</specular
      <emissive>0 1.0 1.0 1</emissive>
    </material>
    <cast_shadows>1</cast_shadows>
  </visual>
  <collision name='collision'>
    <geometry>
      <cylinder>
        <length>0.003</length>
        <radius>0.008</radius>
      </cylinder>
    </geometry>
    <pose>0 0 0 0 0 0</pose>
  </collision>
  <inertial>
    <pose>0 0 0.0 0 0 0</pose>
    <mass>0.0054</mass>
    <inertia>
      <ixx>7.0e-4</ixx>
      <ixy>0</ixy>
      <ixz>0</ixz>
      <iyy>7.0e-4</iyy>
      <iyz>0</iyz>
      <izz>1.3e-3</izz>
    </inertia>
  </inertial>
</link>
```

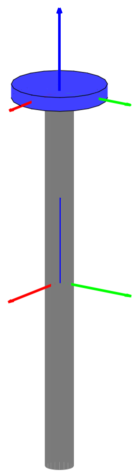


Figure 2.12: Continuum element

The body and joint elements thus far work to convey the physical appearance of the continuum robot and capture the basic motion and shape. In order to further reflect the nature of continuum manipulators, we introduce a custom plugin for Gazebo that goes one step further to also approximate the material properties of a continuum robot's backbone. In a relatively simple model, conveyed in Figure 2.13, the plugin exerts opposing torque to both axes of each universal joint in direct proportion to the displacement of the axes. This approximation of the material "stiffness" can be adjusted during the setup of simulation and customized for each section of a continuum robot.

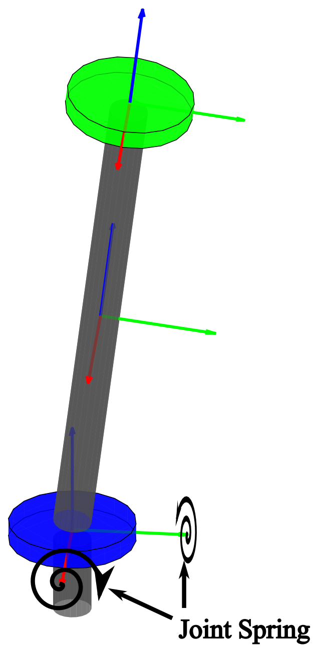


Figure 2.13: Illustration of "jointSpring2" plugin that simulates torsional springs placed on each of the axes of the universal joints. This plugin simulates the material stiffness by opposing bending.

2.3.2 Creating a Visual Simulation of Continuum Robots for ROS: RVIZ

RVIZ, short for ROS Visualizer, is the central visual tool associated with ROS for simulating robot state based on feedback from a physical robot platform or from physics simulators such as

Gazebo [122]. RVIZ also provides for depicting sensor data such as encoder positions converted to robot configuration or point clouds and image data from a robot's environment. RVIZ utilizes a different descriptor language from Gazebo which does not allow for universal joints. The Unified Robot Descriptor Format (URDF) language used in RVIZ only allows for basic revolute, prismatic, and fixed joints (URDF also recognizes planar and floating joints, but these are not useful to us). Instead, we utilize the Jones kinematics to create a visual continuum robot for RVIZ, much like that which is created in [90]. Since this solution is not unique, we skip the details of its creation and only note that when communicating between the physics enabled model and RVIZ, we can utilize the mapping from Jones to Allen kinematics to allow the models to collaborate.

2.3.3 RVIZ and MoveIt

In conjunction with RVIZ, the tool MoveIt can be used to perform basic motion planning of a continuum manipulator. Using the URDF model, MoveIt can incrementally plan steps between two desirable configurations. The solution/tool can deploy a multitude of variations in mapping between two states, but commonly hosts flavors of Rapidly-exploring Random Trees (RRT) to bridge between the two states. Using RVIZ, this planned trajectory can then be sent to the physical robot or the physics based simulation in Gazebo. Examples of MoveIt performing RRT stepping between configurations in shown in Figure 2.14.

With the ability to connect to the various tools available to ROS, a final goal for this package will be the planning and execution of a collaborative task alongside other robotic platforms in simulation. One such concerted effort could be the exploration and inspection of a virtual ISS alongside Robonaut [123] (see Figure 2.15).

2.4 Discussion

The underlying goal of this work was to design and construct a continuum device capable of providing intuitive haptic feedback to an operator while teleoperating a wide range of continuum manipulators. Much of the design work has been completed and presented here but further testing remains necessary to validate the long term success of this strategy.

There were numerous constraints that influenced the design of the mechanism. First, the system needed to minimize resistance to motion seen by the operator when there is no acting force

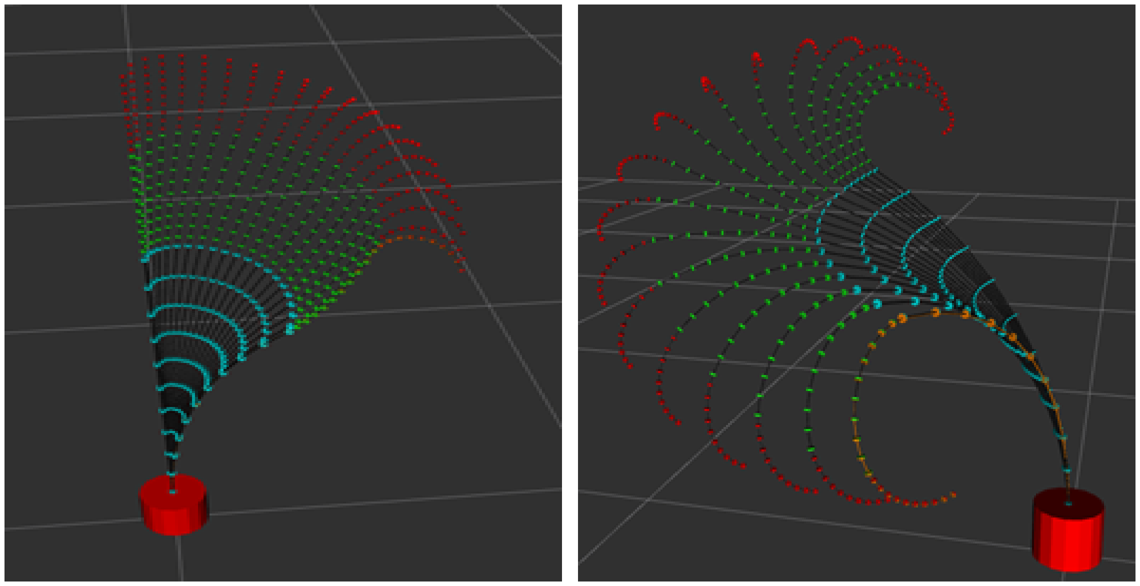


Figure 2.14: An example of MoveIt coupled with RVIZ to plan and visual an RRT selected path between two configurations.

Figure 2.15: Simulating collaboration with ROS enabled platforms to complete tasks aboard the International Space Station.

from the device. This led to the use of DC motors without a gear system. This reduces the maximum opposing force the system can exert but gives the least inherent resistance. Second was the desire to develop a compact device that did not have an overly bulky actuator package. As depicted in Figure 2.3b, the complete actuator package, logic, and power supply measures 20.3cm in diameter and 30.9cm in height. Another constraint was the desire for intuitive feedback, both visual and haptic, which led to the use of the kinematically similar three section MiniOct design as the base for developing a haptic device.

In the initial stages of developing this device, choosing the mode of feedback for developing a new haptic device was a cost and reward driven process. As mentioned in [82], tactile and vibrotactile feedback are common modes. We determined that for an initial investigation, the process of creating a synthetic skin for the continuum device, while intriguing, would likely be very difficult to realistically achieve and keep the device portable and relatively simple. The tactile interface would likely be very useful for conveying contact between the slave system and the environment, but would also require a significant amount of information about the slave system in order to accurately portray points of contact. Vibrotactile feedback was much easier to envision as the actuators are relatively small and would likely only require simple error models. A drawback of a vibrotactile mode would be the need to run wires through continuum sections and it could prove difficult to localize the vibration to a single side of the device or single section for a multi-section system. An admittance feedback device was not investigated in detail, but could prove to be an area of future exploration.

Through initial testing of the device and observations about the design of the system, there are some improvements that can be made to the current design. As with many bilateral teleoperation schemes, the performance of the operator and their ability to perceive mismatches between the master and slave device can depend on the refresh rate and lag experienced by the operator. Initial tests on a local system using serial communication has achieved a round trip refresh rate of 26Hz, the delay from which will likely be noticed by the user. A proposed solution to this will be to use a faster mode of communication, such as SPI or I2C for local communication. As mentioned in Section 2.2.5, the HaptOct as a standalone device has an average refresh rate above 2.5kHz, which provides opportunity for the development of more complex virtual environments that do not rely on communication lags while maintaining smooth operation. When performing the experiments over a distance of hundreds of miles (performed from Houston, Texas to Clemson, South Carolina in the U.S.A.) we experienced a lag of approximately 0.1 seconds. We did not analyze the impact of this delay on performance, though it was negligible enough to perform the experiments and collect the results presented in Sections 2.2.9.2 and 2.2.9.3. Future work can now explore appropriate means for addressing lag of varying severity.

Another limitation is that the device can only oppose the operator when extending any one of the three sides of the device. The limitation arises from the fact that the device always inherently assumes that the slave system can retract or straighten from an extended or curved configuration. This assumption is not that disruptive in a static environment, but could prove erroneous for dynamic environments in which a slave system might become trapped.

The largest challenge for this haptic device (and any haptic device with regards to continuum manipulators) is the conclusion that the feedback is only as informative and accurate as the error model that is available for the slave system. One example of this could be a scenario in which a distal section of a slave system has come in contact with an object, but is able to bend or extend normally. However, the proximal sections begin to curve due to the compliant nature of continuum robots. The operator may not perceive this deformation until attempting to later manipulate the proximal section. Another scenario particular to pneumatically actuated continuum manipulators is the increase in stiffness of a muscle as it is extended. This internal stiffness may be difficult to detect through error modeling, but would be useful feedback for the operator to know that the system is approaching its physical limits.

Chapter 3

Continuum Robot Modeling

The following chapter will explore continuum robot dynamics as approximated as a serial chain of rigid bodies. First, we will briefly review previous work that approximates a continuum section as a single rigid body representing the chord of the arc formed by a constant-curvature continuum section. We then introduce a new model that extends this approximation to also partially capture non-constant curvature within a continuum section. The model is derived using both the Euler-Lagrange method and the Newton-Euler recursive method, and considers gravitational effects and material properties. In order to approximate the full state of the introduced model, we present a energy minimization method for estimating the unobservable model states and verify the method using a series of static samples and external measurements. Finally, we apply a set of simple inputs and compare the predicted behavior of the simulation to the actual response of the OctArm robot and discuss the results and implications.

3.1 Approximating Continuum Robots Through Rigid Bodies

In considering continuum kinematics, some of the most well-known early models for continuum robots drew inspiration from traditional rigid-body kinematics to relate continuum DoF to those of standard revolute and prismatic joints. Similarly, in [72], it was shown that the traditional and well-established topic of rigid-body dynamics can also provide insight and approximate models

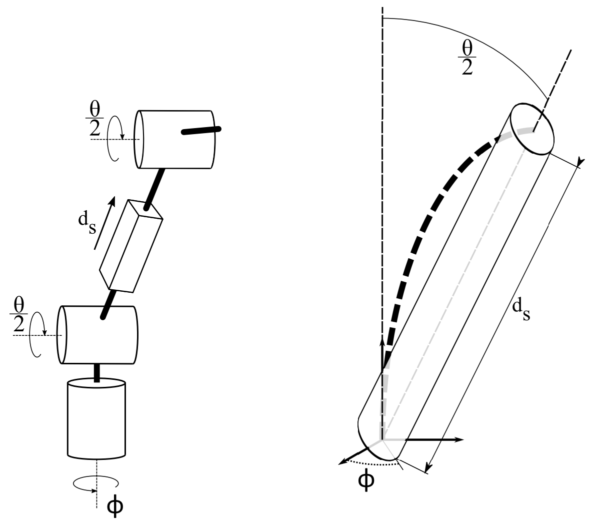


Figure 3.1: Constant-curvature approximate kinematics for a single continuum section as conveyed by an RRPR rigid-link robot.

for the dynamics of continuum manipulators.

3.1.1 Constant-curvature Approximate Dynamics

In [72] and [73], a dynamic model is introduced that expands the concept introduced in [124], whereby the shape and motion of continuum robots are approximated through a virtual rigid-link robot. In the kinematic model, the shape of a continuum section curve is approximated as the chord formed by a curving continuum section, stretching from the section's base to its end-point. Orientation and direction of the chord are approximated by revolute joints located at the base of the robot section. The dynamic model in [73] expands this idea by assigning mass and inertia to the chord in the form of a uniform cylinder, as conveyed in Figure 3.1. The cylinder's orientation, length, and direction of bend is related to the rigid-link kinematics through the Jones kinematic model and subsequent dynamics are derived to approximate this system based on the virtual rigid-link kinematics. This work was novel from the aspect of approximating a continuum robot by rigid bodies on a large scale (previous works approximate continuum robots as a series of infinitesimally small rigid elements). The results of the model combined with a basic dynamics cancelling controller showed potential for the method in controlling general continuum motion for multiple sections.

3.2 Piece-wise Constant Curvature Approximate Dynamics

In considering the work presented in [73], a desirable refinement would be to further the approximation of dynamics for continuum manipulators to produce a more accurate, but still computationally tractable model. In particular the following model, introduced in this dissertation, will be used to approximate the non-constant curvature manifest in continuum sections through a piece-wise constant curvature approximation, as well as to introduce new material property terms. Figure 3.2 demonstrates a simple interpretation of what a piece-wise constant curvature can do to approximate the continuum dynamics versus the constant curvature model. As seen, the piece-wise constant model will contain two discrete elements per section; the first approximates the curvature between the base and mid-point of a section and the second approximates the curvature between the mid and end-point of each section.

In [73], the model accounted for extensible continuum robots by modeling all three DoF for a section. In the parameterization presented in the following, we have effectively doubled the DoF present in a single section. In order to keep the number of DoF per continuum section to be more reasonable, we model the continuum manipulator to be a non-extensible section, resulting in a fixed arc-length and restricting the DoF per section to 4 DoF, two values of u and two values of v.

3.2.1 Rigid-Link Kinematics

In using a rigid-link approximation, we reap the benefit of well established kinematics for rotations about single axes in rigid-link robots. For this model, we use the Allen kinematic parameterization which will translate to two orthogonal rotation axes at the base of each segment, demonstrated in Figure 3.3. The standard Denavit-Hartenberg method for describing the transformation matrix for each DoF is applicable here, but for the sake of keeping unique, identifiable transformation matrices, we use a global coordinate system translated along the length of the robot. Using the global coordinate frame, we can use the standard rotation matrices R_x and R_y to represent the rotations u and v with no translation component and then add a pure translation along the Z-axis to represent the length of each rigid segment. Thus the transformation matrices of interest are:

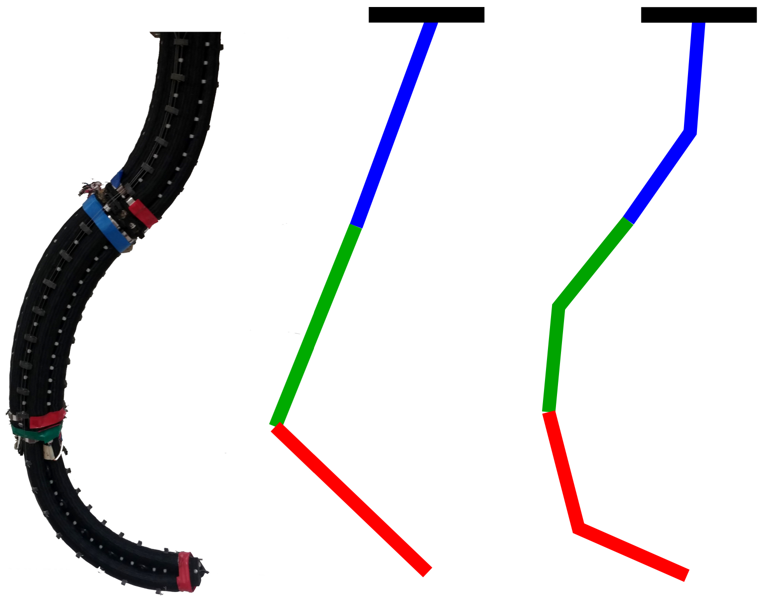


Figure 3.2: Comparing Approximate Models. The OctArm (left) approximated as a constant-curvature series of chord lengths (middle) and as piece-wise constant-curvature 2 segment curves (right).

$$H_u(i) = \begin{bmatrix} \mathbf{R}_{x,u_i} & \mathbf{0}_3 \\ \mathbf{0}_3^T & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(u_i) & -\sin(u_i) & 0 \\ 0 & \sin(u_i) & \cos(u_i) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

$$H_v(i) = \begin{bmatrix} \mathbf{R}_{y,v_i} & \mathbf{0}_3 \\ \mathbf{0}_3^T & 1 \end{bmatrix} = \begin{bmatrix} \cos(v_i) & 0 & \sin(v_i) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(v_i) & 0 & \cos(u_i) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

$$H_s(i) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_{s_i} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

where d_{s_i} is the chord length that approximates the arc-length of the section, approximately half of the length of a full continuum section. The matrices $\mathbf{R}_{x,u} \in \mathbb{R}^{3\times3}$ and $\mathbf{R}_{y,v} \in \mathbb{R}^{3\times3}$ are the standard rotation matrices $\mathbf{R}_x$ at angle u_i and $\mathbf{R}_y$ at angle v_i, respectively. While the order of the rotations conveyed by u and v can be set arbitrarily (so long as the order is consistent along the robot), for consistency and for dynamics derivation, we establish that the order of rotations is u and then v, followed by the translation along the length of the segment. This gives the total transformation matrix for each segment to be:

$$H(i) = H_u(i)H_v(i)H_s(i) \tag{3.4}$$

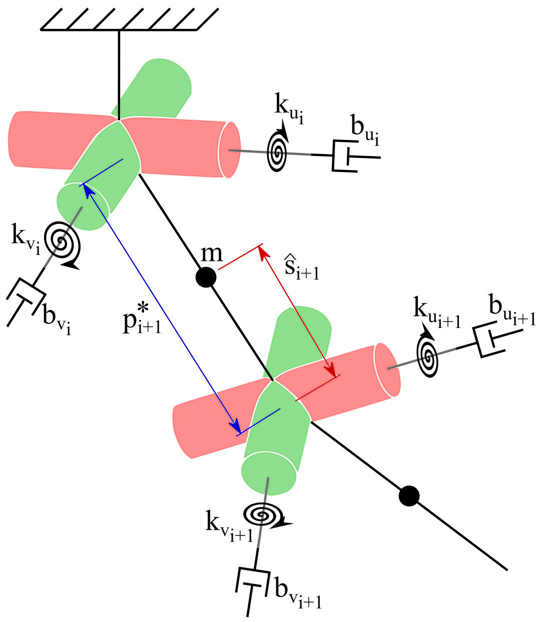


Figure 3.3: Joint kinematics and physical properties. The u-approximated joints (red) are coincident with the local Y-Axis and v-approximated joints are coincident with the X-axis. Vector p^*_{i+1} points from origin O_i to origin O_{i+1}. Vector $\hat{s}_{i+1}$ points from origin O_{i+1} to the center of mass of link i.

3.2.2 Euler-Lagrange Dynamic Model

In deriving a series of closed form equations for the dynamic model, we first use the well-established Euler-Lagrange mechanics, in which the Lagrangian in Eq. 3.5, relates the total kinetic energy ($\mathcal{K}$) and total potential energy ($\mathcal{P}$) of the system. We obtain the equations of motion for the approximate model according to Eq. 3.6.

$$\mathcal{L} = \mathcal{K} - \mathcal{P} \tag{3.5}$$

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}_i} - \frac{\partial \mathcal{L}}{\partial q_i} = \tau_i \tag{3.6}$$

The vector $\mathbf{q}$, representing the position of joints of the system, is comprised of the alternating approximate joint angles u_i and v_i for segment i as follows:

$$\mathbf{q} = [u_1 \quad v_1 \quad u_2 \quad v_2 \quad \dots \quad u_n \quad v_n]^T \tag{3.7}$$

3.2.2.1 Kinetic Energy

The kinetic energy of the approximate robot consists of the energy due to linear velocity and angular velocity, as conveyed for a single body by:

$$\mathcal{K} = \frac{1}{2} m v^T v + \frac{1}{2} \omega^T \mathcal{I} \omega \tag{3.8}$$

where m is the mass of the body, v is the linear velocity of the body, ω is the angular velocity, and $\mathcal{I} \in \mathbb{R}^{3\times 3}$ is the inertia tensor of the body. In this model, all bodies in the approximate model are described as cylinders with radius r and height h, where rotation occurs about the end-points of the cylinder, resulting in the inertia tensor matrix:

$$\mathcal{I} = \begin{bmatrix} \frac{1}{12}m(3r^2+h^2) & 0 & 0 \\ 0 & \frac{1}{12}m(3r^2+h^2) & 0 \\ 0 & 0 & \frac{1}{2}mr^2 \end{bmatrix}. \tag{3.9}$$

The kinetic energy of the full, multi-body system is represented by the inertia matrix ($M(q) \in \mathbb{R}^{2n\times 2n}$), calculated using the Jacobian and mass/inertia tensor:

$$\mathcal{K} = \frac{1}{2} q^T M(q) q \tag{3.10}$$

$$M(q) = \sum_{i=1}^{n} J_i^T(q) \mathcal{M}_i J_i(q) \tag{3.11}$$

where $J_i \in \mathbb{R}^{6\times 2n}$ is the Jacobian corresponding to segment i of the robot, n is the total number of discrete segments, and $\mathcal{M}_i \in \mathbb{R}^{6\times 6}$ is defined as:

$$\mathcal{M}_i = \begin{bmatrix} m_i \mathrm{I}_3 & \dots \\ \dots & \mathcal{I}_i \end{bmatrix} \tag{3.12}$$

with $I_3 \in \mathbb{R}_{3\times 3}$ defined to be the identity matrix and $\mathcal{I}_i$ is the inertia tensor for segment i. The standard Jacobian for segment i is composed as $J_i = [J_{v_i}^T J_{\omega_i}^T]$, in which $J_{v_i} \in \mathbb{R}^{3\times 2n}$ is the linear Jacobian, found by taking the partial derivatives of the position of the body's center of mass with respect to q, and $J_{\omega_i} \in \mathbb{R}^{3\times 2n}$ is the angular Jacobian, composed as follows:

$$J_\omega = \begin{array}{c} \begin{bmatrix} \mathbf{x}_0 & \mathbf{y}_1 & \mathbf{x}_2 & \mathbf{y}_3 & \dots & \mathbf{x}_{2i-2} & \mathbf{y}_{2i-1} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \end{bmatrix} \\ \begin{matrix} u_1 & v_1 & u_2 & v_2 & \dots & u_i & v_i & u_{i+1} & v_{i+1} & \dots & u_n & v_n \end{matrix} \end{array}. \tag{3.13}$$

The angular Jacobian in this model is composed of alternating axes of rotation just as the kinematic model, based in global frames, alternates between rotations about the x-axis for u-related joints and about the y-axis for v-related joints. This differs from the Denavit-Hartenberg method in which all rotations occur about the z-axis, causing the angular Jacobian to have the form $J_\omega = [\mathbf{z}_0 \quad \mathbf{z}_1 \quad \dots \quad \mathbf{z}_{2n}]$. The particular vectors $\mathbf{x}_0$,$\mathbf{y}_1$,... are found from the robot's kinematics as follows:

$$H_0^1 = H_u(1) = \begin{bmatrix} \mathbf{x}_0 & \mathbf{y}_0 & \mathbf{z}_0 & \mathbf{p}_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.14}$$

$$H_0^2 = H_0^1 H_v(2) H_s(2) = \begin{bmatrix} \mathbf{x}_1 & \mathbf{y}_1 & \mathbf{z}_1 & \mathbf{p}_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.15}$$

$$H_0^3 = H_0^2 H_u(3) = \begin{bmatrix} \mathbf{x}_2 & \mathbf{y}_2 & \mathbf{z}_2 & \mathbf{p}_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.16}$$

$$\vdots \tag{3.17}$$

Note that the placement of H_s after H_v in 3.15 does not impact the values of $\mathbf{x}_i$ and $\mathbf{y}_i$ and purely serves to preserve the accuracy of the overall kinematics.

After finding the inertia matrix M, the Centripetal-Coriolis matrix is found using the standard Christoffel symbols such that for $C(q, \dot{q}) \in \mathbb{R}^{2n \times 2n}$:

$$c_{kj} = \sum_{i=1}^{2n} c_{ijk}(q)\dot{q} \tag{3.18}$$

$$= \sum_{i=1}^{2n} \frac{1}{2}\left\{\frac{\partial m_{kj}}{\partial q_j} + \frac{\partial m_{ki}}{\partial m_{q_j}} - \frac{\partial m_{ij}}{\partial q_k}\right\}\dot{q}_i \tag{3.19}$$

where c_{kj} is the value in the k-th row, j-th column of $C(q, \dot{q})$ and similarly m_{kj} is the value in the k-th row, j-th column of $M(q)$. This definition of $C(q, \dot{q})$ provides for the skew-symmetric relationship between $M(q)$ and $C(q, \dot{q})$ such that:

$$N(q, \dot{q}) = \dot{M}(q) - 2C(q, \dot{q}) \tag{3.20}$$

is skew-symmetric, which is an important property for the study of stability in future control applications.

3.2.2.2 Potential Energy

There are two potential energy terms to factor in for this model, the gravitational potential energy and the elastic potential energy due to bending. The gravitational potential energy, denoted P_g is calculated as the change in height of each center-of-mass (CoM) as returned by the kinematics multiplied with the gravitational constant g. The basic definition of P_g can be given as:

$$P_g = \sum_{i}^{n} m_i g h_i(q) \tag{3.21}$$

for the CoM of each segment, where $h_i(q)$ is the change in height for the CoM as a function of the joint angles q.

The elastic potential due to bending is approximated as a virtual torsion spring, as conveyed in Figure 3.3, where the stored energy is the product of the displacement of each discrete joint (the values u, v per segment) and the torsional stiffness of each respective joint: k_u and k_v. The total energy due to bending is:

$$P_e = \frac{1}{2}\mathbf{q}^T[k_{u_1}, k_{v_1}, k_{u_2}, k_{v_2}, \ldots, k_{u_n}, k_{v_n}]^T\mathbf{q} \tag{3.22}$$

One final term for the model is energy dissipation due to damping, a property inherent to the material properties of continuum robots. Similar to the material stiffness, the damping effect is approximated as a linear rotational damper placed at each revolute joint in the model, with damping coefficients b_u and b_v for u and v related joints, respectively. The total energy dissipation due to damping is related through the Rayleigh dissipation function [125]:

$$P_d = \frac{1}{2}\dot{\mathbf{q}}^T[b_{u_1}, b_{v_1}, b_{u_2}, b_{v_2}, \ldots, b_{u_n}, b_{v_n}]^T\dot{\mathbf{q}} \tag{3.23}$$

3.2.2.3 Equations of Motion

Using equations 3.12, 3.21, 3.22, and 3.23, we can update the Lagrangian as:

$$\mathcal{L} = \mathcal{K} - \mathcal{P} = M - P_g - P_e - P_d, \tag{3.24}$$

giving the equations of motion through Lagrange's equation in 3.6, to be:

$$\frac{d}{dt}\frac{\partial M}{\partial \dot{q}_i} - \frac{\partial M}{\partial q_i} + \frac{\partial P_g}{\partial q_i} + \frac{\partial P_e}{\partial q_i} + \frac{\partial P_d}{\partial \dot{q}_i} = \tau_i \tag{3.25}$$

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) + E(q) + R(\dot{q}) = \tau \tag{3.26}$$

where τ is the input torque vector composed of the input torque for each joint. For a two section continuum robot, the components of equation 3.26 will have the form:

$$M(q) \triangleq \begin{bmatrix} M_{11} & M_{12} & M_{13} & M_{14} & M_{15} & M_{16} & M_{17} & M_{18} \\ M_{21} & M_{22} & M_{23} & M_{24} & M_{25} & M_{26} & M_{27} & M_{28} \\ M_{31} & M_{32} & M_{33} & M_{34} & M_{35} & M_{36} & M_{37} & M_{38} \\ M_{41} & M_{42} & M_{43} & M_{44} & M_{45} & M_{46} & M_{47} & M_{48} \\ M_{51} & M_{52} & M_{53} & M_{54} & M_{55} & M_{56} & M_{57} & M_{58} \\ M_{61} & M_{62} & M_{63} & M_{64} & M_{65} & M_{66} & M_{67} & M_{68} \\ M_{71} & M_{72} & M_{73} & M_{74} & M_{75} & M_{76} & M_{77} & M_{78} \\ M_{81} & M_{82} & M_{83} & M_{84} & M_{85} & M_{86} & M_{87} & M_{88} \end{bmatrix} \tag{3.27}$$

$$C(q, \dot{q}) \triangleq \begin{bmatrix} C_{11} & C_{12} & C_{13} & C_{14} & C_{15} & C_{16} & C_{17} & C_{18} \\ C_{21} & C_{22} & C_{23} & C_{24} & C_{25} & C_{26} & C_{27} & C_{28} \\ C_{31} & C_{32} & C_{33} & C_{34} & C_{35} & C_{36} & C_{37} & C_{38} \\ C_{41} & C_{42} & C_{43} & C_{44} & C_{45} & C_{46} & C_{47} & C_{48} \\ C_{51} & C_{52} & C_{53} & C_{54} & C_{55} & C_{56} & C_{57} & C_{58} \\ C_{61} & C_{62} & C_{63} & C_{64} & C_{65} & C_{66} & C_{67} & C_{68} \\ C_{71} & C_{72} & C_{73} & C_{74} & C_{75} & C_{76} & C_{77} & C_{78} \\ C_{81} & C_{82} & C_{83} & C_{84} & C_{85} & C_{86} & C_{87} & C_{88} \end{bmatrix} \tag{3.28}$$

$$G(q) = \begin{bmatrix} \frac{\partial P_g}{\partial u_1} & \frac{\partial P_g}{\partial v_1} & \frac{\partial P_g}{\partial u_2} & \frac{\partial P_g}{\partial v_2} & \frac{\partial P_g}{\partial u_3} & \frac{\partial P_g}{\partial v_3} & \frac{\partial P_g}{\partial u_4} & \frac{\partial P_g}{\partial v_4} \end{bmatrix}^T \tag{3.29}$$

$$E(q) = [k_{u_1} u_1 \quad k_{v_1} v_1 \quad k_{u_2} u_2 \quad k_{v_2} v_2 \quad k_{u_3} u_3 \quad k_{v_3} v_3 \quad k_{u_4} u_4 \quad k_{v_4} v_4]^T \tag{3.30}$$

$$R(\dot{q}) = [b_{u_1} \dot{u}_1 \quad b_{v_1} \dot{v}_1 \quad b_{u_2} \dot{u}_2 \quad b_{v_2} \dot{v}_2 \quad b_{u_3} \dot{u}_3 \quad b_{v_3} \dot{v}_3 \quad b_{u_4} \dot{u}_4 \quad b_{v_4} \dot{v}_4]^T \tag{3.31}$$

3.2.3 Newton-Euler Dynamics

In evaluating the dynamics of continuum robots with three or more sections, and consequently higher DoF, the closed form equations resulting from the Euler-Lagrange method can quickly become computationally expensive and impractical for real-time implementation, also rendering control via the dynamic model untenable. In order to assess the proposed approximate

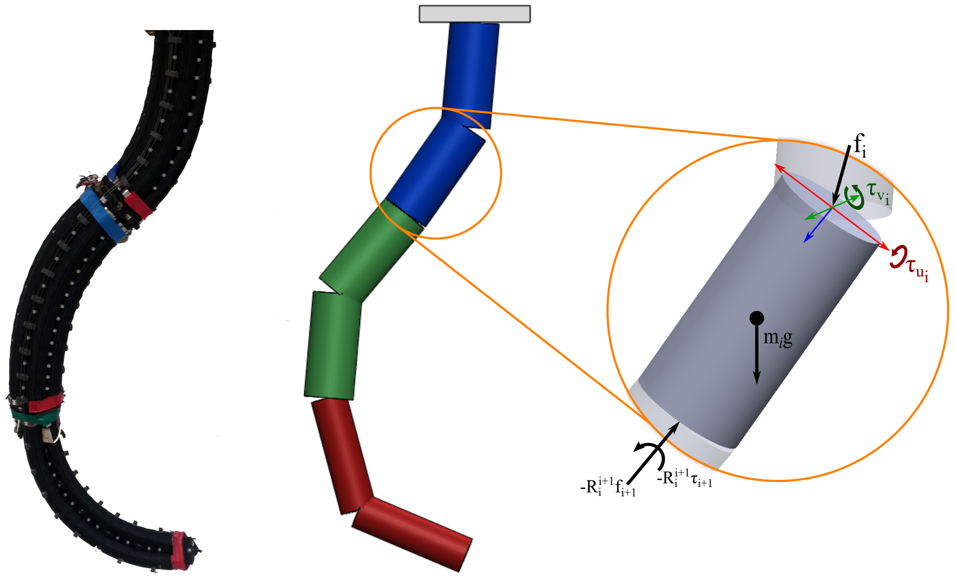


Figure 3.4: Newton-Euler Dynamics

dynamics introduced above for higher DoF, we derive the equivalent model using the recursive Newton-Euler method. In particular, we take advantage of the algorithm presented in [126]. We review the algorithm below, and introduce the relevant matrices for the approximate continuum model.

There are two parts to the Newton-Euler recursion, a forward iterative process in which linear and angular accelerations for each body's CoM are calculated, and a backward iterative process in which the forces and moments experienced by each body are calculated. In addition to the forces and moments acting on each body, the backward recursion also calculates the corresponding torques that are being applied at each joint in order to produce the previously calculated accelerations and forces. In [126] , the values needed for the two processes, namely angular velocity (ω_i), angular acceleration ($\dot{\omega}_i$), linear velocity ($\mathbf{v}_i$), and linear acceleration ($\dot{\mathbf{v}}_i$) in the forward iterations, and total force ($\mathbf{F}_i$) and moment ($\mathbf{N}_i$) in the backwards iterations, are calculated with reference to each link's local (fixed in the link) coordinate frame. This eliminates the need to transform inertia matrix values to a global base frame instead of the more easily defined local frame or the need to track coordinate changes along the length of the robot.

Using the following matrix definitions and properties:

$$\mathbf{R}_0^i = \mathbf{R}_0^1\mathbf{R}_1^2 \dots \mathbf{R}_{i-1}^i \tag{3.32}$$

$$(\mathbf{R}_{i-1}^i)^{-1} = (\mathbf{R}_{i-1}^i)^T = (\mathbf{R}_i^{i-1}) \tag{3.33}$$

in which $\mathbf{R}_{i-1}^i$ is the rotation matrix relating the change in orientation from coordinate frame $i-1$ to frame i, one can directly calculate the velocities and accelerations in the local link frames using the following equations:

$$\begin{aligned}
\mathbf{R}_i^0\boldsymbol{\omega}_i &= \begin{cases} \mathbf{R}_i^{i-1}(\mathbf{R}_{i-1}^0\boldsymbol{\omega}_{i-1} + \mathbf{z}_0\dot{q}_i) & \text{, if joint i is rotational,} \\ \mathbf{R}_i^{i-1}(\mathbf{R}_{i-1}^0\boldsymbol{\omega}_{i-1}) & \text{, if joint i is translational,} \end{cases} \\
\mathbf{R}_i^0\dot{\boldsymbol{\omega}}_i &= \begin{cases} \mathbf{R}_i^{i-1}[\mathbf{R}_{i-1}^0\dot{\boldsymbol{\omega}}_{i-1} + \mathbf{z}_0\ddot{q}_i + (\mathbf{R}_{i-1}^0\boldsymbol{\omega}_{i-1}) \times (\mathbf{z}_0\dot{q}_i)] & \text{, if joint i is rotational,} \\ \mathbf{R}_i^{i-1}(\mathbf{R}_{i-1}^0\dot{\boldsymbol{\omega}}_{i-1}) & \text{, if joint i is translational,} \end{cases} \\
\mathbf{R}_i^0\mathbf{v}_i &= \begin{cases} (\mathbf{R}_i^0\boldsymbol{\omega}_i) \times (\mathbf{R}_i^0\mathbf{p}_i{}^*) + \mathbf{R}_i^{i-1}(\mathbf{R}_{i-1}^0\mathbf{v}_{i-1}) & \text{, if joint i is rotational,} \\ \mathbf{R}_i^{i-1}(\mathbf{z}_0\dot{q}_i + \mathbf{R}_{i-1}^0\mathbf{v}_{i-1}) + (\mathbf{R}_i^0\boldsymbol{\omega}_i) \times (\mathbf{R}_i^0\mathbf{p}_i{}^*) & \text{, if joint i is translational,} \end{cases} \\
\mathbf{R}_i^0\dot{\mathbf{v}}_i &= \begin{cases} (\mathbf{R}_i^0\dot{\boldsymbol{\omega}}_i) \times (\mathbf{R}_i^0\mathbf{p}_i{}^*) + (\mathbf{R}_i^0\boldsymbol{\omega}_i) \times \\ [(\mathbf{R}_i^0\boldsymbol{\omega}_i) \times (\mathbf{R}_i^0\mathbf{p}_i{}^*)] + \mathbf{R}_i^{i-1}(\mathbf{R}_{i-1}^0\dot{\mathbf{v}}_{i-1}) & \text{, if i is rotational,} \\ \mathbf{R}_i^{i-1}(\mathbf{z}_0\ddot{q}_i + \mathbf{R}_{i-1}^0\dot{\mathbf{v}}_{i-1}) + (\mathbf{R}_i^0\dot{\boldsymbol{\omega}}_i) \times (\mathbf{R}_i^0\mathbf{p}_i{}^*) \\ +2(\mathbf{R}_i^0\boldsymbol{\omega}_i) \times (\mathbf{R}_i^0\mathbf{z}_0\dot{q}_i) \\ +(\mathbf{R}_i^0\boldsymbol{\omega}_i) \times [(\mathbf{R}_i^0\boldsymbol{\omega}_i) \times (\mathbf{R}_i^0\mathbf{p}_i{}^*)] & \text{, if i is translational,} \end{cases} \\
\mathbf{R}_i^0\hat{\mathbf{v}}_i &= (\mathbf{R}_i^0\boldsymbol{\omega}_i) \times (\mathbf{R}_i^0\hat{\mathbf{s}}_i) + \mathbf{R}_i^0\mathbf{v}_i \\
\mathbf{R}_i^0\dot{\hat{\mathbf{v}}}_i &= (\mathbf{R}_i^0\dot{\boldsymbol{\omega}}_i) \times (\mathbf{R}_i^0\hat{\mathbf{s}}_i) + (\mathbf{R}_i^0\boldsymbol{\omega}_i) \times [(\mathbf{R}_i^0\boldsymbol{\omega}_i) \times (\mathbf{R}_i^0\hat{\mathbf{s}}_i)] + \mathbf{R}_i^0\dot{\mathbf{v}}_i
\end{aligned} \tag{3.34}$$

where $z_0 = [0, 0, 1]^T$, and $\dot{q}_i$ and $\ddot{q}_i$ is the rotational velocity and acceleration of joint i, respectively. The values $\mathbf{R}_i^0\omega_i$ and $\mathbf{R}_i^0\dot{\omega}_i$ are the angular velocity and acceleration of the CoM of link i with respect to the local frame. The values $\mathbf{R}_i^0 v_i$ and $\mathbf{R}_i^0\dot{v}_i$ are the linear velocity and acceleration of end point of each section, respectively, while $\mathbf{R}_i^0 \hat{v}_i$ and $\mathbf{R}_i^0 \dot{\hat{v}}_i$ are the linear velocity and acceleration for the CoM of the link, again in the local frame. Finally, $\mathbf{p}_i^*$ is defined as the vector pointing from the origin of frame i to the origin of frame $i+1$,as depicted in Figure 3.3. Similarly, $\hat{s}_i$ is defined as the vector from the center of frame $i+1$ to the CoM of link i with respect to frame i. We will wait to explicitly define $\mathbf{p}_i^*$ and $\hat{s}_i$ until after we introduce the specific equations used in this model.

A consequential note concerning these and future Newton-Euler equations in this section: the matrices $\mathbf{R}_i^0$ and $\mathbf{R}_{i-1}^0$ are never explicitly calculated, instead they are compounded with each iteration and become inseparable from their partner, be it ω_i, $\dot{\omega}_i$, etc., which is why $\mathbf{R}_{i-1}^0$ is always surrounded with parentheses anytime the pair is multiplied with another term.

Once linear and angular accelerations are calculated, the total moment and force acting on each link of the robot can be calculated using the following equations:

$$\mathbf{R}_i^0 \mathbf{F}_i = m_i \mathbf{R}_i^0 \dot{\hat{\mathbf{v}}}_i \tag{3.35}$$

$$\mathbf{R}_i^0 \mathbf{N}_i = (\mathbf{R}_i^0 \mathcal{I}_i \mathbf{R}_0^i)(\mathbf{R}_i^0 \boldsymbol{\omega}_i) + (\mathbf{R}_i^0 \boldsymbol{\omega}_i) \times [(\mathbf{R}_i^0 \mathcal{I}_i \mathbf{R}_0^i)(\mathbf{R}_i^0 \boldsymbol{\omega}_i)] \tag{3.36}$$

where m_i again represents the mass of the segment and $\mathbf{R}_i^0 \mathcal{I}_i \mathbf{R}_0^i$ is the equivalent of the inertia tensor for the rigid body, defined in Eq 3.9.

$$\mathbf{R}_i^0 \mathbf{f}_i = \mathbf{R}_i^{i+1}(\mathbf{R}_{i+1}^0 \mathbf{f}_{i+1}) + \mathbf{R}_i^0 \mathbf{F}_i$$

$$\begin{aligned} \mathbf{R}_i^0 \mathbf{n}_i &= \mathbf{R}_i^{i+1}[\mathbf{R}_{i+1}^0 \mathbf{n}_{i+1} + (\mathbf{R}_{i+1}^0 \mathbf{p}_i{}^*) \times (\mathbf{R}_{i+1}^0 \mathbf{f}_{i+1})] + (\mathbf{R}_i^0 \mathbf{p}_i{}^* + \mathbf{R}_i^0 \hat{\mathbf{s}}_i) \times (\mathbf{R}_i^0 \mathbf{F}_i) + \mathbf{R}_i^0 \mathbf{N}_i \\ &= \mathbf{R}_i^{i+1}[\mathbf{R}_{i+1}^0 \mathbf{n}_{i+1} + (\mathbf{R}_{i+1}^i(\mathbf{R}_i^0 \mathbf{p}_i{}^*)) \times (\mathbf{R}_{i+1}^0 \mathbf{f}_{i+1})] + (\mathbf{R}_i^0 \mathbf{p}_i{}^* + \mathbf{R}_i^0 \hat{\mathbf{s}}_i) \times (\mathbf{R}_i^0 \mathbf{F}_i) + \mathbf{R}_i^0 \mathbf{N}_i \end{aligned}$$

$$\tau_i = \begin{cases} (\mathbf{R}_i^0 \mathbf{n}_i)^T (\mathbf{R}_i^{i+1} \mathbf{z}_0) + b_i \dot{q}_i + k_i q_i & \text{, if link is rotational,} \\ (\mathbf{R}_i^0 \mathbf{f}_i)^T (\mathbf{R}_i^{i+1} \mathbf{z}_0) + b_i \dot{q}_i + k_i q_i & \text{, if link is translational.} \end{cases} \tag{3.37}$$

Given the equations for both the forward and backward recursions for the model, and using the knowledge that the model introduced herein for continuum robots only consists of revolute joints, we can narrow down the equations we need and provide the form of equations for joints related to the bending values u and v separately. The subsequent equations and various definitions will be used for the final realization of the recursive algorithm introduced in this dissertation.

$$\mathbf{R}_i^0\boldsymbol{\omega}_i = \begin{cases} \mathbf{R}_{x,q_i}^T[\mathbf{R}_{i-1}^0\boldsymbol{\omega}_{i-1} + \mathbf{x}_0\dot{q}_i] & \text{, if i is odd (u-designated joint),} \\ \mathbf{R}_{y,q_i}^T[\mathbf{R}_{i-1}^0\boldsymbol{\omega}_{i-1} + \mathbf{y}_0\dot{q}_i] & \text{, if i is even (v-designated joint),} \end{cases} \tag{3.38}$$

$$\mathbf{R}_i^0\dot{\boldsymbol{\omega}}_i = \begin{cases} \mathbf{R}_{x,q_i}^T[\mathbf{R}_{i-1}^0\dot{\boldsymbol{\omega}}_{i-1} + \mathbf{x}_0\ddot{q}_i + (\mathbf{R}_{i-1}^0\boldsymbol{\omega}_{i-1}) \times (\mathbf{x}_0\dot{q}_i)] & \text{, if i is odd,} \\ \mathbf{R}_{y,q_i}^T[\mathbf{R}_{i-1}^0\dot{\boldsymbol{\omega}}_{i-1} + \mathbf{y}_0\ddot{q}_i + (\mathbf{R}_{i-1}^0\boldsymbol{\omega}_{i-1}) \times (\mathbf{y}_0\dot{q}_i)] & \text{, if i is even,} \end{cases} \tag{3.39}$$

$$\mathbf{R}_i^0\mathbf{v}_i = \begin{cases} \mathbf{R}_{x,q_i}^T(\mathbf{R}_{i-1}^0\mathbf{v}_{i-1}) & \text{, if i is odd,} \\ (\mathbf{R}_i^0\boldsymbol{\omega}_i) \times (\mathbf{R}_i^0\mathbf{p}_i{}^*) + \mathbf{R}_{y,q_i}^T(\mathbf{R}_{i-1}^0\mathbf{v}_{i-1}) & \text{, if i is even,} \end{cases} \tag{3.40}$$

$$\mathbf{R}_i^0\dot{\mathbf{v}}_i = \begin{cases} \mathbf{R}_{x,q_i}^T(\mathbf{R}_{i-1}^0\dot{\mathbf{v}}_{i-1}) & \text{, if i is odd,} \\ (\mathbf{R}_i^0\dot{\boldsymbol{\omega}}_i) \times (\mathbf{R}_i^0\mathbf{p}_i{}^*) + (\mathbf{R}_i^0\boldsymbol{\omega}_i)\times & \\ [(\mathbf{R}_i^0\boldsymbol{\omega}_i) \times (\mathbf{R}_i^0\mathbf{p}_i{}^*)] + \mathbf{R}_v^T(q_i)(\mathbf{R}_{i-1}^0\dot{\mathbf{v}}_{i-1}) & \text{, if i is even,} \end{cases} \tag{3.41}$$

$$\mathbf{R}_i^0\hat{\mathbf{v}}_i = \begin{cases} \mathbf{R}_i^0\mathbf{v}_i & \text{, if i is odd} \\ (\mathbf{R}_i^0\boldsymbol{\omega}_i) \times (\mathbf{R}_i^0\hat{\mathbf{s}}_i) + \mathbf{R}_i^0\mathbf{v}_i & \text{, if i is even} \end{cases} \tag{3.42}$$

$$\mathbf{R}_i^0\dot{\hat{\mathbf{v}}}_i = \begin{cases} \mathbf{R}_i^0\dot{\mathbf{v}}_i & \text{, if i is odd} \\ (\mathbf{R}_i^0\dot{\boldsymbol{\omega}}_i) \times (\mathbf{R}_i^0\hat{\mathbf{s}}_i) + (\mathbf{R}_i^0\boldsymbol{\omega}_i) \times [(\mathbf{R}_i^0\boldsymbol{\omega}_i) \times (\mathbf{R}_i^0\hat{\mathbf{s}}_i)] + \mathbf{R}_i^0\dot{\mathbf{v}}_i & \text{, if i is even} \end{cases} \tag{3.43}$$

$$\mathbf{R}_i^0\mathbf{F}_i = \begin{cases} [0 \quad 0 \quad 0]^T & \text{, if i is odd} \\ m_i\mathbf{R}_i^0\dot{\hat{\mathbf{v}}}_i & \text{, if i is even} \end{cases} \tag{3.44}$$

$$\mathbf{R}_i^0\mathbf{N}_i = \begin{cases} [0 \quad 0 \quad 0]^T & \text{, it i is odd} \\ (\mathbf{R}_i^0\mathcal{I}_i\mathbf{R}_0^i)(\mathbf{R}_i^0\boldsymbol{\omega}_i) + (\mathbf{R}_i^0\boldsymbol{\omega}_i) \times [(\mathbf{R}_i^0\mathcal{I}_i\mathbf{R}_0^i)(\mathbf{R}_i^0\boldsymbol{\omega}_i)] & \text{, if i is even} \end{cases} \tag{3.45}$$

where $\mathbf{x}_0 = [1 \quad 0 \quad 0]^T$ and $\mathbf{y}_0 = [0 \quad 1 \quad 0]^T$. The vectors $\mathbf{p}_i^*$ and $\hat{\mathbf{s}}_i$ that we neglected to define earlier are $\mathbf{p}_i^* = [0, 0, d_{s_i}]$ and $\hat{\mathbf{s}}_i = [0, 0, -d_{s_i}/2]$. As there is no displacement when traveling from joint u to joint v, the terms connected to $\mathbf{p}_i^*$ and $\hat{\mathbf{s}}_i$ disappear. Finally, we update the backward recursion equations to identify the equations used for u and v specified joints:

$$\mathbf{R}_i^0\mathbf{f}_i = \begin{cases} \mathbf{R}_{y,q_{i+1}}(\mathbf{R}_{i+1}^0\mathbf{f}_{i+1}) + \mathbf{R}_i^0\mathbf{F}_i & \text{, if i is odd} \\ \mathbf{R}_{x,q_{i+1}}(\mathbf{R}_{i+1}^0\mathbf{f}_{i+1}) + \mathbf{R}_i^0\mathbf{F}_i & \text{, if i is even} \end{cases} \tag{3.46}$$

$$\mathbf{R}_i^0\mathbf{n}_i = \begin{cases} \mathbf{R}_{y,q_{i+1}}[\mathbf{R}_{i+1}^0\mathbf{n}_{i+1}] + \mathbf{R}_i^0\mathbf{N}_i & \text{, if i is odd} \\ \mathbf{R}_{x,q_{i+1}}[\mathbf{R}_{i+1}^0\mathbf{n}_{i+1} + (\mathbf{R}_{x,q_{i+1}}^T(\mathbf{R}_i^0\mathbf{p}_i{}^*)) \times (\mathbf{R}_{i+1}^0\mathbf{f}_{i+1})] + \\ (\mathbf{R}_i^0\mathbf{p}_i{}^* + \mathbf{R}_i^0\hat{\mathbf{s}}_i) \times (\mathbf{R}_i^0\mathbf{F}_i) + \mathbf{R}_i^0\mathbf{N}_i & \text{, if i is even} \end{cases} \tag{3.47}$$

$$\tau_i = \begin{cases} (\mathbf{R}_i^0\mathbf{n}_i)^T(\mathbf{R}_{x,q_i}\mathbf{x}_0) + b_i\dot{q}_i + k_iq_i & \text{, if i is odd,} \\ (\mathbf{R}_i^0\mathbf{n}_i)^T(\mathbf{R}_{y,q_i}\mathbf{y}_0) + b_i\dot{q}_i + k_iq_i & \text{, if i is even,} \end{cases} \tag{3.48}$$

3.2.3.1 Newton-Euler Recursive Algorithm

The Newton-Euler recursive dynamic model uses equations 3.38-3.48 and follows the steps below in order to calculate both the linear and rotational motion of the robot and the input torques driving the motion.

1. Set constants and initial conditions:

   n_j = 2n (number of joints, 2 per segment)
   $\mathbf{R}_0\omega_0 = \mathbf{R}_0\dot{\omega}_0 = \mathbf{0}$
   $\mathbf{R}_0\mathbf{v}_0 = \mathbf{0}$
   $\mathbf{R}_0\dot{\mathbf{v}}_0 = [0 \quad 0 \quad g]^T$ (linear acceleration of base towards earth)
2. Joint variables are q_i, $\dot{q}_i$, and $\ddot{q}_i$ for i=1,2,...,n_j
3. Link-wise variables are i, $\mathbf{F}_i, \mathbf{N}_i, \mathbf{f}_i, \mathbf{n}_i$, and τ_i
4. For no load at end-effector, initialize $\mathbf{R}_{i+1}^0\mathbf{f}_{i+1} = \mathbf{0}$ and $\mathbf{R}_{i+1}^0\mathbf{n}_{i+1} = \mathbf{0}$.
5. Execute Algorithm 1.

Algorithm 1 Newton-Euler Recursive Dynamics

```
1: Initialize $i = 1$
2: Compute $\mathbf{R}_i^0\omega_i$, $\mathbf{R}_i^0\dot{\omega}_i$, $\mathbf{R}_i^0 v_i$, $\mathbf{R}_i^0\dot{v}_i$ by Eq. (3.38), (3.39), (3.40), and (3.41).
3: Compute $\mathbf{R}_i^0\dot{\hat{v}}_i$ by Eq. (3.43).
4: Compute $\mathbf{R}_i^0\mathbf{F}_i$ and $\mathbf{R}_i^0\mathbf{N}_i$ by Eq. (3.44) and (3.45).
5: if i=$n_j$, continue. Otherwise, set i=i+1 and return to Step 2.
6: Compute $\mathbf{R}_i^0\mathbf{f}_i$ and $\mathbf{R}_i^0\mathbf{n}_i$ by Eq. (3.46) and (3.47).
7: Compute $\tau_i$ by Eq. (3.48)
8: If i=1, stop. Otherwise, set i=i-1 and return to Step 5.
```

3.3 State Estimation

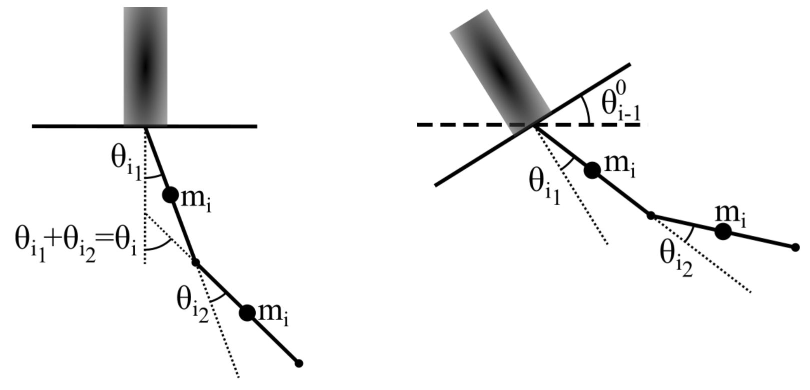
In order to approximate the missing state information (i.e. the distribution of curvature within a given section), we employ the relatively common method from finite element analysis of determining the minimum energy state of the system that satisfies the boundary conditions of the state. Before proceeding, we establish the following assumptions:

1. For a given continuum section, all curvature occurs within a single plane (i.e. the system does not experience torsion along the backbone), where the plane of bending is coincident with the value ϕ for a given section.

2. Mass is uniformly distributed along each continuum section, resulting in two equal cylinders with the CoM located halfway along the center length line of each cylinder.

3. Materials properties such as spring stiffness and damping coefficients are uniform within a given section.

The basic case, depicted in Figure 3.5, exhibits a single continuum section, approximated as two rigid bodies with single-axis revolute joints forming the connection between the bodies and between the proximal segment and the anchor point. In visualizing the potential energy of this system (and in considering the energy equations proposed in the Euler-Lagrange model) it is fairly simple and reasonable to state that the potential of the system is composed of the potential due to gravity and the energy stored due to the torsion springs. Thus, the total potential energy can be



(a) Single approximate continuum section at equilibrium

(b) Rotation at base of single section

Figure 3.5: Single continuum section approximated via two segments of equal length

shown to be:

$$\Pi = \frac{1}{2}k\theta_{i1}^2 + \frac{1}{2}k\theta_{i2}^2 + \frac{mgL}{2}\Big[1 - \cos(\theta_{i1})\Big] + mgL\Big[1 - \cos(\theta_{i1})\Big] + \frac{mgL}{2}\Big[\cos(\theta_{i1}) - \cos(\theta_{i1} + \theta_{i2})\Big]$$

$$= \frac{1}{2}k\theta_{i1}^2 + \frac{1}{2}k\theta_{i2}^2 + \frac{3mgL}{2}\Big[1 - \cos(\theta_{i1})\Big] + \frac{mgL}{2}\Big[\cos(\theta_{i1}) - \cos(\theta_{i1} + \theta_{i2})\Big]$$

where θ_{i1} and θ_{i2} are the bending magnitudes of the first and second joint, respectively, and the total rotation of the section's end-point is $\theta_i = \theta_{i1} + \theta_{i2}$. The value Π is the total potential energy of the system, with components P_e and P_g being the potential energy due to spring displacement and the potential energy due to gravity, respectively, as with the Euler-Lagrange Model.

In considering more complex planar examples, where we might have multiple sections, we can expand the potential energy equation to a generalized form for $\frac{n}{2}$ continuum sections, approximating by a total of n continuum segments:

$$\Pi = P_e + P_g \tag{3.49}$$

$$P_e = \frac{1}{2}\sum_{i=1}^{n} k_i \theta_i^2 \tag{3.50}$$

$$P_g = \sum_{i=1}^{n} \frac{(2(n-i)+1)m_i g L_i}{2}\left[\cos\left(\sum_{j=0}^{i-1}\theta_j\right) - \cos\left(\sum_{j=0}^{i}\theta_j\right)\right] \tag{3.51}$$

where k_i is the spring stiffness of each section, m_i is the mass of segment i, and L_i is the chord length of the segment. This form also provides the ability to adjust for rotations at the base of the robot as depicted in Figure 3.5b.

When leaving the plane, the equation for the gravitational potential energy alters slightly for non-distal sections. Given a two section, four segment approximation (n=4), the total potential energy for the proximal section is:

$$P_{g_i} = \sum_{i=1}^{2} \frac{(2(n-i)+1)m_i g L_i}{2}\left[\cos\left(\sum_{j=0}^{i-1}\theta_j\right) - \cos\left(\sum_{j=0}^{i}\theta_j\right)\right] + (e_z^T \mathbf{p}_{i+1} - e_z^T \mathbf{R}_{y,\theta_i}\mathbf{R}_{z,\Delta\phi}\mathbf{p}_{i+1}) \tag{3.52}$$

where $e_z = [0,0,1]^T$ and $R_{y,\theta}$ and $R_{z,\Delta\phi}$ are the base rotation matrices about the y and z-axis, respectively. The vector $\mathbf{p}_{i+1}$ is the location of the CoM of the distal segment in the segment's local coordinate frame. The new terms represent the projection of distal sections onto the plane of bending of section i. When considering travel in the opposite direction to determine the amount of rotation occurring before a given section, it is useful to break down the magnitude of a section's curvature into two values:

$$\theta_{i-1}^0 = [\theta_{i-1}^N, \theta_{i-1}^*] \tag{3.53}$$

$$= [\theta_{i-1}^0 \sin(\phi_{i-1} - \phi_i), \theta_{i-1}^0 \cos(\phi_{i-1} - \phi_i)]. \tag{3.54}$$

The value θ_{i-1}^* represents the curvature along the direction of θ_i of the proximal sections and θ_{i-1}^N represents the curvature component that is normal to the plane of bending for section i, as related by the difference between ϕ_i and ϕ_{i-1}.

The total potential energy for distal sections is impacted by the configuration of the proximal sections using this definition of θ_{i-1}^N and θ_{i-1}^0:

$$\Pi = P_e + \cos(\theta_{i-1}^N)P_g \tag{3.55}$$

$$= P_e + \cos(\theta_{i-1}^0 \sin(\phi_{i-1} - \phi_i))P_g \tag{3.56}$$

The intuition of this result is that when $\|\theta_{i-1}^N\| = \frac{\pi}{2}$, the distal section is parallel to the ground and the potential energy due to gravity is eliminated. This results in the total potential energy only comprising of the energy stored in the virtual springs, which at its minimum would return the result $\theta_{i1} = \theta_{i2}$, or exhibiting constant curvature.

3.3.1 Finding Minimized State

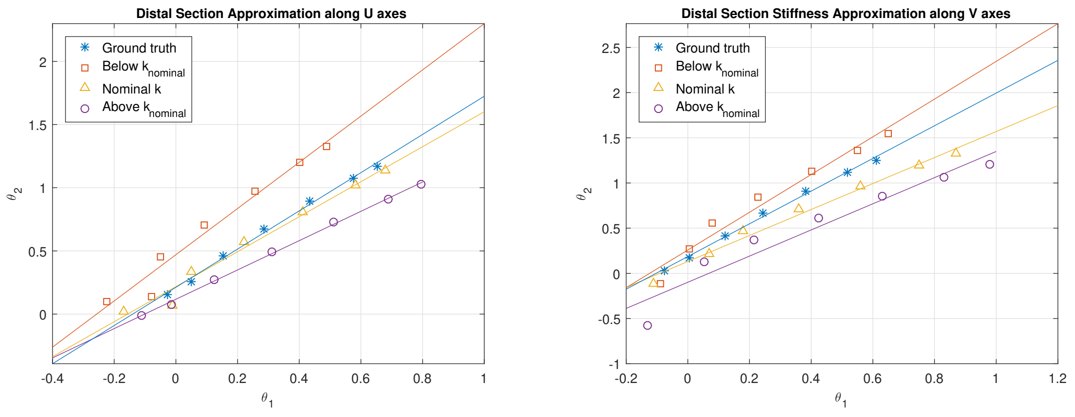
Once we obtain the expression for determining the systems total potential energy in 3.55, we can iteratively solve for the angles θ_{i1} and θ_{i2}, working from the distal section to the proximal section, by finding the angles that minimize the energy in a given section. This is possible given the assumption that we can measure the total bend present in a section at a given time (either through string encoders or other sensors that provide relative orientation of a section's end-point to its base frame). Thus, we are merely minimizing the distribution of curvature in a section, using known information (total bend in the section, material properties, and orientation of the section).

The actual minimization was carried out herein via the $fminsearch$ in MATLAB, a function from the Optimization Toolbox which is a derivative free method for minimizing single or multi-variable functions. We seed the minimizer with a guess at the minimum state, in this case the guess is based on constant curvature. Since we know the total bend of a section, measured as θ_i, and that $\theta_{i1} + \theta_{i2} = \theta_i$, we can rewrite the equations to be optimized for each section as:

$$\Pi = \frac{1}{2}k_i[\theta_{i1}^2 + (\theta_i - \theta_{i1})^2] + P_g \tag{3.57}$$

3.3.2 Validating Non-constant Curvature Approximation

We verify the minimization by comparing the results from "fminsearch" to external measurements of orientation along a section. External measurements are provided by a series of 3 inertia measurement units (IMUs) placed along the length of a section. The first sensor is placed at the base of the section in order to relate the base orientation. The remaining sensors are placed at the mid-point and end-point of the section, to measure the midpoint orientation and end-point orientation relative to the section's base. It is worth noting that as with the potential energy optimization, this method is only truly reflective of the state when the section is stationary and acceleration measured by the IMU is merely the distribution of acceleration due to gravity across the 3 Cartesian axes. For verification, we sample a series of static configurations of the continuum robot, taking first measurements and approximations of the distal section, and then measurements and approximations of the middle section. Figures 3.6 and 3.7 plot the results of these measurements. As we can see from both sets of data, there is a strong correlation between the trend of the results returned by both the approximation and the measurements returned by the accelerometers. Where we see the largest divergence of the methods is around the time that the sections are under-actuated, and thus able to have "S" shaped curves, where the base of the section bends one direction due to gravity and the later portion of the section bends in the opposite direction due to applied pressures.



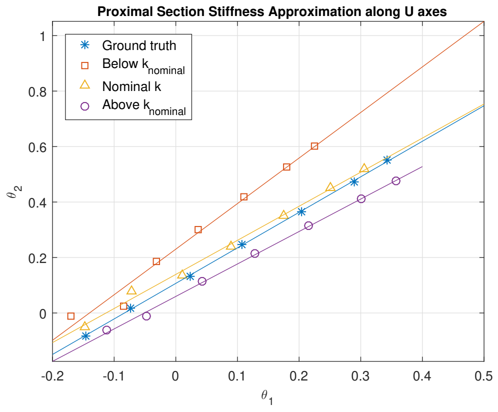
(a) Kinematic values u(t) and v(t) during rotation of distal section.

(b) Generated forces corresponding to bending of distal section with known planar obstacle.

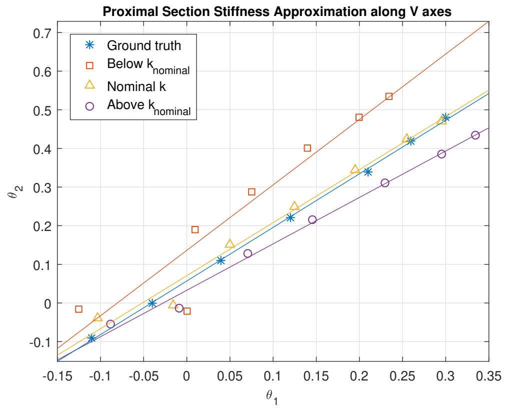
Figure 3.6: Evaluating state estimation of proximal section and impact of stiffness on curvature distribution.

3.4 Experimental Evaluation of Piece-Wise Model

In order to model the OctArm as a non-extensible continuum robot, each of its sections is initialized to halfway between its maximum and minimum length. Due to imperfections in the OctArm's construction, this also results in the OctArm not starting out "perfectly straight" in the



(a) Kinematic values u(t) and v(t) during rotation of distal section.



(b) Generated forces corresponding to bending of distal section with known planar obstacle.

Figure 3.7: Evaluating state estimation of distal section and impact of stiffness on curvature distribution.

following experiments. In these experiments, since the OctArm is originally a three-section robot, the proximal section will refer to what is introduced as the mid section previously and the distal section will be the tip section of the OctArm. The base section of the OctArm is clamped and restricted from moving for the duration of these experiments. The list of model parameters used in the following simulations are summarized in Table 3.1.

Table 3.1: Dynamic Model Simulation Parameters

Parameter	Proximal Section	Distal Section
Mass [kg]	0.60	0.25
Length [m]	0.19	0.19
Section Radius [m]	0.045	0.03
Stiffness k_u [N·m/rad]	2.4	1.13
Stiffness k_v [N·m/rad]	2.4	1.13
Damping b_u [N·m·s/rad]	0.6	0.15
Damping b_u [N·m·s/rad]	0.6	0.15

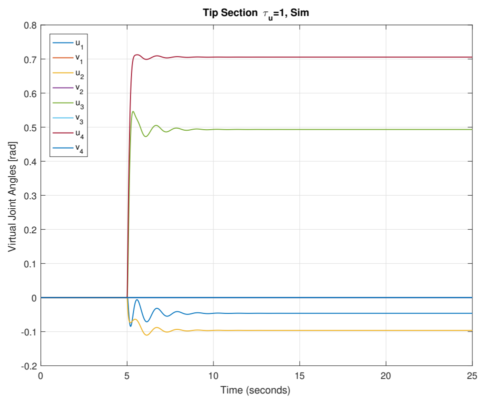
We appraise the efficacy of the model by analyzing a series of simulation results and experiments on physical hardware (the OctArm manipulator introduced in section 2.2), comparing the side-by-side reactions of the simulation and physical hardware to various inputs. Along with the following experimental descriptions below, a summary of the input signals for each experiment are given in Table 3.2, where the function $u_5(t)$ represents a heaviside step function with delay $t = 5$s.

The first experiment applied a heaviside step signal to the distal section to the u-corresponding joints with a magnitude of $\tau = 1$, and a start time of $t = 5$s. From the plots in Figure 3.8, it can

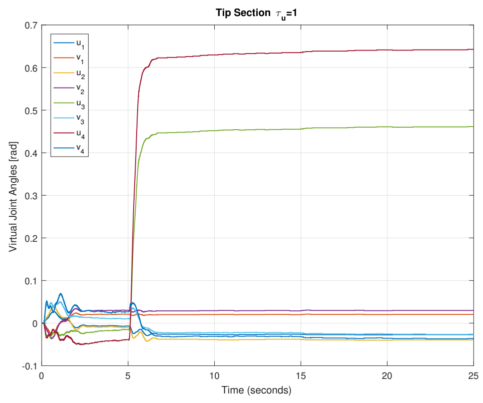
Table 3.2: Experiment Summary

Experiment	τ_{u_m}	τ_{v_m}	τ_{u_t}	τ_{v_t}
Distal Actuation	0	0	$u_5(t)$	0
Proximal Actuation (u-axes)	$u_5(t)$	0	0	0
Proximal Actuation (v-axes)	0	$u_5(t)$	0	0
Two-section Motion (v-axes)	0	$u_5(t)$	0	$u_5(t)$
S-bend (opposing v-axes)	$-2 \cdot u_5(t)$	0	$u_5(t)$	0
Orthogonal bending planes	0	$u_5(t)$	$u_5(t)$	0
Dynamic two-section motion	$2\sin(2t) \cdot u_5(t)$	$2\sin(2t + \pi/2) \cdot u_5(t)$	0	$u_5(t)$

be seen that the simulated model and the physical robot have very similar responses to the signal and settle to similar distributions of curvature. There is a slight drop in magnitude in observing the total bend of the physical robot against the simulation. The simulation model also predicts the reactionary bend in the proximal section that results from the CoM of the distal section moving significantly away from the initial Z-axis. The physical robot exhibits similar behavior, but again at a reduced magnitude compared to the simulation.



(a) Distal actuation in simulation

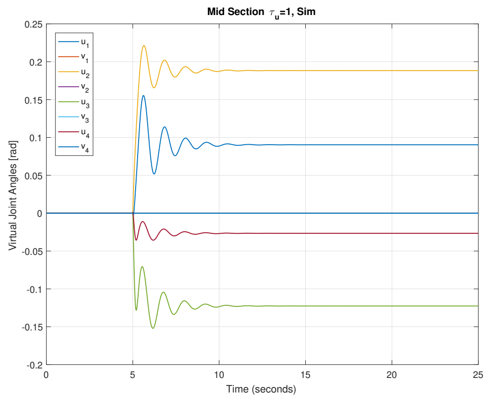


(b) Distal actuation on OctArm

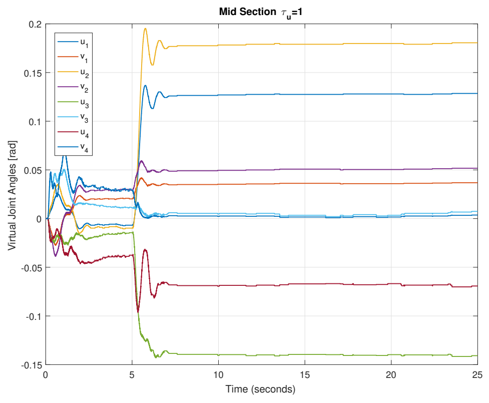
Figure 3.8: Experiment 1: A comparison of simulation model and physical OctArm robot with a unit step signal of magnitude $\tau = 1$ applied to U-designated joints in the distal section.

The second experiment applies the same signal from before, but now only to the proximal section's u-corresponding joints. Again we see a nice match between the reaction of the simulation and the physical robot. We also see a closer match in oscillations and magnitude of the settled state in this experiment, likely due in part to the better match in stiffness seen in 3.3.2.

The third experiment again applies the heaviside function to the proximal section, this time along the v-corresponding joints. In observing the simulation results, we see an identical reaction



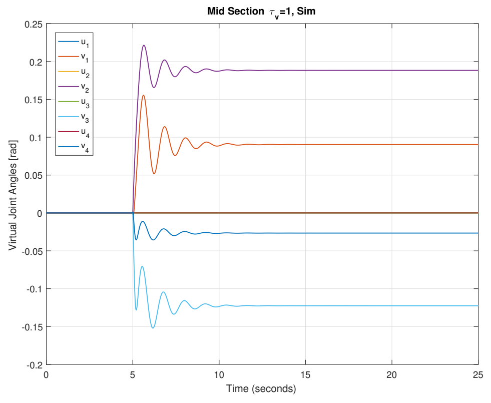
(a) Proximal actuation in simulation



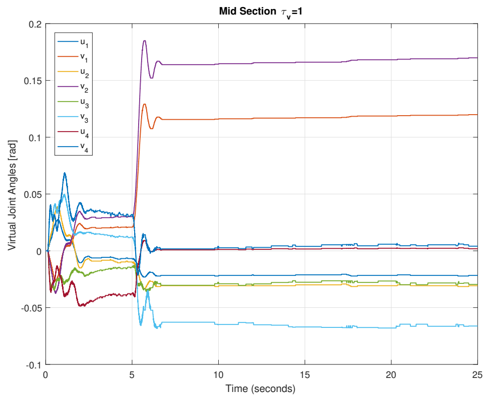
(b) Proximal actuation on OctArm

Figure 3.9: Experiment 2: A comparison of simulation model and physical OctArm robot with a value of $\tau = 1$ applied to U-designated joints in the proximal section.

to the previous simulation, albeit along the v-axes. In observing the response of the physical robot, we can observe a difference between this response and that seen in Figure 3.9b. This difference can partially be attributed to the fact that the OctArm is not symmetric about both axes of rotation and likely has a difference in stiffness between the u-axis and the v-axis.



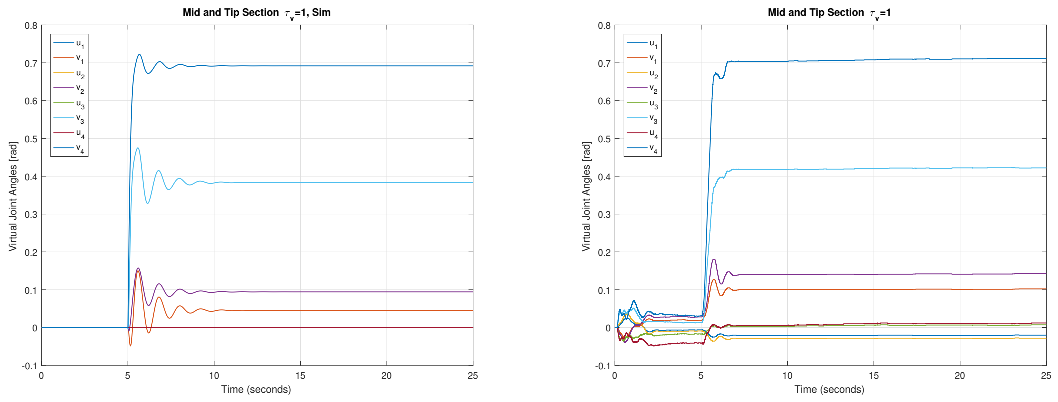
(a) Proximal section actuation along V-axes in simulation



(b) Proximal section actuation along V-axes on OctArm

Figure 3.10: Experiment 3: A comparison of simulation model and physical OctArm robot with a value of $\tau = 1$ applied to V-designated joints in the proximal section.

The remaining experiments actuate both sections simultaneously. In the first, a heaviside signal with magnitude $\tau = 1$ is applied to the v-axes of both sections, ideally leading to a planar curve. The results in Figure 3.11, again provide a good correlation between the simulation and physical system. While there is a slight variation in magnitudes, the model accurately predicts that the proximal section will exhibit significantly less curvature, despite receiving the same input, as it is supporting the mass of the distal section.



(a) Proximal and distal actuation of V-designated joints in simulation

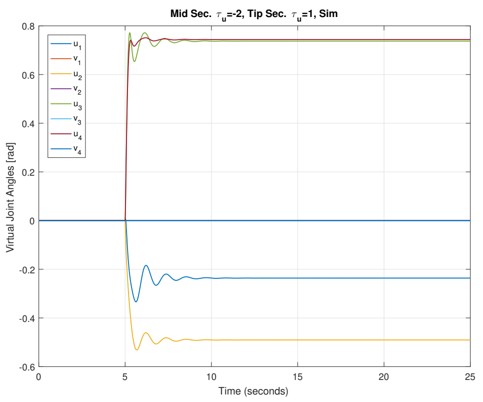
(b) Proximal and distal actuation of V-designated joints on OctArm

Figure 3.11: Experiment 4: A comparison of simulation model and physical OctArm robot with a value of $\tau = 1$ applied to V-designated joints in both the proximal and distal sections.

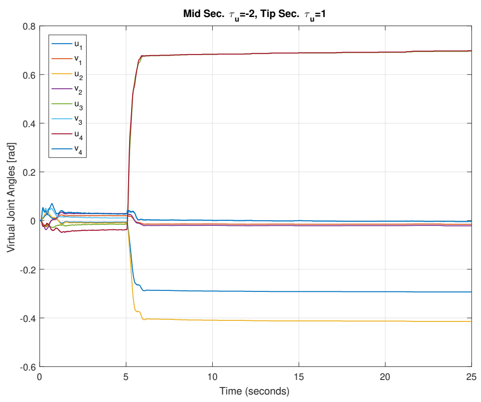
In experiment 5, we again exhibit planar bending, but this time with the two sections bending in opposing directions, giving an “S” curve. The inputs for this experiment were $\tau = -2$ for the proximal section u-axes and $\tau = 1$ for the distal section u-axes. Interestingly, as seen in Figure 3.12, both the physical robot and the model predicted near constant curvature in the distal section as a result of the initial orientation caused by the proximal section. The proximal section has a good match in overall bend between both systems, while the proximal section shows a small magnitude than predicted by the simulation.

The last experiment utilizing unit step functions in this series actuated both sections at orthogonal planes of bending, i.e. the proximal section is actuated along the v-axes and the distal section is actuated along the u-axes. The magnitude for both inputs is again $\tau = 1$. Again we see a good match between the response predicted in simulation and the physical system response, with a slight mismatch in magnitudes of the desired motion and in the reactionary displacements of the unactuated axes.

Finally, we look at the response of the model to a dynamic motion, highlighted in Figure 3.14. In this test, sinusoidal torques are applied to the proximal section axes with a phase shift of $\frac{\pi}{2}$ between the u-axes and the v-axes, inputs that should cause the proximal section to execute a

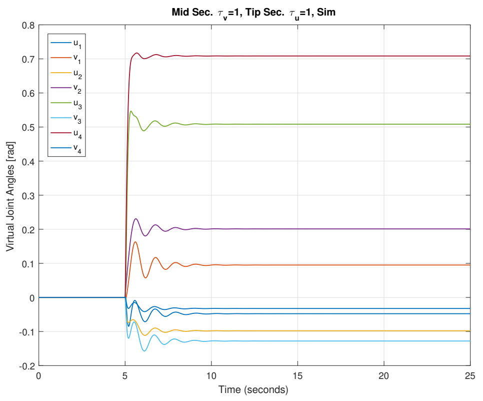


(a) Opposing "S" actuation of proximal and distal sections in simulation.

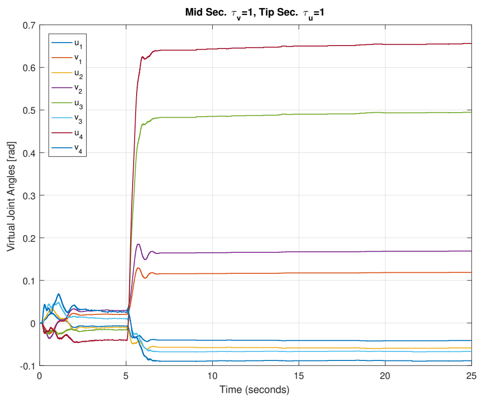


(b) Opposing "S" actuation of proximal and distal sections on OctArm.

Figure 3.12: Experiment 5: A comparison of the simulation model and physical OctArm robot with a value of $\tau = -2$ applied to U-designated joints in the proximal section and $\tau = 1$ to the U-designated joint in the distal section, causing two sections to bend in opposite directions in the same plane.



(a) Proximal and distal section actuation at orthogonal bending planes in simulation. Proximal section actuated about V-designated joints, and the distal section about U-designated joints.



(b) Proximal and distal section actuation at orthogonal bending planes on OctArm. Proximal section actuated about V-designated joints, and distal section about U-designated joints.

Figure 3.13: Experiment 6: A comparison of simulation model and physical OctArm robot with a value of $\tau = 1$ applied to V-designated joints in the proximal section and $\tau = 1$ to U-designated joints in the distal section.

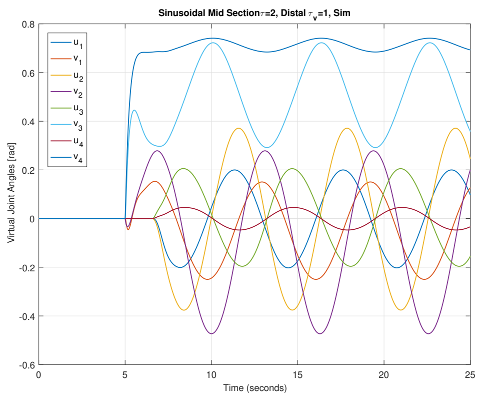
circular motion with its end-point. The distal section is actuated using a heaviside step function as before, applied to the v-axes with magnitude $\tau = 1$. The result of this experiment provides numerous insights while still exhibiting many of the behaviors from the previous experiments. As with the previous experiments, we observe a nice match in overall behavior between the idealized model in simulation and the physical robot's state estimation.

One insight gained from this experiment was the observation of hysteresis in the proximal section about the $v$axes. In simulation, we see a vertical shift in the sinusoid corresponding to proximal v-joint values (v_1 and v_2) relative to the values of the u-axes. This was expected as the proximal section rotated into and out of phase with the distal section. However, in the physical robot's response, we see this shift exists when v of the proximal section is positive (i.e. bending in phase with the distal section), but does not exist when the proximal section is bending in the opposite direction of the distal section (when v_1 and v_2 are negative). This suggests that the stiffness of the physical robot changes between rotating in the positive v direction and rotating in the negative v direction, a conclusion that is supported by the fact that the physical construction of the OctArm has two muscles in tandem opposing a single muscle when bending about the virtual v-axes.

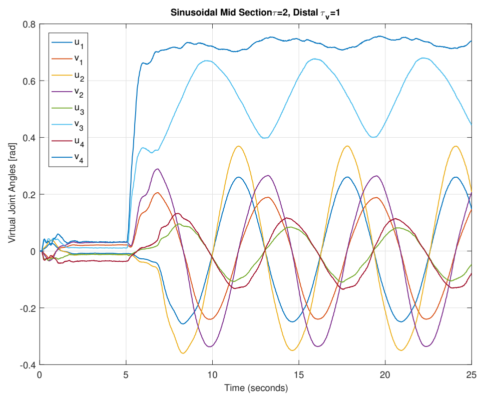
Another insight this experiment provided concerns the mismatch of the distal section's u-axes, which are not actively actuated but are able to deflect due to internal loads and changing orientation with respect to gravity. In the simulation, we observed these values to exhibit beam mechanic tendencies, in which the proximal joint of the section deflects at greater magnitudes. In the physical robot, we observe the opposite effect, where the value u_4, corresponding to the distal u-related axis of the distal section reports, greater rotation than the joint value u_3. We concluded that this mismatch in distal u value trends comes from the state estimator for the physical robot not allowing for non-planar bending, meaning that u and v values within a section must trend in the same way (i.e. increasing or decreasing along a section's length). In this experiment, the actuation of the v-related joints outweighs the beam related deflections predicted in simulation.

3.5 Discussion

The initial results of the dynamic model in showed an excellent match between the simulation of the robot dynamics and the physical device. The largest deviations seen were in the overall bending magnitude of each section and the amount of oscillation in the simulation about the static equilibrium. In addressing the difference in oscillations, it is possible to experimentally adjust the damping coefficients in the model to better match the physical system. An example of this is shown in Figure 3.15, where the simulation from experiment 5 is performed again after increasing the damping coefficient for both sections. This produces a much more accurate approximation of the



(a) Simulation of executing a circular motion via the proximal section using sinusoidal torque inputs.



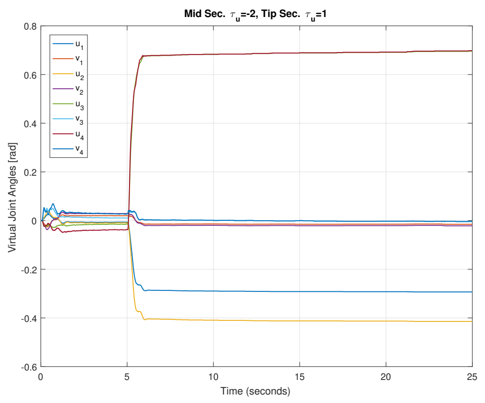
(b) State estimation of physical robot while executing circular motion of proximal section.

Figure 3.14: Experiment 7: Circular motion of the proximal section through sinusoidal torques applied to the proximal continuum section. The distal section is actuated with a value of $\tau = 1$ applied to V-designated joints.

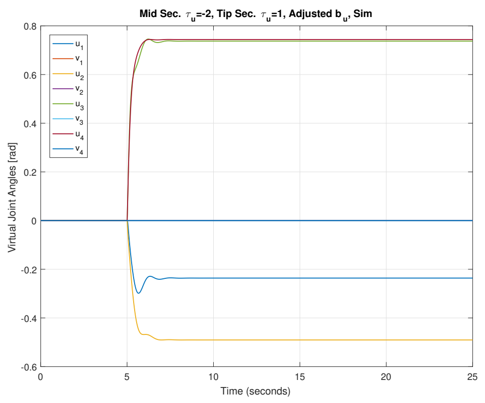
oscillations in the physical robot.

In considering the slight difference in magnitude between simulation and the physical system, there are a couple of factors that could be responsible. The first would be an imperfect tuning between what a value of $\tau = 1$ means in the simulation versus on the physical robot. In the physical system, τ is applied differentially to each muscle as designated by the kinematics, specifically the relationships between the actuator lengths and the kinematic parameters as in equations 2.6–2.8. Thus, τ_u and τ_v are distributed to the OctArm's muscle according to the relationship between l_1, l_2, l_3 and u, v when arc-length s is held constant. The τ applied to the physical robot is scaled by a single constant for each section and each direction of bending. These constants were found experimentally by actuating the simulation model with a known τ and then adjusting the scalar multiplier for the physical robot until the shape of the physical robot matches closely with the simulation. In adopting this approach, it is possible that the scalar needs to vary with the state of the robot or possibly that the estimation of this scalar should be based on several samples of τ to the physical system and simulation.

Another possible cause for the difference in the magnitude in bending is unmodeled dissipating forces such as friction between the OctArm's muscles and the outer mesh that constrict their motion. One solution could be to introduce model elements that address the hysteresis of pneumatic muscle braiding and observe if this improves the predicted magnitude of bending. There are already



(a) Opposing "S" actuation of proximal and distal sections on OctArm.



(b) Adjusted damping coefficient in simulation to better match with the physical robot response.

Figure 3.15: Comparison of experiment 5 after adjusting the damping coefficient for both the proximal and distal sections.

works in the literature that explore such models, [127,128], that could potentially address this error.

In considering the high number of approximated parameters in this model and the abstract meaning of many of the parameters, it is almost certain to contain inaccurate model parameters. A desirable result would be to minimize the error in parameter estimation and reduce the number of "guesses" made about parameters. While it is unlikely to handle the high number of unknown parameters, adaptive control could be a good candidate for estimating the mass of the robot during execution, which can then adjust the relevant parameters such as stiffness in order to maintain the previously established relationship between these values that produces a good match between simulation and physical device.

We did not show extensive study of the model's performance during highly dynamic motions, but we can see from the initial results that the simulation model predicts greater reactionary motion in the proximal section whenever the distal section is actuating. The design of a controller that can cancel out the reaction of the proximal section to motion in the distal section, i.e. reduce the impact of the dynamics, could be very useful when testing more dynamics motions.

In considering the state estimation provided in section 3.3, the method has been shown to be adequate for slow moving motions and when settled in static configurations, but at higher velocities it is unknown how the estimate will behave, though we can be certain that the estimate will lose accuracy. This problem could benefit from the application of filtering methods to relate state velocities and momentum to actual distribution of curvature. Alternatively, sensor fusion methods could prove helpful in increasing the accuracy of state estimates during high velocity motions. An example scenario could be a camera system providing intermediate information with high accuracy and the string encoders providing fast but low accuracy information. The IMUs used to calibrate the state estimator could also be used in a sensor fusion methodology in order to capture the rotational velocities of points along the robot's length.

Chapter 4

Promoting Automated Tasks for Continuum Robots

The following chapter presents two separate contributions towards the automation of continuum robots. The first presents the application of a reinforcement learning method in order to optimize a continuum robot's task to search its workspace for points of interest. The novelty of this effort is to automate continuum robot navigation in space through aliased states and without *a priori* knowledge of its environment, desired state, or optimal path. The second contribution automates the motion of a continuum robot's end-effector through its workspace without knowledge of the robot's kinematics or dynamics, providing a model free approach that also requires very little sensor information. Additionally, the method is able to achieve accuracy on the order of the state-of-the-art kinematic and dynamic models for continuum robots. Discussion for both contributions is given at the conclusion of the chapter

4.1 Policy Optimization for Search Tasks

A model-free policy optimization method is proposed in this work in order to shape a continuum robot's search policy for objects of interest. This method allows us to develop a global policy quickly while removing potential problems that redundancy and aliased states cause for deterministic methods.

4.1.1 Formulation

We formulate the control problem of a continuum robot section as a discounted Markov Decision Process [129] defined by the tuple $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, P, r, \gamma\}$, where $\mathcal{S}$ denotes the state space, $\mathcal{A}$ is the action space available to the robot, $P : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ is the state transition function, $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the reward that the environment emits on each transition, and $\gamma \in [0, 1]$ is the discount factor. Regarding $\mathcal{S}$, for a single continuum section, we define two DoF: κ, the curvature of the continuum section, and ϕ, the plane of bending for each section. Traditionally, many continuum robots also have the ability to extend and retract along their backbone; we assume fixed backbone length in this work. We create a discrete set of all states by describing each DoF as spanning $n_\kappa \in \mathbb{N}$ and $n_\phi \in \mathbb{N}$ discrete values, distributed evenly over a defined range for each value. Given i number of sections, the total number of states is $(n_\kappa \times n_\phi)^i$, giving $\mathcal{S}_{n_\kappa \times n_\phi \times \ldots}$. In this work, we limit our actions to a single step transition along one DoF at a time in order to simplify the list of available actions and create simple connections between states. For each DoF, we increment the DoF up ($a = +1$) or down ($a = -1$) along their discrete set of values, or remain at the current value ($a = 0$). Given i sections as before, the total number of actions is $3 \times i \times n_{DoF}$, where n_{DoF} is the number of DoF available to the section.

As is common among reinforcement learning methods, we empirically define a series of rewards and penalties associated with actions taken by the robot in order to shape the final policy. The reward function is primarily designed to promote actions that lead to a state that can view the goal object. Equally, it penalizes actions that leave such states in order to return to states that are not able to see the goal. We design the largest penalty to occur when a chosen action leads to an invalid state or invalid state transition. Examples of this would be trying to bend a continuum section beyond the physical limits of the robot or attempting to transition to a neighboring state that is blocked by a physical object. For all other actions, the algorithm issues a step penalty in order to encourage reaching the goal state in a finite number of steps. Thus, the reward structure,

R_s^a we employ is as follows:

$$R_s^a = \begin{cases} -100 & \text{, if } s' \text{ is invalid} \\ 5 & \text{, new state sees goal} \\ -5 & \text{, leaves state that sees goal} \\ -0.05 & \text{, general movement cost} \end{cases} \tag{4.1}$$

4.1.2 Actor-Critic Policy Optimization

We assume a model-free reinforcement learning setup, where the robot does not have direct knowledge about the transition function, P, and reward, r, and can only experience them through interacting with the environment. In particular, at a given time step t, the robot observes the current state $\mathbf{s}_t \in \mathcal{S}$ and samples an action $\mathbf{a}_t \in \mathcal{A}$ from a policy $\pi : \mathcal{S} \to \mathcal{A}$. This leads to a new state $\mathbf{s}_{t+1}$ that rewards the robot with r_t. Our goal is to solve for the policy that optimizes the robot's expected sum of discounted rewards.

Policy gradient methods allow us to maximize the expected cumulative reward by directly searching in the policy space, reducing the amount of memory needed to store quality of states and actions information as with Value Iteration and Q-learning methods, while they are also the preferred class of methods for learning controls in continuous state-action spaces. Here, we consider parameterized policies $\pi_\theta(\mathbf{a}|\mathbf{s})$ and hence the objective of the learning process is to find the parameters θ that maximize

$$J(\theta) = \mathbb{E}_{\mathcal{M},\pi_\theta}[\sum_{t=0}^{\infty} \gamma^t r_t | \pi_\theta] \tag{4.2}$$

where $\mathbb{E}_{\mathcal{M},\pi_\theta}[\cdot]$ denotes the expected value of the Markov Decision process for a given policy π_θ.

Given the above objective function, $J(\theta)$, we adjust θ through gradient ascent where the gradient of the expected reward can be determined according to the policy gradient theorem [130]:

$$\begin{aligned} &\nabla_\theta J(\theta) = \\ &\mathbb{E}_{\mathbf{a}_t \sim \pi_\theta(\cdot|s_t)} \left[\sum_t \nabla_\theta \log \pi_\theta(\mathbf{a}_t|\mathbf{s}_t) Q^{\pi_\theta}(\mathbf{s}, \mathbf{a}) \,|\, \mathbf{s}_t \right] \end{aligned} \tag{4.3}$$

where $Q(\mathbf{s}_t, \mathbf{a}_t) = \mathbb{E}_{a \sim \pi(\cdot|s_t),\mathcal{M}} \left[\sum_{l=0} r_{t+l}\right]$ denotes the action-value Q function. To reduce the variance of the policy gradient estimate and increase stability, we consider an actor-critic policy gradient framework [130]. In particular, we replace the estimate of the Q-value provided by the cumulative reward in Eq. 4.3 with a function approximator (critic) which is learned in tandem with the policy (actor). The critic evaluates the quality of the policy for a current set of policy parameters. The actor then shapes the policy parameters in response to the output from the critic. It is important to note that the vector value θ is of the same dimension as our state-action feature vector, which we define when implementing the solution in Section 4.1.3.

In the problem we are exploring in this work, our continuum manipulator is capable of assuming a discrete number of states and to perform a discrete set of actions in order to transition between these states. As such, we have chosen to use the Softmax policy [130] which states that the probability of an action is proportional to the exponential of a linear combination of features $\Phi(s,a)$:

$$\pi_\theta(s,a) \quad \propto \quad e^{\Phi(s,a)^T\theta} \tag{4.4}$$

Using this policy, our learned policy parameters θ are defined to be coefficients for each of our features.

Given the well-defined nature of the Softmax policy, the relevant score function is:

$$\nabla_\theta \log \pi_\theta(s,a) = \Phi(s,a) - \mathbb{E}[\Phi(s,\cdot)] \tag{4.5}$$

where $\mathbb{E}[\Phi(s,\cdot)]$ is the expected feature vector at state s.

Regarding the critic that evaluates our policy, we consider a linear approximation of the value function, $Q^\pi(s,a)$, by linearly combining the features via a weight vector w:

$$Q_w(s,a) = \Phi(s,a)^T w \approx Q^{\pi_\theta}(s,a) \tag{4.6}$$

The critic is updated at each time step using linear Temporal Difference (TD) learning that adjusts the parameters w of the Q-function based on the TD error, *delta*, and the state-action features. We refer the reader to Algorithm 1 for an overview of our actor-critic framework for learning an optimal policy. Here, each learning iteration generates sample(s) from the current policy, uses these samples to update the critic function, and updates the policy parameters based on the critic and the gradient of the objective function. Learning rates α and β adjust step size for learned parameters, and the discount factor γ determines the impact of future rewards.

Algorithm 2 Actor-Critic Policy Gradient

```
1: function QAC
2:     Initialize $s, \theta$
3:     Sample a $\sim \pi_\theta$
4:     for each step do
5:         Sample reward $r = R_s^a$, get transition $s' \sim P_s^a$
6:         Sample action $a' \sim \pi_\theta(s', a')$
7:         $\delta = r + \gamma Q_w(s', a') - Q_w(s, a)$
8:         $\theta = \theta + \alpha \nabla_\theta \log \pi_\theta(s, a) Q_w(s, a)$
9:         $w \leftarrow w + \beta \delta \Phi(s, a)$
10:        $a \leftarrow a', s \leftarrow s'$
11:    end for
12: end function
```

4.1.3 Simulation Validation

We verify the functionality of the algorithm using a simulation model of the Tendril robot [131] placed in the Gazebo physics simulator environment [122]. The Tendril is a continuum robot comprising of a backbone made using a carbon fiber tube, plastic spacers for the routing actuating tendons, and an actuator package that pulls on the tendons to create bending. The physical Tendril is long and thin, with either 2 or 3 independent continuum sections. The Tendril is simulated in Gazebo using a series of small, rigid linkages connected in series that approximate the continuum shape of the actual Tendril. The end-effector of the simulated manipulator is fitted with a camera that is oriented in line with the tendril's backbone, much like an endoscope. In order to simplify the image processing task and focus on the policy optimization, the Tendril is placed in an empty simulation world with a single object that represents the goal we wish to locate with the robot's camera. An example of the empty world scenario and the viewpoint of the simulated Tendril is given in Figure 4.1, where the view of the Tendril is seen in the lower left corner.

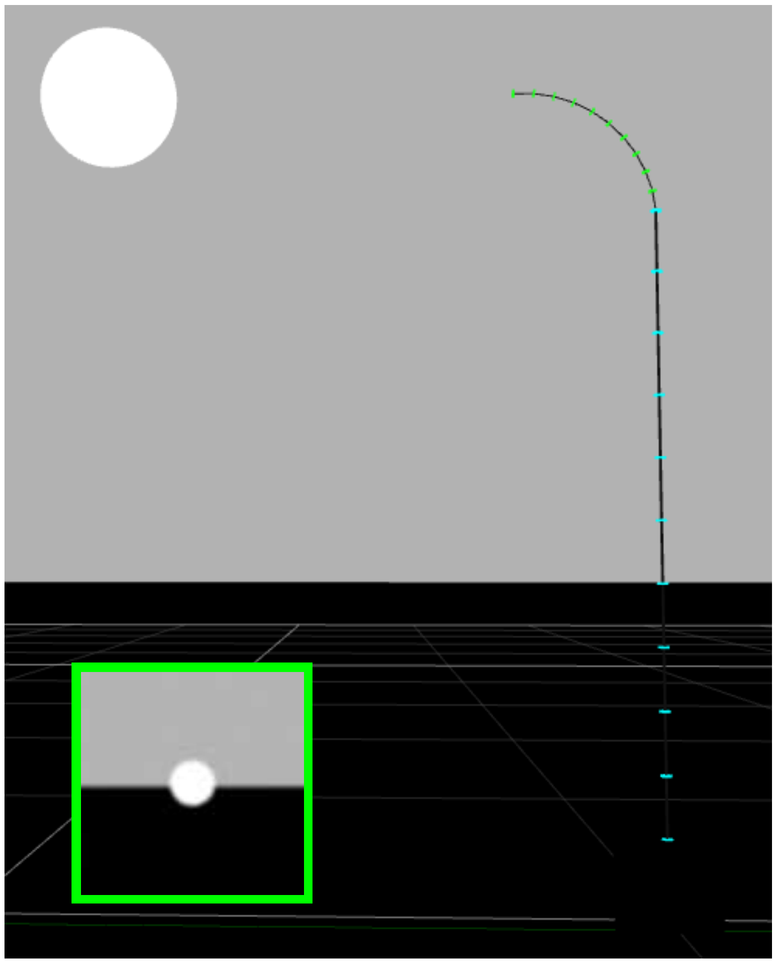


Figure 4.1: Empty World Simulation with Tendril Robot

4.1.4 Defining Features

Necessary to the implementation of our formulation is the definition of the feature vector. As this solution is designed for inspection purposes, our state feature vector $\Phi(s)$ is defined based on feedback from a camera and low-level image processing:

$$\Phi(s) = [I_{avg}, S_{obj}] \tag{4.7}$$

where I_{avg} is the average intensity of the image and S_{obj} indicates if a goal object is present and the size of the object relative to the camera frame size. Both feature values are normalized to the closed range $[0, 1]$, and the size of goal object is saturated to a threshold equivalent to occupying 1/10th the area of the camera view.

In order to simplify the execution of the method around edge states (i.e. the boundaries of our state space), we preserve the same action set for all states and instead apply an action-based feature, and corresponding penalty in our reward function, for state-action pairs that attempt to assume an invalid state. The feature, represented as B in equation 4.8, exists as a binary feature: 1 when the chosen action crosses a boundary (such as exceeding bending limits), and 0 when the chosen action leads to another valid state.

$$\Phi(s, a) = [\Delta I_{avg}, \Delta S_{obj}, B] \tag{4.8}$$

The remaining features in our state-action feature vector $(\Delta I_{avg}, \Delta S_{obj})$ are the changes in the state features I_{avg} and S_{obj}, respectively, between state s and the state s' reached upon performing action a.

In analyzing the algorithm given this feature definition, it can be seen that the nature of our state-action feature vector will always produce a non-zero probability of staying in a arbitrary state at any given time. In other words, $\Phi(s, a) = [0, 0, 0]$ when the action is $a = 0$ across all DoF, giving $e^{[0,0,0]^T\theta} = 1$. Therefore, we modify the policy for these actions in each state as:

$$\pi_\theta = S_{obj} e^{\Phi(s,a)^T\theta} \tag{4.9}$$

This modification to the policy removes the probability of choosing actions that stay in a state for any configuration that does not see the goal object and scales the probability of staying in a goal state according to how well the state "sees" the goal, as designated by S_{obj}.

4.1.5 Planar Task Space Exploration

In this first experiment, we start with a two-section continuum manipulator capable of independent planar bending in each section (i.e. $n_{DoF} = 1$). For clarity of visualizing the states, we indirectly provide curvature values (κ) as bending angles, which can then be converted to curvature for a fixed length backbone using $\kappa = \frac{\text{bending angle}}{\text{arc-length}}$. We allow the proximal section three bending values: bending angle of zero (straight backbone), and ±90°(i.e. bending left and right at 90°). Separately, we allow the distal section five values: straight, and bending angles of ±90°, ±180°. Therefore, the total number of possible states is 15. Examples of physical meaning for these states can be seen depicted in Figure 4.2. In evaluating the ability to locate an object of interest, we placed the singular goal object 0.8m left of the base of the robot and 1m vertically up from the base, which is conveyed by the blue orb in the upper left corner of each state image in Figure 4.2. Finally, for this experiment, we set the learning and discount rates to: $\alpha = 0.1$, $\beta = 0.3$, and $\gamma = 0.95$, which we experimentally found to work well for solution convergence.

We ran the algorithm 10 times while initialized at each of the possible configurations for a total of 150 executions. Each execution was allowed to run for 1500 iterations with the learning rates given above. The average of the policies obtained from each of the 150 runs is described in Table 4.1, where the numbers given per action per state are the percent chance that the action will be taken when in that state. The actions listed in the table (Up/Down/Left/Right) refer to the transitions seen in the visual interpretation of the final policy in Figure 4.2. For each depicted state, the arrows flowing from the state represent the possible action transitions and are shaded according to the likelihood of that action being taken and transitioning to a neighboring state. All action transitions appearing as grey have a probability of either zero or near zero (<0.5%) of being chosen. The three states capable of seeing the goal are highlighted in the figure (States 5, 9, and 12), and are the only states that contain action transitions indicating a probability of staying in the present state.

As can be seen in the results of our simplified example, in the states where the goal object is well seen (States 9 and 12), we see a greater chance of staying in those states. In states neighboring the goal states (i.e. one action step), we also observe a markedly high percent chance (>75%) of taking the action that get us directly to a goal state. In states more than a step away, we see nearly

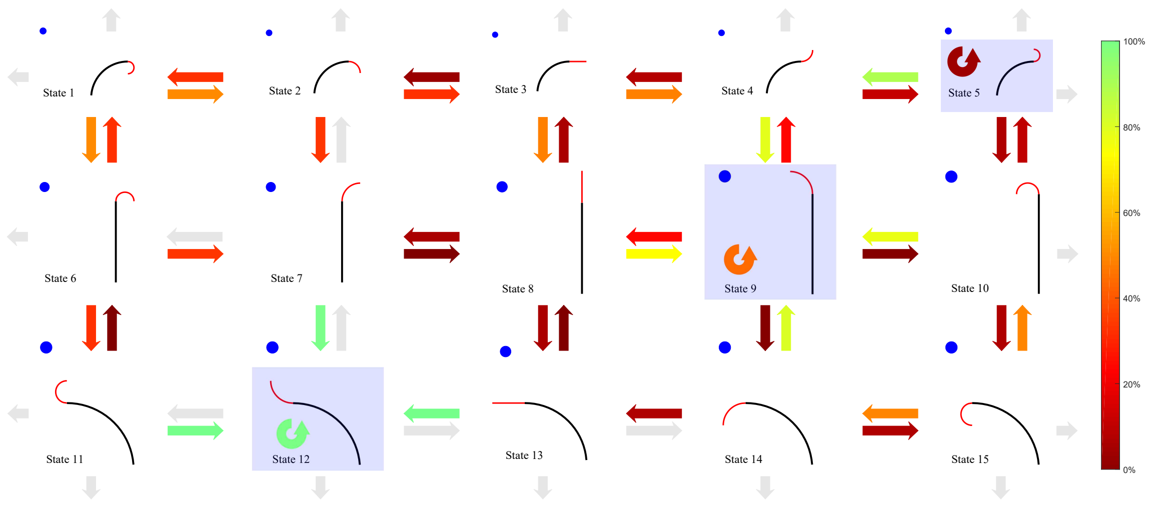


Figure 4.2: Optimized Search Policy for Planar Two-section Continuum Manipulator: Colors correspond to probability of state-action transition occurring. Grey transitions indicate probability of zero.

Table 4.1: State Action Probabilities According to Policy for Planar Search

Action	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15
Up	0.1	0.0	0.0	0.0	0.0	33.1	0.3	9.0	25.8	13.3	0.7	0.3	0.8	80.0	49.9
Stay	0.0	0.0	0.0	0.0	2.1	0.0	0.0	0.0	21.9	0.0	0.0	49.6	0.0	0.0	0.0
Down	49.9	33.7	47.5	77.0	8.5	33.2	98.8	9.0	2.3	9.6	0.0	0.0	0.0	0.0	0.1
Left	0.1	33.2	4.9	9.7	87.3	0.0	0.3	9.2	25.7	77.2	0.0	0.3	98.8	10.1	49.9
Stay	0.0	0.0	0.0	0.0	2.1	0.0	0.0	0.0	21.9	0.0	0.0	49.6	0.0	0.0	0.0
Right	49.9	33.1	47.6	13.2	0.0	33.7	0.7	72.7	2.3	0.0	99.3	0.3	0.4	9.9	0.1

uniform distribution among the valid actions, which is to be expected in an aliased state that does not provide much feedback to the system. Also, as designed in the feature set, in all edge states, any actions that lead to states beyond the limitations of the robot converge to zero or near zero percent chance of begin chosen. Overall, this is the expected optimal policy.

4.1.6 Spatial Task Space Exploration

We extend the above example by adding two additional DoF: direction of bending for the proximal section (ϕ_{prox}) and for the distal section (ϕ_{dis}), giving $n_{DoF} = 2$. The number of states quickly extends beyond the amount that can be reported here in detail. Instead, we report the total number of states, location of the goal, and the convergence to a stable policy. To start, we define our state set. In this experiment, we allow ϕ_{prox} and ϕ_{dis} to have 3 values: 0°, 120°, and 240°. We keep the same range of bending angles for the proximal and distal sections as the planar experiment. Given this, our total number of states is: $3 \cdot 3 \cdot 3 \cdot 5 = 135$. We can see a visual expression of these states in Figure 4.3, where we have also placed an example goal object at $[x, y, z] = [-1, 0.5, 1.25]$. We use the same learning rate α and discount value γ from before. We modify the learning rate β to be 0.2, which we found slightly improved performance on our hardware.

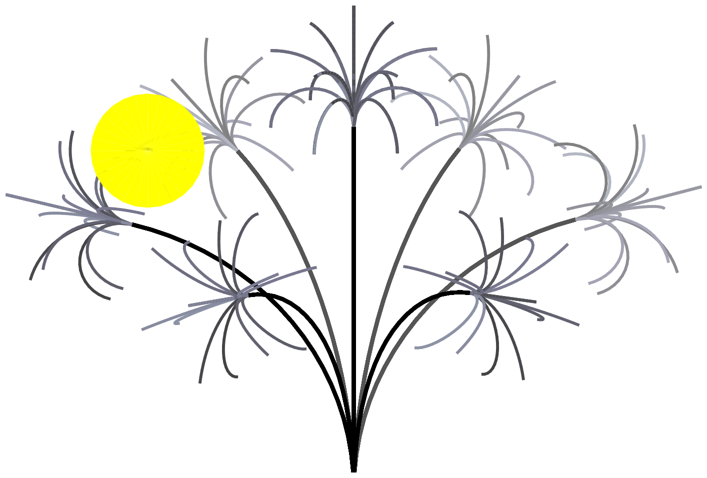


Figure 4.3: Task Space of Tendril in Open space

Given that our state space in this example is too large to reproduce visually here, we instead track the convergence of our policy parameters θ to a stable set of values. We ran the optimization algorithm 5 times from randomly selected starting states. Figure 4.4 depicts the average change in the three θ values over 2000 iterations of the policy optimization algorithm and includes the standard deviation of the 5 trials as shaded regions.

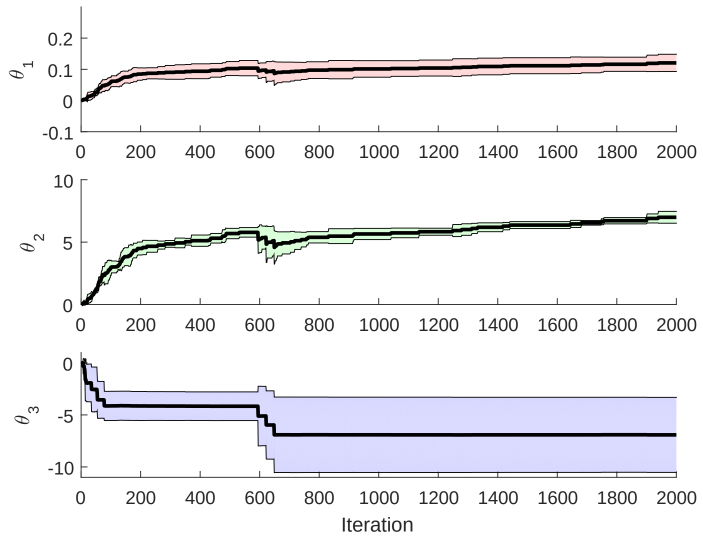


Figure 4.4: Convergence of Policy Parameters for 4 DoF (Solid lines denotes the average and shaded regions the standard deviation of policy parameter values over 5 trials.)

As can be seen, values θ_1 and θ_3 settle around approximately 800 iterations. The value of θ_2 increases slightly after this point, but generally begins to plateau enough to consider it a sufficient condition to exit the learning process. In practicality, we can design exit conditions (such as no change in policy for x iterations) to exit the learning process.

From our knowledge of the features that describe the state of our robot, we can draw conclusions from the relative magnitude and sign of the three θ values and their impact on our policy. It can be seen that any action that crosses a boundary (B=1), simulated or physical, will have a large negative component in the exponent, giving a probability approaching zero of that action being chosen. Even in the event that a goal state is on the other side of the boundary, the magnitude of the boundary associated parameter is generally higher than that of the goal. This is in part due to the penalty associated with crossing a boundary being significantly higher than the reward for reaching the goal.

In evaluating values θ_1 and θ_2, we can see a positive association with both the average intensity of the image (I_{avg}) and the feature indicating the size of the goal object (S_{obj}). Clearly, seeing the goal is more impactful on our policy as indicated by the difference in magnitude of the parameters. However, because our goal is a bright object in a dark environment, there is still a positive association between increased light intensity and reward for finding the goal. Seeing this behavior, we can potentially draw parallels to the policy slightly favoring an increase in brightness.

4.1.7 Impact of Learning Rates

Following our observations with the convergence of policy parameters in the previous experiment, we conducted an empirical study of the impact of varying the learning rates, α and β, on the solution. For each learning rate, we adjusted the values independently and averaged 10 samples at each of the selected test values. For varying α, these values were: 0.001, 0.01, 0.1, 0.5, and 1.0. For β, we tested β values: 0.002, 0.02, 0.2, 0.3, 0.6, and 1.0. When varying α, we set $\beta = 0.2$, and when varying β, $\alpha = 0.1$. Figures 4.5 and 4.6 show the result of varying α and β, respectively on all three of the policy parameters while using the same experimental setup as the previous experiment.

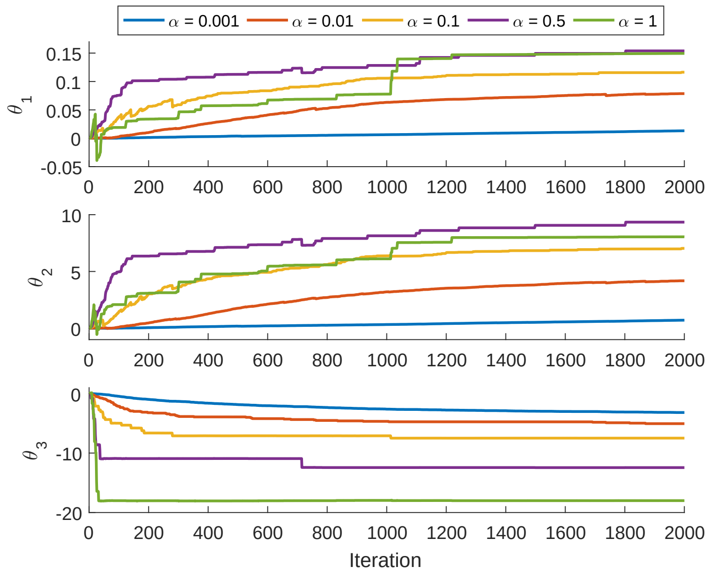


Figure 4.5: Impact of learning rate α on solution convergence ($\beta = 0.2$)

As can be seen in Figure 4.5, we generally see an increase in the rate of convergence as α increases, with some instability when $\alpha = 1$. With respect to the α value used in our experiments, $\alpha = 0.1$, we see that this choice of α shares desirable characteristics with both the smaller and larger α values. At α=0.1, the solution has a relatively smooth convergence, similar to the small α values, while having a faster rate of convergence like the higher values.

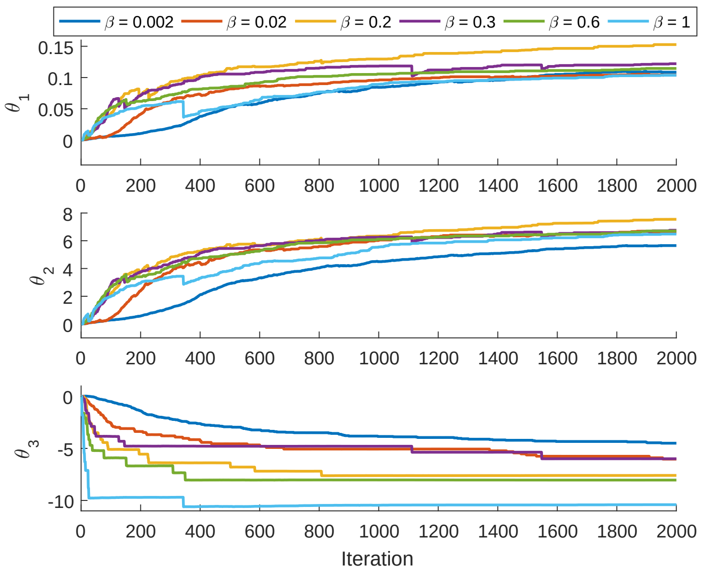


Figure 4.6: Impact of learning rate β on solution convergence ($\alpha = 0.1$)

In observing the impact of β on our solution, it is clear that the value of β does not an impact solution convergence with the same distinction when observing policy parameters θ_1 and θ_2. However, in observing parameter θ_3, we see a trend similar to that of α, in that higher β values cause faster, less smooth convergence, and lower values have smooth curves, but are slower to converge. The one value that breaks this trend for this example is $\beta = 0.3$, which acts closer to the small values of β than values of a similar magnitude.

4.1.8 Real-world experiments

4.2 Kinematic Model Free Robot Control

In this section, we explore the application of Kinematic-Model-Free (KMF) robot control as a potential solution for the many challenges facing task space path planning and automation of continuum robots, while simultaneously reducing the need for complex sensors or extensive knowledge about the continuum robot. In previous works by the authors [132–134], the KMF method has been shown to be extremely effective in permitting a rigid-link robot to learn approximations of local kinematics and dynamics (termed ”kinodynamics”) at various points in the robot's taskspace. These approximations then enabled a robot to follow various trajectories and even adapt to changes in the robot's kinematic structure. The approach learns the local kinodynamics through a series of exploratory actuation primitives and a k-Nearest Neighbor algorithm. The algorithm can predict what inputs to the robot's actuators will result in a motion towards a desired set point. A major advantage of this approach is the simplification of the feedback system: only a camera is needed to track the location of the end-effector relative to the location of the desired set point.

This section gives an overview of the established KMF method, detailing the overall structure of the algorithm and the specific details we utilize in our realization. Further, we present a series of mappings that convert actuation primitives from KMF into universal actions for continuum manipulators by exploiting ideal kinematics.

4.2.1 KMF Algorithm

As detailed in [132], [133], and [134], KMF operates on a learn-as-you-go premise by providing a robot with test motions, or actuation primitives, and then recording the resulting motion of the end-effector after the primitives are applied. A collection of these exploratory primitives across the robot's work-space can then be used to approximate the local kinematics and dynamics of the system and provide a best-fit approximation of what actuation would provide desired motion. After the conclusion of each motion, the resulting movement is compared to the anticipated motion of the end-effector in order to evaluate the accuracy of the approximated "kinodynamic" model. If significant difference exists between expectation and reality, the algorithm triggers a new exploratory phase in order to better sample the local space. In this work, we use a slightly modified implementation of that proposed in the original KMF works. First, we start with the premise of *actuation primitives*: a control signal $\tau(t)$, in this case either a voltage or pressure, that varies as a function of time:

$$\tau(t) = \begin{cases} \tau_p & \text{if } t \in [t_0, t_0 + d_p) \\ 0 & \text{if } t \in (-\infty, t_0) \cup (t_0 + d_p, \infty) \end{cases}, \tag{4.10}$$

where τ_p is defined to be the magnitude of the actuation primitive and d_p is the duration of the primitive. The value t_0 denotes the start time of the action. In this implementation, the value d_p is constant at 1s throughout execution, and all individual actuation primitives share the same start value t_0 for each separate motion. Throughout the execution of KMF, the controller is recording the set p_i of all meaningful actuation primitives executed on the robot, including primitives from both exploration behavior and model predicted behavior.

Next, we describe the process of using the collected data set p_i to produce desirable actuation primitives that will drive the end-effector to a goal location. In this implementation, we will generalize the dimension of our actuator primitives, and subsequent results, to match our later implementation on hardware. To begin, let $\hat{p}$ be an actuation primitive whose parameters $\tau(\hat{p})$ will cause the end-effector to move towards a desired goal. We assume no knowledge about the robot's kinematics or dynamics, and given this, we must estimate the values $\tau(\hat{p})$ that will give us the desired motion. The desired primitive consists of n elements – one for each DoF:

$$b_1 = \begin{bmatrix} \tau_p^1(\hat{p}) \\ \tau_p^2(\hat{p}) \\ \vdots \\ \tau_p^n(\hat{p}) \end{bmatrix} . \tag{4.11}$$

For traditional rigid-link robots, the number of elements in the primitive is also equivalent to the number of actuators. In this implementation, as with the initial implementation, the estimation of $\hat{p}$ is to be determined as a linear combination of the k-nearest neighbor (k-NN) primitives previously executed and saved in the controller's memory. These k-NN primitives are selected according to the distance between the current end-effector location and the starting position of each primitive executed in memory. The resulting k-NN primitives are labeled as $p_1 \ldots p_k$. The linear combination of these k-NN primitives can be expressed in the matrix form:

$$A_1 x = b_1 \tag{4.12}$$

where $x = [x_0, x_1, \ldots, x_k]^T$, is an as yet unknown weight vector. The matrix A_1 contains the parameters of the k-NN primitives:

$$A_1 = \begin{pmatrix} 1 & \tau_p^1(p_1) & \tau_p^1(p_2) & \ldots & \tau_p^1(p_k) \\ 1 & \tau_p^2(p_1) & \tau_p^2(p_2) & \ldots & \tau_p^2(p_k) \\ & \vdots & & & \\ 1 & \tau_p^n(p_1) & \tau_p^n(p_2) & \ldots & \tau_p^n(p_k) \end{pmatrix}_{n \times (k+1)} , \tag{4.13}$$

where $\tau_p(p_i)$ is the magnitude of the i-th actuation primitive. We solve for the unknown coefficients x_i using readily available information concerning the results of our previously experienced k-NN actuation primitives. We thus describe the matrix:

$$A_2 = \begin{pmatrix} 1 & \Delta x(p_1) & \Delta x(p_2) & \dots & \Delta x(p_k) \\ 1 & \Delta y(p_1) & \Delta y(p_2) & \dots & \Delta y(p_k) \\ & \vdots & & & \\ 1 & \Delta z(p_1) & \Delta z(p_2) & \dots & \Delta z(p_k) \end{pmatrix}_{3\times(k+1)} , \tag{4.14}$$

in which $[\Delta x(p_i)\Delta y(p_i)\Delta z(p_i)]^T$ is the relative displacement experienced by the end-effector upon execution of the primitive p_i. Utilizing knowledge of both the manipulator's current end-effector location and the location of the goal destination, we can choose a simple desired displacement for the end-effector to move towards the goal. If the distance between the end-effector is sufficiently small, we can choose the next desired motion to move directly to the desired goal or take an incremental step towards our goal. Regardless, the desired motion is summarized as a relative displacement of the end-effector in global coordinates:

$$b_2 = \begin{bmatrix} \Delta x(\hat{p}) \\ \Delta y(\hat{p}) \\ \Delta z(\hat{p}) \end{bmatrix} \tag{4.15}$$

After designating our desired motion, we can calculate the coefficients $\{x_i\}$ by solving for x in the equation:

$$A_2 x = b_2 \tag{4.16}$$

As discussed in [132], the rank of matrix A_2 is not guaranteed to be full, allowing variability in the solution for x. We once again solve this problem using least squares regression to find a best-fit approximation for x. Once calculated, we can use equation 4.13 to find the desired primitive parameters $\tau_p(\hat{p})$ in b_1. One final adjustment is the weighting of the k-NN primitives according to the distance between the current end-effector location and the starting location of each primitive. By adding this set of weights, $w_i \dots w_k$, we obtain a weighted least squares solution when solving

4.16. Thus, equations 4.13 and 4.16 are adjusted as follows:

$$\begin{cases} A_1 W x = b_1 \\ A_2 W x = b_2 \end{cases}, \tag{4.17}$$

where $W = \text{diag}(1, w_1, w_2, \ldots, w_k)$.

4.2.2 Continuum OctArm Implementation

In this work, we once again choose to model the continuum kinematics using the Allen kinematic parameterization mentioned in Section 2.1. We choose this kinematic description from among the other valid models due to the simplicity of mapping from kinematic values to actuator values, as provided in [112] and reproduced below:

$$l_1 = s(t) - d \cdot v(t) \tag{4.18}$$

$$l_2 = s(t) + \frac{d \cdot v(t)}{2} + \frac{\sqrt{3}d \cdot u(t)}{2} \tag{4.19}$$

$$l_3 = s(t) + \frac{d \cdot v(t)}{2} - \frac{\sqrt{3}d \cdot u(t)}{2} \tag{4.20}$$

4.2.3 KMF for continuum manipulator

As outlined in the initial development of KMF, the action primitives, $\tau(t)$, supplied by the method are not limited by action type or actuation method. When adapted to continuum robots, there are two main types of actuation to be considered: tendon driven devices actuated through electric motors and pneumatically driven artificial muscles. The cases of both extensible and non-extensible manipulators also need to be taken into account. In considering non-extensible, tendon driven robots, one cannot simply pull on a single tendon and achieve desired motion without first or simultaneously letting all opposing tendons go slack or at least reduce tension in an amount proportional to the tendon being pulled on. This means that for fixed length robots, we need to address coupling for the robot by applying differential actuation.

Following from this idea of differential actuation, we make use of the kinematics mentioned in equations 4.18-4.20 to relate individual actuator values to KMF primitives. Here, for both extensible and non-extensible continuum robots, we can map one primitive to the kinematic value u to cause differential bending along the local Y-axis, as related by the sign change of the coefficient for u

in actuators 2 and 3. Likewise, we can map another primitive to the value v to drive differential bending along the local X-axis. Exclusively for extensible continuum manipulators, a final primitive can be mapped to the arc-length value s that is present for all actuators. More explicitly for the OctArm, we can increase and decrease pressure to respective muscles using the following equations:

$$p_1 = \int \tau_s(t)dt - d\int \tau_v(t)dt \tag{4.21}$$

$$p_2 = \int \tau_s(t)dt + \frac{d}{2}\int \tau_v(t)dt + \frac{\sqrt{3}d}{2}\int \tau_u(t)dt \tag{4.22}$$

$$p_3 = \int \tau_s(t)dt + \frac{d}{2}\int \tau_v(t)dt - \frac{\sqrt{3}d}{2}\int \tau_u(t)dt \tag{4.23}$$

where p_i is the pressure value for muscle i in a section, and τ_u, τ_v, and τ_s are the primitives mapped to the kinematic values u, v, and s, respectively.

4.2.4 Mapping KMF to OctArm DoF

In order to implement KMF for the 9 DoF OctArm and observe the efficacy of the method, we began with a simplified mapping of KMF primitives to OctArm DoF. Here, we present 3 mappings of KMF primitives to OctArm DoF, gradually increasing the complexity and redundancy of the system in order to assess KMF.

4.2.4.1 Mapping 1: 3 DoF

In this first mapping, we treat the OctArm as a single continuum section, providing identical inputs to muscle 1 of each section, and the same setup for muscles 2 and 3. Given that the OctArm is extensible by design, we can use the primitive mapping described for the basic extensible continuum section:

$$[\tau_1, \tau_2, \tau_3] = [\tau_u, \tau_v, \tau_s] \tag{4.24}$$

This mapping accentuates the under-actuated nature of continuum robots by displaying non-constant curvature. Non-constant curvature will be especially prevalent in the proximal (base) section, which must support the load of both the middle and distal sections. Even in this reduced number of DoF, traditional task-space planning and control methods potentially suffer from modeling and sensing errors.

As noted when first reporting KMF, the order in which the primitives are arranged in this mapping, and subsequent mappings, is not important to the method. The order provided is simply for reporting purposes and for clarity of the mapping.

4.2.4.2 Mapping 2: 4 DoF

For the second mapping, we treat the OctArm as a non-extensible, 2 section continuum manipulator. We accomplish this by treating the base and mid-sections of the OctArm as one section, receiving matching inputs to each of the muscle groups as in the first mapping. To simulate non-extensibility, we initialize each OctArm section to their respective mid-range extension value, essentially half-way between their minimum and maximum pressure. After initialization, only differential inputs utilizing u and v are given to the system. This provides the system with 4 DoF, with the primitive vector:

$$[\tau_1, \tau_2, \tau_3, \tau_4] = [\tau_{u_b}, \tau_{v_b}, \tau_{u_t}, \tau_{v_t}] \tag{4.25}$$

The second mapping serves to demonstrate using KMF in the non-extensible class of continuum robots. It also serves to introduce redundancy into the system.

4.2.4.3 Mapping 3: 6DoF

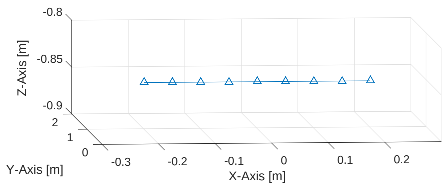
The final mapping makes use of each section of the OctArm independently of the others. As with the second mapping, we model the sections as constant length and provide only differential inputs to each section, giving the system a total of 6 DoF. Our primitive vector for this case is:

$$[\tau_1, \tau_2, \tau_3, \tau_4, \tau_5, \tau_6] = [\tau_{u_b}, \tau_{v_b}, \tau_{u_m}, \tau_{v_m}, \tau_{u_t}, \tau_{v_t}] \tag{4.26}$$

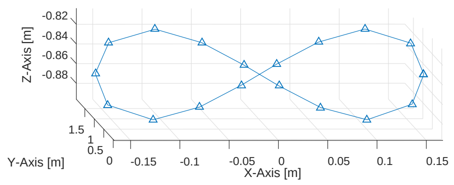
This mapping represents a 3-section non-extensible continuum robot, and implements the largest degree of redundancy explored in this work.

4.2.5 Experimental Validation

In evaluating the efficacy of KMF across the different mappings, we consider two paths in the OctArm's taskspace for the end-effector to follow. As part of a class of robots whose hardware naturally traces curves and bending motions, one of the most difficult motions for a continuum robot to perform for the end-effector is a straight line. Consequently, the first path we test is a straight



(a) Line path in Kinect coordinate system



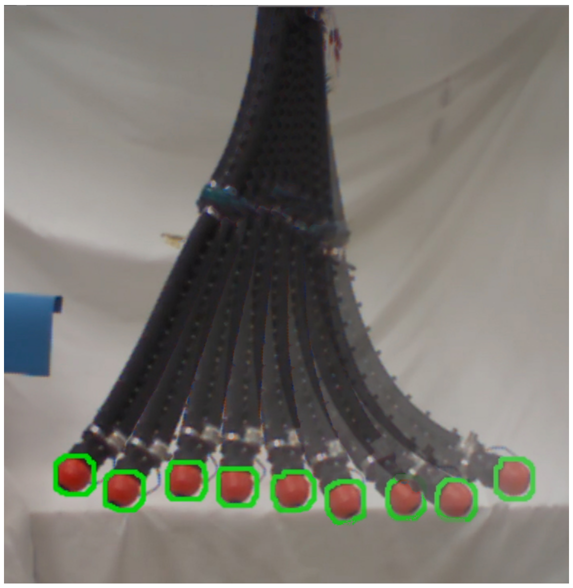
(b) Proposed lemniscate path in Kinect coordinate system

Figure 4.7: Nominal Trajectories for End-effector tracing

line that runs through the taskspace, parallel to the x-axis. The path can be seen in Figure 4.7a. The exact path is a series of 9 points spaced evenly between [x,y,z] = [-0.2m,0.95m,-0.85m] and [0.2m,0.95m,-0.85m] in the camera's coordinate frame. In executing the path, the OctArm must travel to the start point (x=0.2m) then follow the full length of the path and back again to the start of the line.

The second path considered is a planar lemniscate path sitting a plane that is parallel, but offset, from the OctArm's local XZ-plane. The path consists of 19 discrete points (the start and stop point are the same), shown in Figure 4.7b, and spaced approximately 5cm apart. For both paths, the end-effector must arrive within 2cm of error (as depicted in the plots by the green region) of the current goal point before moving to the next. An example of the OctArm executing both the line and lemniscate path can be seen in Figure 4.8.

Tracking of the OctArm end-effector along the desired trajectories is achieved through the use of a Microsoft Kinect [135] to track the center of the OctArm end-effector using hue filtering. We then use the Kinect API to map color pixel coordinates to depth values and consequently to real-world coordinates. Goal points along the desired paths are provided through a MATLAB script, with the coordinates themselves set with respect to the Kinect coordinate system. Kinect tracking is implemented in C++ using Visual Studio 2015, and the same program also sends the end-effector coordinates to a Simulink model via network socket. A second Simulink model is responsible for integrating the actuation primitives into pressure values that drive the pressure regulators controlling the OctArm. Finally, the MATLAB script that provides the goal locations as the end-effector moves through the task space also hosts the core KMF algorithm by receiving end-effector location, calculating and recording primitives and motions, and providing primitives to the Simulink model driving the OctArm.


(a) OctArm following line path through task space

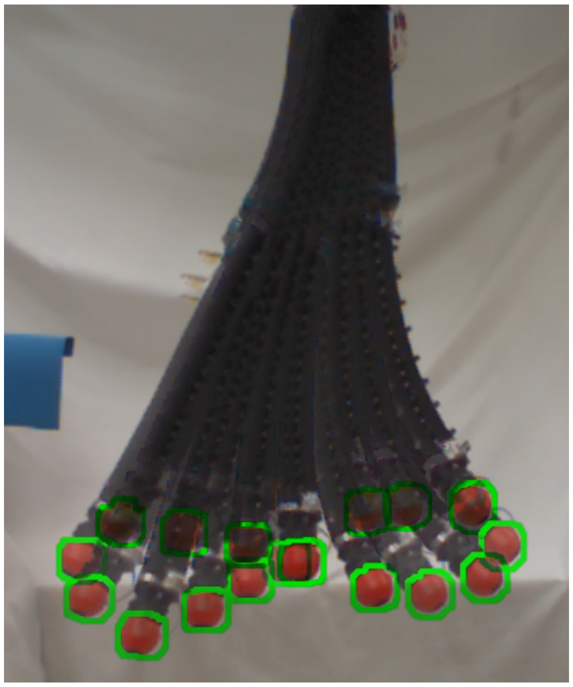
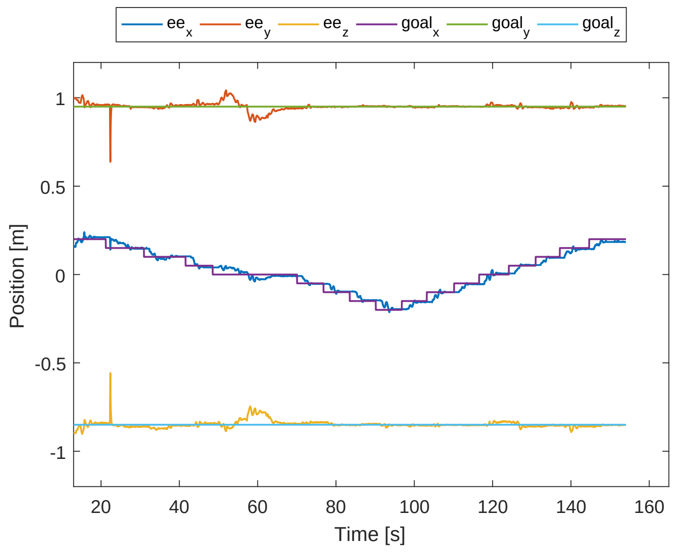

(b) OctArm following lemniscate path in task space.

Figure 4.8: OctArm Manipulator tracing paths in work space.

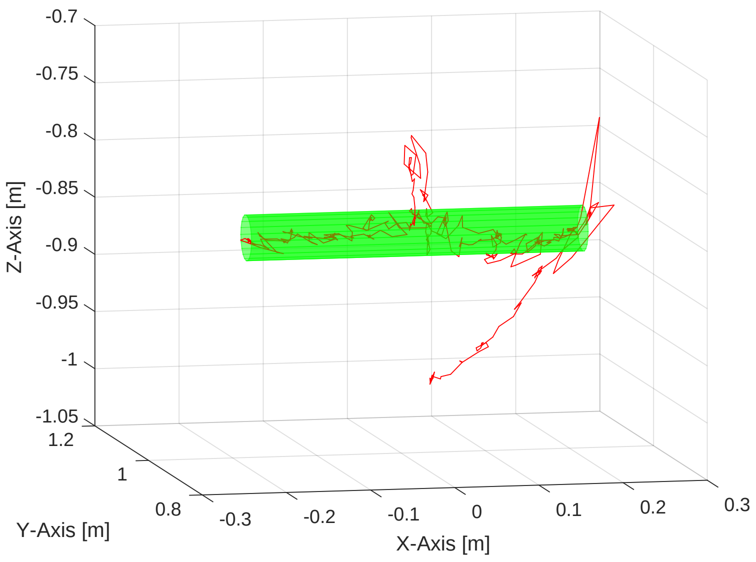
4.2.5.1 Results: 3 DoF Mapping

In implementing the first mapping, the KMF method was tasked with solving what would ideally be a relatively simple mapping of 3 DoF to 3 dimensions of movement for the end-effector. In reality, and as mentioned briefly when introducing the mapping, the compliance of the OctArm and the fact that the system is under the effects of gravity makes this a difficult problem to model, much less solve. In observing robot performance under KMF, seen in Figures 4.9 and 4.10, it can be seen that with sufficient exploration time the method is able to track both test trajectories in Figure 4.7 well and in reasonable time.

For the line trajectory, as seen in Figure 4.9a, starting from the time of arrival at the first point, KMF actuates the robot to step along the remaining 16 consecutive points with little to no error outside of the 2cm limit in under 3 minutes. This value is more impressive when considering that the system is paused (not a result of KMF) 3 seconds between reaching each goal point and starting to move to the next point on the path (1 second to ensure that system is settled, 2 seconds of clarity of the result).



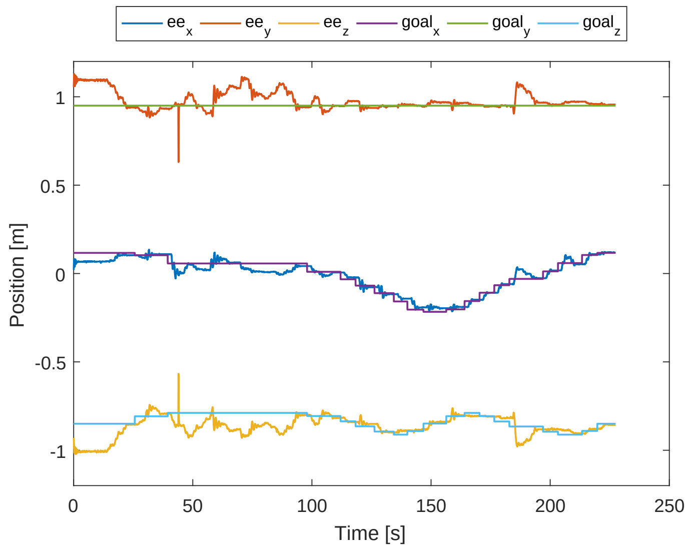
(a) 3DoF Mapping tracking of line path



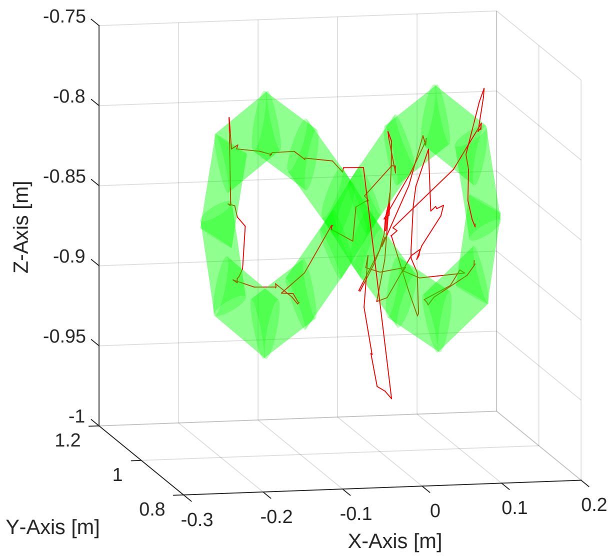
(b) Measured end-effector location while following line path (3DoF)

Figure 4.9: 3DoF OctArm line following

The 3 DoF mapping also proves capable of completing the lemniscate path, which requires additional motion compensating against gravity. Here, we see more travel outside of the nominal path we expect between consecutive points, but still have eventual recovery and convergence to each point. The travel outside the nominal path could potentially be improved either through smaller steps along the path or providing further exploration in this region of the taskspace.



(a) 3DoF Mapping tracking of line path



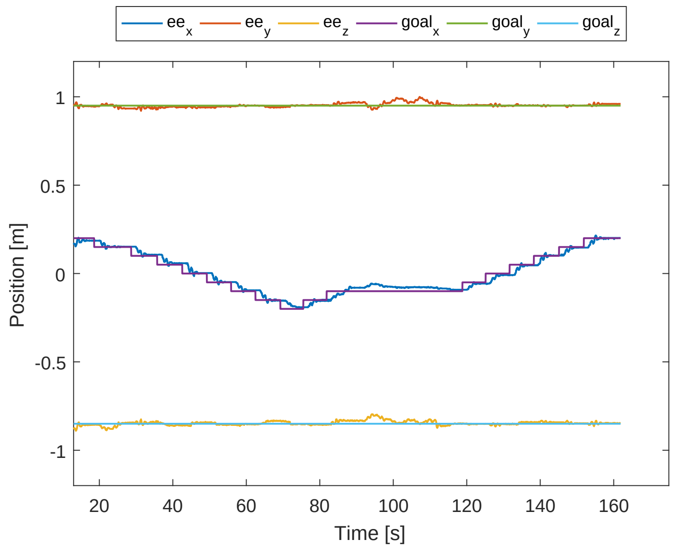
(b) Measured end-effector location while following line path (3DoF)

Figure 4.10: 3DoF lemniscate path following

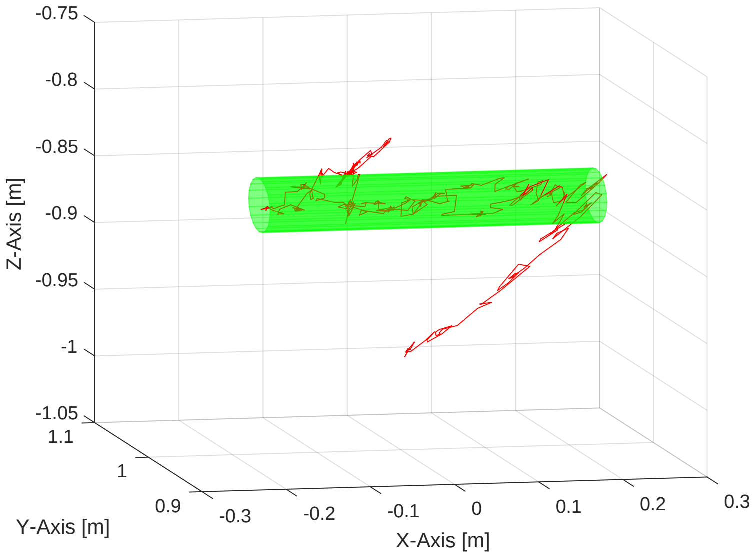
4.2.5.2 Results: 4 DoF Mapping

In implementing the second mapping, there was uncertainty about how introducing redundancy would impact the performance of the algorithm. In observing the results in Figures 4.11 and 4.12, we can clearly see that, at least for these two paths, the addition of another DoF did not greatly hinder the performance. While not shown in these snapshot results, it was noticeable during training that it took longer for the 4 DoF mapping to obtain enough experience to traverse the space intentionally. This is due in part to the fact that the 4-dimensional exploration primitives, even when orthogonal, could not guarantee a diverse set of motion in the task-space. The same can be said for the previous 3 DoF mapping given the non-linearity of the system, but the extra degree of redundancy does not appear to help exploration.

When comparing the results of the 3 and 4 DoF mappings at following the line path, there is little discernible difference between the two performances. In general, we observe that both mappings can still make inaccurate predictions at times, depending on the closest neighbor candidates at each given point, but both have similar completion time.



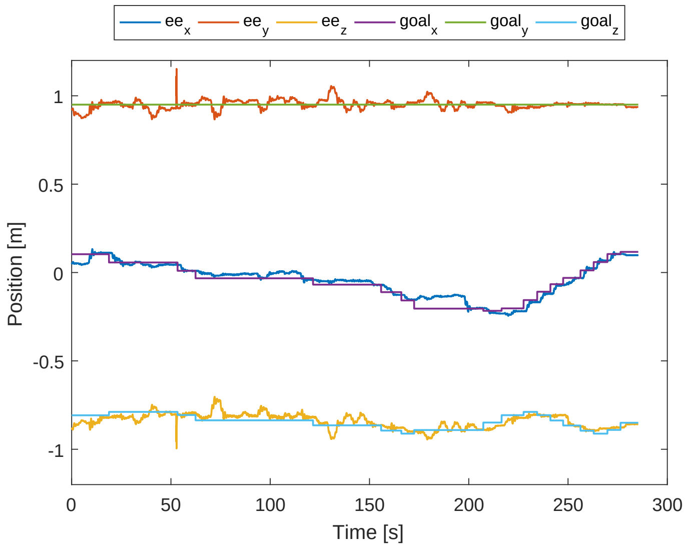
(a) 4DoF Mapping tracking of line path



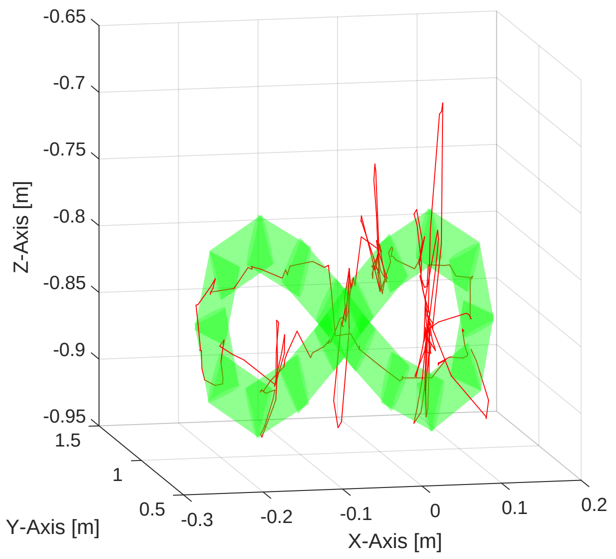
(b) Measured end-effector location while following line path (4DoF)

Figure 4.11: 4DoF line following

Likewise for the lemniscate path, we see similar times of completion and accuracy of performance between the 3 and 4 DoF mappings. It is worth noting for later discussion that both mappings appear to deviate less from the acceptable region of error in the left loop of the path.



(a) 4DoF Mapping tracking of line path



(b) Measured end-effector location while following line path (4DoF)

Figure 4.12: 4DoF lemniscate following

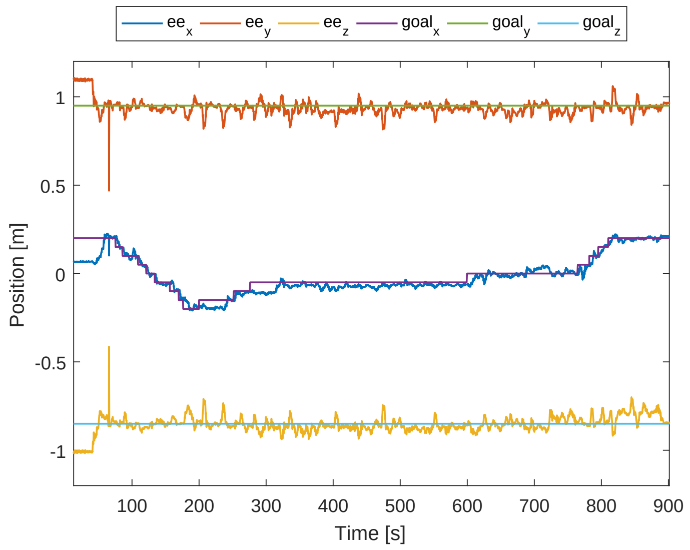
4.2.5.3 Results: 6 DoF Mapping

As with the 4 DoF mapping, it was unknown exactly what impact the extra redundancy would have on the performance, though it was anticipated that 6 DoF would be a greater challenge in generating a solution than the 4 DoF mapping, as is the case with most cases in motion planning involving increased degrees of freedom. As can be seen in Figure 4.13a, the first notable impact is on completion time. The result seen here is one of faster examples we recorded using this mapping, but is still 4-5 times slower than the 3 and 4 DoF mappings. In observing Figure 4.13b, it becomes apparent that a large portion of that time difference is spent re-exploring and miscalculating primitives in an already explored region.

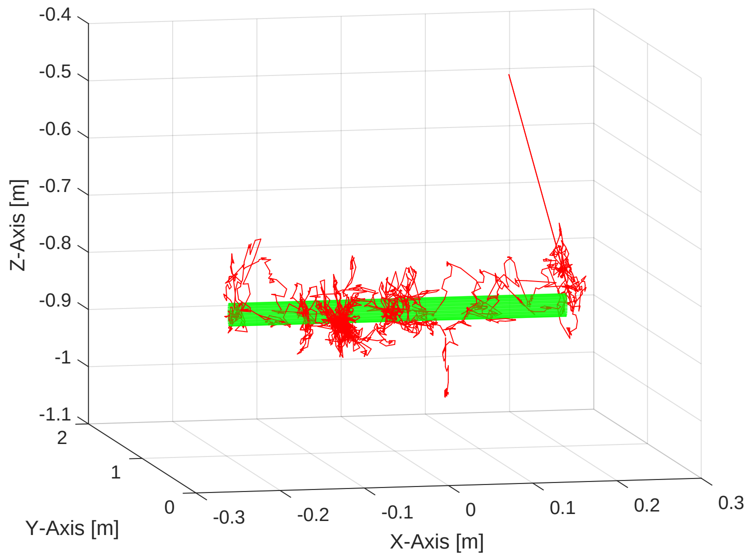
The 6 DoF mappings suffers from the same issue observed in the 4 DoF mapping in that there is no guarantee of sufficient taskspace exploration to use for predictions when given a diverse, or even orthogonal, set of actuation primitives. Predictably, where the 6 DoF mapping appears to suffer more than the 4 DoF mapping relates to the extra increase in redundancy. The extra redundancy increases the chance that two neighboring primitives could have been sampled from very different and relatively remote regions of the robot's configuration space.

Fortunately, KMF has already been shown to be able to adapt to changes in kinematics while actively tracking and learning. This is evidenced by the fact that the 6 DoF mapping is still able to converge to each point along the path, even if taking a different path between each point.

Given the difficulty of solving the line path after several learning trials in this example, we forgo attempting to trace the lemniscate path with the 6 DoF mapping for this reporting.



(a) 4DoF Mapping tracking of line path



(b) Measured end-effector location while following line path (4DoF)

Figure 4.13: 6DoF line following

4.3 Discussion

4.3.1 Reinforcement Learning Discussion

As we expand the DoF and range of motion, we quickly arrive at a scenario where the state-space contains a multitude of local states that are able to “see” the goal object. The algorithm presented here does not guarantee that the robot will arrive at exactly the “best state” that has the closest view of the object. However, the method can arrive to one of the local “best states” distributed throughout the state-space.

Indeed, when also taking into account our modification in Eq. 4.9, we can see an example of the policy settling in “good enough” states in our simple scenario. States 5, 9, and 12 are candidates for acceptable states by simply seeing the goal object. In purely quantitative reasoning, state 12 is the “best” goal state in that it has the best view of the goal ($S_{obj} = 1$), followed by state 9 ($S_{obj} = 0.36$), and then state 5 ($S_{obj} = 0.05$). The uniform random policy in aliased states prevents the system from settling in, or oscillating between, two non-goal states, and the remaining states clearly have an eventual path to either state 12 or 9. Even in state 5, in part receiving help from our modification, the policy has greater chance of moving to state 4 and then to state 9 over staying in 5. However, there is not an overwhelming likelihood of the system leaving state 9 in order to settle at state 12.

In order to address scenarios in which the continuum robot is given a higher number of states, we could able to add additional features and reward values that could encourage the robot to converge to states that have a “better” view of the goal. For example, if the planar robot has multiple, neighboring states in which the goal is visible, we could add a feature and related reward based on the distance of the object from the center of the camera. This could drive the robot to take actions that align the object in the middle of the camera for a better view.

In presenting this material, we simplify the image processing method of recognizing our goal object. In practical application, we can substitute our goal related features with those corresponding to the location of an actual feature of interest. Examples could be the results from a template matching algorithm that returns confidence values and locations, among other details.

Another design choice was the use of configuration space over actuation space. By using the configuration space, we have a meaningful interpretation of the system states that can be applicable to a variety of continuum manipulators. In converting to actuation space, we can rely on closed-loop controllers adapted to individual continuum robots to convert values such as curvature and orientation to physical values such as tendon lengths or pneumatic pressures.

As a function of using a simulation model, we make the assumption in this work that our state space and actions are well within the defined configuration space of our system. We also assume that transitions between neighboring states are guaranteed and deterministic. In practice, and subsequently the scope of future application, we will have to rely on control methods to achieve state transitions, and relax the guarantee of expected state transition. The need for reliable control methods in spatial motion is still a topic of research in continuum robotics [3], but it will not be necessary to require methods dependent on complex dynamic models to perform state transitions for a set of static states.

The simulation model in Gazebo enables us to quickly test multiple scenarios and extend the size of the robot's state-space. The simulation results presented here also offer a path to using detailed simulations to complete offline training in simulation followed by online execution of the learned policy on a physical robot. In [133], the authors explored how KMF is capable of overcoming various unknown kinematic constraints and adjustments. In this work, we did not actively pursue adding unmodeled constraints to our assessment, but during our final testing the OctArm developed a considerable leak in muscle 1 of the distal section. The leak results in a considerable curve in the section when prior to the leak the section would be straight. Normally, such a leak would be catastrophic to control systems and models and prevent proper function, but in the case of KMF, this leak did not prove detrimental to the performance.

With regards to increasing redundancy in our results, KMF proved to be able to enable the OctArm manipulator to arrive at points in the task space regardless of the number of DoF. Where redundancy does become a factor is in the transitioning between two consecutive points, which can vary depending on the configuration of the OctArm at the start of the motion. In the case of 3 DoF and 4 DoF, we observe desirable motion between discrete points in both paths, generally staying within the desired error. For 6 DoF, we observe greater variation in the motion between points and the configurations themselves at each point in time. While this trend is to be expected for more complex scenarios, we still observe the ability of KMF to improve performance as the system explores more of the taskspace. This is another example of KMF adapting to new changes in the local kinodynamics of the OctArm, as a new arrival at the same point in the task-space no longer means the same kinematic structure of the OctArm.

One challenge to this work, and to tracking end-effectors from a single camera in general, was the need to avoid occlusion of the end-effector from the Kinect camera. We accomplish this by implementing artificial bending limits at the actuator level, while still allowing KMF to be unaware of the system state. The limit itself is created by applying a dynamic saturation level for the maximum and minimum pressure for muscles 2 and 3, respectively. Since muscles 2 and 3 are responsible for bending towards and away from the camera, this dynamic saturation limit essentially prevents muscle 2 from being pressurized significantly greater than muscle 3 and likewise prevents muscle 3 from reducing pressure to be significantly below that of muscle 2. In future applications, this challenge of occlusion could be solved by using an array of cameras to track the end-effector.

As alluded to when introducing the respective mappings for the OctArm, part of the aim of this work is to display the applicability of KMF to continuum robotics beyond the OctArm. The results here show that KMF is capable of handling both the theoretical hyper-redundancy of continuum robots as well as designed redundancy of continuum manipulators with multiple sections. These results also lend credit to the idea that KMF primitives could be used as torque inputs to tendon driven continuum robots that only have the ability to bend and thus any inputs must be simultaneously pulling and releasing tendon.

4.3.2 Benchmarking KMF for Continuum Robots

In providing comparison for KMF performance to more traditional methods for closed-loop control of continuum robots, we discuss a collection of results from the literature that highlight both the success of KMF robot control in this paper and the difficulty of trying to directly compare the success with other methods on both the OctArm and similar hardware. To begin, we look at two works that explore the accuracy of forward kinematic models with respect to end-effector location explicitly performed on the OctArm manipulator. In [120], a series of forward kinematic models based upon constant curvature assumptions are directly compared for accuracy relative to real-world measurements of the OctArm's end-effector. The results here show that while some models perform better than others, the greatest accuracy seen across the samples of the robot workspace was in excess of 5% (as measured with respect to the OctArm's overall length), or over 5cm. These models also rely on the internal measurement of the OctArm's shape, giving no direct way to accurately relate assumption based measurements to real-world coordinates. In [58], the work compares the accuracy of constant curvature kinematics to statics-based models that are derived to be geometrically exact and provide a relationship between input pressure and end-effector location for the OctArm. In this work, the geometrically exact models prove to be better at predicting gravity related deflections for the OctArm than constant curvature models, but still provides approximately 5% error between the predicted and actual end-effector locations, and does not provide a means for mapping real-world coordinates to actuator inputs for control.

Next, we discuss a series of works that explore predicting and controlling end-effector locations of continuum robots, both in static and dynamic experiments. The works utilize various continuum robots with a variety of material and dynamic model related parameters. In [66], a combination of variable curvature Cosserat-rod based static and Lagrange dynamic models are presented to provide control for a two-section continuum manipulator that is similar in composition to the OctArm, containing a mixture of rigid and soft materials and using pneumatic actuation. While reporting much improved accuracy for their model in comparison to the previous state of the art, they report 6-8% error in static experiments and excess of 16% error in dynamic motions of the manipulator's end-effector. Another relevant work, [64], describes the statics and dynamics of a tendon driven robot with a flexible backbone through a combination of Cosserat-rod and Cosserat-string models. This work reports a 1.7% error between the predicted and measured end-effector location in static experiments for a single-section continuum robot, but does not provide a method for predicting actuator inputs for a desired end-effector location, which is likely not a unique solution for multi-section continuum robots. Finally, in [136], a non-dynamic model based approach based on forward and inverse kinematics coupled with an adaptive neural network control is tested. The combination of forward kinematics and the adaptive neural control provide for approximately 1% error in end-effector location when tracking multi-point paths. The paths are created using the robot's forward kinematics and are well suited for a continuum robot's inclination for following curved paths.

In all, these results from the literature, among others, show various results below a 10% error margin with respect to robot length but also carry higher computational, sensing, and modeling costs. Thus, KMF robot control presented here establishes that reasonable tracking error for notably difficult motions and static end-effector location control ($\leq$2% error) can be obtained with this class of robot with a simple input-output framework and end-effector tracking through a 3D camera or network of standard cameras.

4.3.3 Expanding KMF to Space Deployment

One of the many potential advantages of KMF with respect to continuum robot automation, and robot automation in general, derives from the simplicity of the system required to implement it. Traditional robot control and planning relies on sensing, often on-board, to close the control loop and provide the system state to various models. In long-term deployment environments such as the International Space Station (ISS), the future Lunar Gateway, or even extra-planetary bodies, critical sensor malfunctions can be catastrophic to a mission.

In the case of our ongoing research, the previously mentioned Tendril robot [137], is designed for applications concerning inspection and monitoring in Space operations. One of the target scenarios for this robot is deployment on the ISS for automated inspection of hard-to-reach locations and commensurately reducing astronaut workload. An example simulation of this scenario with Tendril deployed in and looking around the ISS can be seen in Figure 4.14. The simulated Tendril actuation has been driven by the KMF approach reported herein. Both aboard the ISS and the future Lunar Gateway, there are potential times when a deployed robot cannot be serviced, even for a faulty sensor, endangering mission success. KMF offers an alternative loop for maintaining these automated systems.

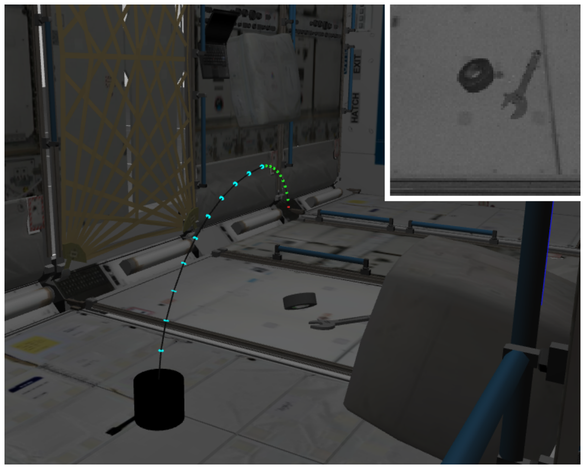


Figure 4.14: Tendril Continuum Manipulator in Simulated Inspection Task on ISS

Chapter 5

Conclusions and Future Work

The work presented in this dissertation has covered a number of areas of continuum robot research, spanning interfacing, modeling, and automation. In Chapter 2, a novel hardware interface for the bilateral teleoperation of continuum manipulators was presented, along with an expansion of software that enables continuum robots to coordinate and collaborate with various robotics platforms and systems. The creation of a bilateral haptic interface for continuum robots opens a new area of research to explore the impact of lag and round-trip delay on teleoperation of continuum robots, a field already currently explored in more traditional robotic platforms. In Chapter 3, we introduced an approximate dynamic model for continuum robots based on well-established theory for rigid-link robots, expanding previous approximate models in order to include non-constant curvature and relate inherent materials properties present in continuum robots to rigid-link related mechanisms and ideas. Finally, Chapter 4 explored automation of continuum robot tasks through two novel approaches to learning, in one case a reinforcement learning paradigm to optimize search, and in another exploring the ability to perform tasks without complex models or sensing.

5.1 Theoretical Implications and Recommendations for Further Research

5.1.1 Interfacing

The introduction of a continuum haptic interface in Chapter 2 of this dissertation presents several opportunities for expanded research. The full expanse of new research opportunities is difficult to predict, but one of the keys areas open to investigation is the impact of lag in the bilateral teleoperation of continuum robots, both in mental workload for a user and in studying what information can be predicted by the interface or lag mitigation techniques. One example of critical need for studying the impact of lag is tied to continuum robot compliance. As mentioned previously, the compliance of continuum robots is useful for environment interaction and consequently can prevent robot damage during collision. However, even continuum robots have limitations on unintended deformation. The lag in perception of a collision with the environment can be the difference between a tolerable collision and a collision in which the robot cannot extract itself or is permanently damaged.

Other potential research could explore evaluating off-the-shelf haptic interfaces, or different haptic modalities applied to the HaptOct, against the work presented herein in order to see which best conveys information to the user. This would require re-evaluation of the modes described briefly in Section 2.4 and the creation of mappings between standard haptic interfaces and continuum robots, but would be an engaging exploration of the human factor in continuum interfacing.

5.1.2 Continuum Robot Simulation

Missing from the physics-enabled simulation model developed and described in Chapter 2 is the impact of collisions on a continuum shape. As visually, compellingly reported in [138], the application of constant tendon displacement vs constant tendon tension can greatly alter the impact of collisions or external loads on the backbone shape. We anticipate the development of additional tools similar to the “jointSpring2” plugin to detect and redistribute internal forces in correspondence to where collisions occur along the length of the manipulator. This would present a challenging research question and could be another helpful tool during the design process of new continuum robots.

The expanded availability of the Gazebo simulation and ROS for continuum robots can also allow us to develop and test life-like scenarios aboard the ISS for the continuum Tendril robot and in collaboration with current platforms such as Robonaut [123].

5.1.3 Dynamic Modeling

In observing the initial results comparing the piece-wise constant curvature model introduced in Chapter 3 of this dissertation against the response of the physical robot, we see strong correlation between the predicted reaction to the motion of the physical robot. While this is a promising result, it was also evident that the model over estimated the displacement in the proximal section as a result of bending in the distal section, likely a result of either incorrect parameter estimation of the stiffness and mass, or due to unmodeled dissipative forces like friction. Future work into approximate modeling could explore further unmodeled elements, or look to improve parameter estimation through learning methods.

Another element from this model that could be improved in future work is the estimation of the robot's state during motion. While the current minimum energy method works well for static states and functions as an approximation during motion, it does not fully capture the effects of momentum on the distribution of curvature within a section. Potential research for resolving this could include filtering methods that predict the actual shape of the robot given state variables. Alternatively, the application of sensor fusion between information from IMUs or external cameras and the internal string encoders could provide improved prediction of the full system state.

Finally, more testing of the model's effectiveness is of immediate interest for future research. This include exploring controllers that best utilize the model and adjust for unmodeled effects such as friction or potentially incorrectly modeled parameters such as mass. One candidate for control research would be an adaptive controller that can correct for approximated parameters within the model. Model predictive controllers could also prove useful for correcting against the errors between the simulation and physical robot reaction to distal section motion.

5.1.4 Automation

In applying reinforcement learning to continuum robots in Chapter 4 of this work, we simplified many of the assumptions made about the robot, including guaranteeing the arrival of the robot to discrete states in its configuration space. A natural progression for future work will be to introduce a more continuous state-space as well as a continuous action space, consequently introducing uncertainty in transitioning between states and in state perception. The expanded work could then explore the refinement of a Gaussian distribution based policy. Upon moving to continuous action spaces, future work could also test high-capacity function approximators such as neural networks to represent the policy and value function where raw pixel data from the robot's camera be used as state inputs. Such ideas will require a more robust policy optimization approach such as the Proximal Policy Optimization [139] that uses a modified objective to the MDP problem to estimate the expectation of the current policy. Future work could also explore how maximum entropy RL frameworks such as the recent Soft Actor-Critic model [140] can extend to the continuum domain, as they are known to be sample efficient and more robust to hyper-parameter tuning which will be needed when training neural network-based Gaussian policies.

The research considering model-free control methods introduced in Chapter 3 also has great potential for expansion in future research. The KMF control method explored here only partially expands the work already developed in [132–134], to the specific case of continuum robots. With respect to continuum robots, KMF could be used in conjunction with multiple tracking points along the length of the robot for attempting whole-arm manipulation or for controlling higher DoF as in the 6 DoF mapping experiments. Future work could also explore different memory management methods in order to reduce the constant exploration that was observed in the 6 DoF mapping. One such method could be evaluating memory based on the success of each motion, where primitives that repeatedly skew predictions from reality are reduced in priority or forgotten altogether, allowing more accurate primitives to drive motion instead of just nearest neighbors.

5.2 Concluding Remarks

While the field of continuum robotics is young relative to the broader robotics field, continuum robotics research greatly benefits from the discoveries and methods made across the spectrum of robotics. We hope the work presented in this dissertation serves in part to show the applicability of general robotics concepts to continuum manipulators while also highlighting some of the niche advantages and capabilities of this unique class of manipulators. As we expand the capabilities of continuum robots, such as introducing bilateral teleoperation research opportunities, we can expect that discoveries made in this field can in turn contribute back to the broader robotics community.

Bibliography

[1] G. Robinson and J. Davies, “Continuum robots - a state of the art,” in *Proc. IEEE Int. Conf. Robot. Autom.*, Detroit, MI, 1999, pp. 2849–2854.

[2] W. Kier and K. Smith, “Tongues, tentacles, and trunks: The biomechanics of movement in muscular hydrostats,” *Zoological Journal of the Linnean Society*, vol. 83, no. 4, pp. 307–324, 1985.

[3] T. G. Thuruthel, Y. Ansari, E. Falotico, and C. Laschi, “Control strategies for soft robotic manipulators: A survey,” *Soft robotics*, vol. 5, no. 2, pp. 149–163, 2018.

[4] D. Trivedi, C. Rahn, W. Kier, and I. Walker, “Soft robotics: Biological inspiration, state of the art, and future research,” *Applied Bionics and Biomechanics*, vol. 5, no. 2, pp. 99–117, Jun. 2008.

[5] I. Walker, D. Dawson, T. Flash, F. Grasso, R. Hanlon, B. Hochner, W. Kier, C. Pagano, C. Rahn, and Q. Zhang, “Continuum robot arms inspired by cephalopods,” in *Proc. SPIE Conf. Unmanned Ground Veh. Tech.*, Orlando, FL, 2005, pp. 303–314.

[6] M. B. Wooten and I. D. Walker, “A novel vine-like robot for in-orbit inspection,” in *Proc. Int. Conf. on Env. Sys.*, Bellevue, WA, 2015.

[7] Y. Ozkan-Aydin, M. Murray-Cooper, E. Aydin, E. N. McCaskey, N. Naclerio, E. W. Hawkes, and D. I. Goldman, “Nutation aids heterogeneous substrate exploration in a robophysical root,” in *2019 2nd IEEE International Conference on Soft Robotics (RoboSoft)*. IEEE, 2019, pp. 172–177.

[8] E. Del Dottore, A. Mondini, A. Sadeghi, V. Mattoli, and B. Mazzolai, “Circumnutations as a penetration strategy in a plant-root-inspired robot,” in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016, pp. 4722–4728.

[9] E. W. Hawkes, L. H. Blumenschein, J. D. Greer, and A. M. Okamura, “A soft robot that navigates its environment through growth,” *Science Robotics*, vol. 2, no. 8, 2017.

[10] M. C. Lastinger, S. Verma, A. D. Kapadia, and I. D. Walker, “Tree: A variable topology, branching continuum robot,” in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 5365–5371.

[11] M. B. Wooten and I. D. Walker, “Vine-inspired continuum tendril robots and circumnutations,” *Robotics*, vol. 7, no. 3, p. 58, 2018.

[12] W. McMahan, M.Pritts, V. Chitrakaran, D. Dienno, M. Grissom, B. Jones, M. Csencsits, C. Rahn, D. Dawson, and I. Walker, “Fields trials and testing of octarm continuum robots,” in *Proc. IEEE Int. Conf. Robot. Autom.*, Orlando, FL, 2006, pp. 2336–2341.

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

[14] J. Lock, G. Laing, M. Mahvash, and P. Dupont, “Quasistatic modeling of concentric tube robots with external loads,” in *Proc. IEEE/RSJ Int. Conf. Intel. Robot. Syst.*, Taipei, Taiwan, 2010, pp. 2325–2332.

[15] H. Su, D. Cardona, W. Shang, A. Camilo, G. Cole, D. Caleb Rucker, R. Webster III, and G. Fischer, “A mri-guided concentric tube continuum robot with piezoelectric actuation: A feasibility study,” in *Proc. IEEE Int. Conf. Robot. Autom.*, St. Paul, MN, 2012, pp. 1939–1945.

[16] G. Chirikjian and J. Burdick, “A modal approach to hyper-redundant manipulator kinematics,” *IEEE Trans. Robot. Autom.*, vol. 10, no. 3, pp. 343–354, Jun. 1994.

[17] M. Grissom, V. Chitrakaran, D. Dienno, M. Csencsits, M. Pritts, B. Jones, W. McMahan, D. Dawson, C. Rahn, and I. Walker, “Design and experimental testing of the octarm soft robot manipulator,” in *Proc. SPIE Conf. Unmanned Sys. Tech.*, Kissimee, FL, 2006, pp. 109–114.

[18] D. Braganza, M. L. McIntyre, D. M. Dawson, and I. D. Walker, “Whole arm grasping control for redundant robot manipulators,” in *Proc. American Control Conf.*, Minneapolis, MN, 2006, pp. 3194–3199.

[19] A. Kapadia and I. Walker, “Self-motion analysis of extensible continuum manipulators,” in *Proc. IEEE Int. Conf. Robot. Autom.*, Karlsruhe, Germany, 2013, pp. 3105–3110.

[20] I. Walker, “Continuous backbone ”continuum” robot manipulators: A review,” *ISRN Robotics*, vol. 2013, no. 1, pp. 1–19, Jul. 2013.

[21] J. Burgner-Kars, D. C. Rucker, and H. Choset, “Continuum robots for medical applications: A survey,” *IEEE Trans. Robot.*, vol. 31, no. 6, pp. 1261–1280, Dec. 2015.

[22] R. Buckingham and A. Graham, “Snaking around a nuclear jungle,” *Ind. Robot: An Int. Jour.*, vol. 32, no. 2, pp. 120–127, Feb. 2005.

[23] E. Butler, R. Hammond-Oakley, S. Chawarski, A. Gosline, P. Codd, T. Anor, J. Madsen, P. Dupont, and J. Lock, “Robotic neuro-endoscope with concentric tube augmentation,” in *Proc. IEEE/RSJ Int. Conf. Intel. Robot. Syst.*, Vilamoura, Portugal, 2012, pp. 2941–2946.

[24] Y. Chen, J. Liang, and I. Hunter, “Modular continuum robotic endoscope design and path planning,” in *Proc. IEEE Int. Conf. Robot. Autom.*, Hong Kong, China, 2014, pp. 5393–5398.

[25] N. Simaan, R. Taylor, and P. Flint, “A dextrous system for laryngeal surgery,” in *Proc. IEEE Int. Conf. Robot. Autom.*, New Orleans, LA, 2004, pp. 351–357.

[26] M. Tonapi, I. Godage, A. Vijaykumar, and I. Walker, “Spatial kinematic modeling of a long and thin continuum robotic cable,” in *Proc. IEEE Int. Conf. Robot. Autom.*, Seattle, WA, 2015, pp. 3755–3761.

[27] L. Dupourqué, F. Masaki, Y. L. Colson, T. Kato, and N. Hata, “Transbronchial biopsy catheter enhanced by a multisection continuum robot with follow-the-leader motion,” *International journal of computer assisted radiology and surgery*, vol. 14, no. 11, pp. 2021–2029, 2019.

[28] M. Neumann and J. Burgner-Kahrs, “Considerations for follow-the-leader motion of extensible tendon-driven continuum robots,” in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016, pp. 917–923.

[29] R. Buckingham, "Snake arm robots," *Ind. Robot: An Int. Jour.*, vol. 29, no. 3, pp. 242–245, Mar. 2002.

[30] J. Mehling, M. Diftler, M. Chu, and M. Valvo, "A minimally invasive tendril robot for in-space inspection," in *Proc Biorobotics Conference*, Pisa, Italy, 2006, pp. 690–695.

[31] J. C. Crusan, R. M. Smith, D. A. Craig, J. M. Caram, J. Guidi, M. Gates, J. M. Krezel, and N. B. Herrmann, "Deep space gateway concept: Extending human presence into cislunar space," in *2018 IEEE Aerospace Conference.* IEEE, 2018, pp. 1–10.

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

[33] J. D. Greer, T. K. Morimoto, A. M. Okamura, and E. W. Hawkes, "A soft, steerable continuum robot that grows via tip extension," *Soft robotics*, vol. 6, no. 1, pp. 95–108, 2019.

[34] J. Xiao and R. Vatcha, "Real-time adaptive motion planning for a continuum manipulator," in *Proc. IEEE/RSJ Int. Conf. Intel. Robot. Syst.*, Taipei, Taiwan, 2010, pp. 5919–5926.

[35] J. D. Greer, L. H. Blumenschein, A. M. Okamura, and E. W. Hawkes, "Obstacle-aided navigation of a soft growing robot," in *2018 IEEE International Conference on Robotics and Automation (ICRA).* IEEE, 2018, pp. 4165–4172.

[36] I. Walker, "Continuum robot appendages for traversal of uneven terrain in in-situ exploration," in *Proc. IEEE Aerospace Conf.*, Big Sky, MT, 2011, pp. 1–8.

[37] D. Nahar, P. Yanik, and I. Walker, "Robot tendrils: Long, thin continuum robots for inspection in space operations," in *Proc. IEEE Aerospace Conference*, Big Sky, MT, 2017, pp. 1–8.

[38] D. Lane, B. Davies, G. Robinson, D. O'Brien, J. Sneddon, E. Seaton, and A. Elfstrom, "Aspects of the design and development of a subsea dextrous grasping system," *IEEE Jour. Ocean. Eng.*, vol. 24, no. 1, pp. 96–111, Jan. 1999.

[39] H. Tsukagoshi, A. Kitagawa, and M. Segawa, "Active hose: An artificial elephant's nose with maneuverability for rescue operations," in *Proc. IEEE Int. Conf. Robot. Autom.*, Seoul, South Korea, 2001, pp. 2454–2459.

[40] G. Immega and K. Antonelli, "The ksi tentacle manipulator," in *Proc. IEEE Int. Conf. Robot. Autom.*, Nagoya, Japan, 1995, pp. 3149–3154.

[41] R. V. Bostelman, J. S. Albus, and R. E. Graham, "Robocrane and emma applied to waste storage tank remediation," in *American Nucelar Society Seventh Topical Meeting on Robotics and Remote Systems*, vol. 2, 1997, pp. 708–713.

[42] I. A. Gravagne, "Design, analysis and experimentation: the fundamentals of continuum robotic manipulators," Ph.D. dissertation, Clemson University, 2002.

[43] B. Jones and I. Walker, "Kinematics for multisection continuum robots," *IEEE Trans. Robot.*, vol. 22, no. 1, pp. 43–57, Feb. 2006.

[44] M. Hannan and I. Walker, "Analysis and experiments with an elephant's trunk robot," *Advanced Robotics*, vol. 15, no. 8, pp. 847–858, Aug. 2001.

[45] R. Webster III and B. A. Jones, "Design and kinematic modeling of constant curvature continuum robots," *Int. Jour. Robots. Res.*, vol. 29, no. 13, pp. 1661–1683, Jul. 2010.

[46] R. Murray, Z. Li, and S. Sastry, *A Mathematical Introduction to Robotic Manipulation.* Boca Raton, FL: CRC Press, 1993.

[47] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Dynamics and Control.* New York, NY: John Wiley and Sons, 2005.

[48] H. Mochiyama and T. Suzuki, "Dynamic modeling of a hyper-flexible manipulator," in *Proc. SICE Annual Conf.*, Osaka, Japan, 2002, pp. 1505–1510.

[49] ——, "Kinematics and dynamics of a cable-like hyper-flexible manipulator," in *Proc. IEEE Int. Conf. Robot. Autom.*, Taipei, Taiwan, 2003, pp. 3672–3677.

[50] H. Mochiyama, "Whole-arm impedance of a serial-chain manipulator," in *Proc. IEEE Int. Conf. Robot. Autom.*, Seoul, Korea, 2001, pp. 2223–2228.

[51] I. Godage, E. Guglielmino, D. Branson, G. MedranoCerda, and D. Caldwell, "Novel modal approach for kinematics of multisection continuum arms," in *Proc. IEEE/RSJ Int. Conf. Intel. Robot. Syst.*, San Francisco, CA, 2011, pp. 1093–1098.

[52] I. Godage, D. Branson, E. Guglielmino, G. MedranoCerda, and D. Caldwell, "Shape function-based kinematics and dynamics for variable-length continuum robotic arms," in *Proc. IEEE Int. Conf. Robot. Autom.*, Shanghai, China, 2011, pp. 452–457.

[53] W. Felt, M. T. andT. Allen, G. Hein, J. Pompa, K. Albert, and D. Remy, "An inductance-based sensing system for bellows-driven continuum joints in soft robots," in *Robotics: Science and Systems*, The Hague, Netherlands, 07 2017.

[54] T. F. Allen, L. Rupert, T. R. Duggan, G. Hein, and K. Albert, "Closed-form non-singular constant-curvature continuum manipulator kinematics," in *2020 3rd IEEE International Conference on Soft Robotics (RoboSoft).* IEEE, 2020, pp. 410–416.

[55] P. Sears and P. Dupont, "A steerable needle technology using curved concentric tubes," in *Proc. IEEE/RSJ Int. Conf. Intel. Robot. Syst.*, Beijing, 2006, pp. 2850–2856.

[56] R. J. Webster III, J. S. Kim, N. J. Cowan, G. S. Chirikjian, and A. M. Okamura, "Nonholonomic modeling of needle steering," *Int. Jour. Robots. Res.*, vol. 25, no. 5-6, pp. 509–525, Jul. 2006.

[57] D. Rucker and R. Webster III, "Deflection-based force sensing for continuum robots: A probabilistic approach," in *Proc. IEEE/RSJ Int. Conf. Intel. Robot. Syst.*, San Francisco, CA, 2011, pp. 3764–3769.

[58] D. Trivedi, A. Lofti, and C. Rahn, "Geometrically exact dynamic models for soft robotic manipulators," in *Proc. IEEE/RSJ Int. Conf. Intel. Robot. Syst.*, San Diego, CA, 2007, pp. 1497–1502.

[59] F. Renda and C. Laschi, "A general mechanical model for tendon-driven continuum manipulators," in *Proc. IEEE Int. Conf. Robot. Autom.*, St. Paul, Minnesota, 2012, pp. 3813–3818.

[60] G. S. Chirikjian, "Hyper-redundant manipulator dynamics: A continuum approximation," *Advanced Robotics*, vol. 9, no. 3, pp. 217–243, 1994.

[61] E. Tatlicioglu, I. Walker, and D. Dawson, "Dynamic modeling for planar extensible continuum robot manipulators," in *Proc. IEEE Int. Conf. Robot. Autom.*, Rome, Italy, 2007, pp. 1357–1362.

[62] I. Gravagne, C. D. Rahn, and I. D. Walker, “Large deflection dynamics and control for planar continuum robots,” *IEEE Trans. Mecha.*, vol. 8, no. 2, pp. 299–307, Jun. 2003.

[63] B. He, Z. Wang, Q. Li, H. Xie, and R. Shen, “An analytic method for the kinematics and dynamics of a multiple-backbone continuum robot,” *International Journal of Advanced Robotic Systems*, vol. 10, no. 1, p. 84, 2013.

[64] 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, 2011.

[65] F. Boyer, V. Lebastard, F. Candelier, and F. Renda, “Dynamics of continuum and soft robots: A strain parameterization based approach,” *IEEE Transactions on Robotics*, 2020.

[66] S. M. H. Sadati, S. E. Naghibi, I. D. Walker, K. Althoefer, and T. Nanayakkara, “Control space reduction and real-time accurate modeling of continuum manipulators using ritz and ritz–galerkin methods,” *IEEE Robotics and Automation Letters*, vol. 3, no. 1, pp. 328–335, 2017.

[67] R. Kang, A. Kazakidi, E. Guglielmino, D. Branson, D. Tsakiris, J. Ekaterinaris, and D. Caldwell, “Dynamic modeling of a hyper-redundant octopus-like manipulator for underwater applications,” in *Proc. IEEE/RSJ Int. Conf. Intel. Robot. Syst.*, San Francisco, CA, 2011, pp. 4054–4059.

[68] G. Gallot, O. Ibrahimand, and W. Khalil, “Dynamic modeling and simulation of a 3-d eel-like robot,” in *Proc. IEEE Int. Conf. Robot. Autom.*, Rome, Italy, 2007, pp. 1486–1491.

[69] A. Amouri, A. Zaatri, and C. Mahfoudi, “Dynamic modeling of a class of continuum manipulators in fixed orientation,” *Journal of Intelligent & Robotic Systems*, vol. 91, no. 3, pp. 413–424, 2018.

[70] A. Marchese, R. Tedrake, and D. Rus, “Dynamics and trajectory optimization for a soft spatial fluidic elastomer manipulator,” in *Proc. IEEE Int. Conf. Robot. Autom.*, Seattle, WA, 2015, pp. 2528–2535.

[71] W. S. Rone and P. Ben-Tzvi, “Continuum robot dynamics utilizing the principle of virtual power,” *IEEE Transactions on Robotics*, vol. 30, no. 1, pp. 275–287, 2013.

[72] C. Wang, J. Wagner, C. G. Frazelle, and I. D. Walker, “Continuum robot control based on virtual discrete-jointed robot models,” in *IECON 2018-44th Annual Conference of the IEEE Industrial Electronics Society*. IEEE, 2018, pp. 2508–2515.

[73] C. Wang, C. G. Frazelle, J. R. Wagner, and I. Walker, “Dynamic control of multi-section three-dimensional continuum manipulators based on virtual discrete-jointed robot models,” *IEEE/ASME Transactions on Mechatronics*, 2020.

[74] T. B. Sheridan, *Telerobotics, Automation, and Human Supervisory Control.* Cambridge, MA, USA: MIT Press, 1992.

[75] P. F. Hokayem and M. W. Spong, “Bilateral teleoperation: An historical survey,” *Automatica*, vol. 42, no. 12, pp. 2035–2057, 2006.

[76] M. Csencsits, B. Jones, and W. McMahan, “User interfaces for continuum robot arms,” in *Proc. IEEE Int. Conf. Intell. Robot. Sys*, Edmonton, Canada, 2005, pp. 3011–3018.

[77] A. Kapadia, I. Walker, and E. Tatlicioglu, "Teleoperation control of a redundant continuum manipulator using a non-redundant rigid-link master," in *Proc. IEEE Int. Conf. Intell. Robot. Sys.*, Algarve, Portugal, 2012, pp. 3105–3110.

[78] C. Frazelle, A. Kapadia, K. Fry, and I. Walker, "Teleoperation mappings from rigid link robots to their extensible continuum counterparts," in *Proc. IEEE Int. Conf. Robot. Autom.*, Stockholm, Sweden, 2016.

[79] C. G. Frazelle, A. Kapadia, and I. Walker, "Developing a kinematically similar master device for extensible continuum robot manipulators," *Journal of Mechanisms and Robotics*, vol. 10, no. 2, 2018.

[80] H.-S. Yoon, S. Oh, J. Jeong, S. Lee, K. Tae, K.-C. Koh, and B. Yi, "Active bending robot endoscope system for navigation through sinus area," in *Proc. IEEE/RSJ Int. Conf. Intel. Robot. Syst.*, San Francisco, CA, 2011, pp. 967–972.

[81] H. El-Hussieny, U. Mehmood, Z. Mehdi, S.-G. Jeong, M. Usman, E. W. Hawkes, A. M. Okamura, and J.-H. Ryu, "Development and evaluation of an intuitive flexible interface for teleoperating soft growing robots," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 4995–5002.

[82] M. Mihelj and J. Podobnik, *Haptics for Virtual Reality and Teleoperation*. Dordrecht, Netherlands: Springer Science+Business Media, 2012.

[83] A. Okamura, "Haptic feedback in robot-assisted minimally invasive surgery," *Current Opinion in Urology*, vol. 19, no. 1, pp. 102–107, 2009.

[84] A. Talasaz, R. V. Patel, and M. D. Naish, "Haptics-enabled teleoperation for robot-assisted tumor localization," in *2010 IEEE International Conference on Robotics and Automation*, 2010, pp. 5340–5345.

[85] S. Hirche and M. Buss, "Human-oriented control for haptic teleoperation," *Proceedings of the IEEE*, vol. 100, no. 3, pp. 623–647, 2012.

[86] D. Escobar-Castillejos, J. Noguez, L. Neri, A. Magana, and B. Benes, "A review of simulators with haptic devices for medical training," *Journal of medical systems*, vol. 40, no. 4, p. 104, 2016.

[87] R. Scott, A. Kapadia, and I. Walker, "Intuitive interfaces for teleoperation of continuum robots," in *International Conference on Applied Human Factors and Ergonomics*. Springer, 2018, pp. 77–89.

[88] A. Nordmann, M. Rolf, and S. Wrede, "Software abstractions for simulation and control of a continuum robot," in *International Conference on Simulation, Modeling, and Programming for Autonomous Robots*. Springer, 2012, pp. 113–124.

[89] C. Yang, J. Yang, X. Wang, and B. Liang, "Control of space flexible manipulator using soft actor-critic and random network distillation," in *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2019, pp. 3019–3024.

[90] H. el-Hussieny. (2021, April) continuum_robot [Source code]. https://github.com/elhussieny/continuum_robot.

[91] J. Till. (2021, April) ContinuumRobotExamples [Source code]. Https://github.com/JohnDTill/ContinuumRobotExamples.

[92] S. H. Sadati, S. E. Naghibi, A. Shiva, B. Michael, L. Renson, M. Howard, C. D. Rucker, K. Althoefer, T. Nanayakkara, S. Zschaler *et al.*, "Tmtdyn: A matlab package for modeling and control of hybrid rigid–continuum robots based on discretized lumped systems and reduced-order models," *The International Journal of Robotics Research*, p. 0278364919881685, 2019.

[93] ROS. (2021, April) Ros robots. https://robots.ros.org/all/.

[94] K. Wu, L. Wu, and H. Ren, "Motion planning of continuum tubular robots based on centerlines extracted from statistical atlas," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 5512–5517.

[95] Z. Hawks, C. Frazelle, K. E. Green, and I. D. Walker, "Motion planning for a continuum robotic mobile lamp: Defining and navigating the configuration space," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 2559–2566.

[96] A. Kuntz, A. W. Mahoney, N. E. Peckman, P. L. Anderson, F. Maldonado, R. J. Webster, and R. Alterovitz, "Motion planning for continuum reconfigurable incisionless surgical parallel robots," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 6463–6469.

[97] A. Ataka, P. Qi, H. Liu, and K. Althoefer, "Real-time planner for multi-segment continuum manipulator in dynamic environments," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 4080–4085.

[98] A. Kuntz, M. Fu, and R. Alterovitz, "Planning high-quality motions for concentric tube robots in point clouds via parallel sampling and optimization," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 2205–2212.

[99] S. Collins, A. Ruina, R. Tedrake, and M. Wisse, "Efficient bipedal robots based on passive-dynamic walkers," *Science*, vol. 307, no. 5712, pp. 1082–1085, 2005.

[100] N. Kohl and P. Stone, "Policy gradient reinforcement learning for fast quadrupedal locomotion," in *IEEE International Conference on Robotics and Automation*, 2004, pp. 2619–2624.

[101] A. Y. Ng, A. Coates, M. Diel, V. Ganapathi, J. Schulte, B. Tse, E. Berger, and E. Liang, "Autonomous inverted helicopter flight via reinforcement learning," in *International Symposium on Experimental Robotic*, 2003, pp. 363–372.

[102] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[103] T. Haarnoja, A. Zhou, S. Ha, J. Tan, G. Tucker, and S. Levine, "Learning to walk via deep reinforcement learning," in *Robotics: Science and Systems*, 2018.

[104] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke, "Sim-to-real: Learning agile locomotion for quadruped robots," in *Robotics: Science and Systems*, 2018.

[105] Z. Xie, P. Clary, J. Dao, P. Morais, J. Hurst, and M. van de Panne, "Iterative reinforcement learning based design of dynamic locomotion skills for cassie," in *Conference on Robot Learning*, 2019.

[106] M. Zhang, X. Geng, J. Bruce, K. Caluwaerts, M. Vespignani, V. SunSpiral, P. Abbeel, and S. Levine, "Deep reinforcement learning for tensegrity robot locomotion," in *IEEE International Conference on Robotics and Automation*, 2017, pp. 634–641.

[107] S. Satheeshbabu, N. K. Uppalapati, G. Chowdhary, and G. Krishnan, "Open loop position control of soft continuum arm using deep reinforcement learning," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 5133–5139.

[108] 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, 2018.

[109] T. Mahl, A. Hildebrandt, and O. Sawodny, "A variable curvature continuum kinematics for kinematic control of the bionic handling assistant," *IEEE Transactions on Robotics*, vol. 30, no. 4, pp. 935–949, 2014.

[110] I. Gravagne and I. D. Walker, "Manipulability, force, and compliance analysis for planar continuum manipulators," *IEEE Trans. Robots. Autom.*, vol. 18, no. 3, pp. 263–273, Jun. 2002.

[111] D. C. Rucker, B. A. Jones, and R. J. Webster III, "A geometrically exact model for externally loaded concentric-tube continuum robots," *IEEE Transactions on Robotics*, vol. 26, no. 5, pp. 769–780, 2010.

[112] C. G. Frazelle, A. D. Kapadia, and I. D. Walker, "A haptic continuum interface for the teleoperation of extensible continuum manipulators," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1875–1882, 2020.

[113] C. Frazelle, J. Rogers, I. Karamouzas, and I. Walker, "Optimizing a continuum manipulator's search policy through model-free reinforcement learning," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Las Vegas, NV, 2020, pp. 5564–5571.

[114] C. Frazelle, I. Walker, A. AlAttar, and P. Kormushev, "Kinematic-model-free control for space operations with continuum manipulators," in *2021 IEEE Aerospace Conf.*, Big Sky, MT, 2021, pp. 1–11.

[115] R. S. Ball, *A Treatise on the Theory of Screws.* Cambridge university press, 1998.

[116] J. Winches. (2017, August) Slack rope detectors. [Online]. Available: http://www.jeamar.com/blocks/slack-rope-detectors/

[117] R. Coulson, M. Robinson, M. Kirkpatrick, and D. R. Berg, "Design and preliminary testing of a continuum assistive robotic manipulator," *Robotics*, vol. 8, no. 4, p. 84, 2019.

[118] H. In, H. Lee, U. Jeong, B. B. Kang, and K.-J. Cho, "Feasibility study of a slack enabling actuator for actuating tendon-driven soft wearable robot without pretension," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 1229–1234.

[119] A. Yeshmukhametov, K. Koganezawa, and Y. Yamamoto, "A novel discrete wire-driven continuum robot arm with passive sliding disc: Design, kinematics and passive tension control," *Robotics*, vol. 8, no. 3, p. 51, 2019.

[120] A. Chawla, C. Frazelle, and I. Walker, "A comparison of constant curvature forward kinematics for multisection continuum manipulators," in *2018 Second IEEE International Conference on Robotic Computing (IRC)*. IEEE, 2018, pp. 217–223.

[121] Arduino. (2016, May) Arduino due. [Online]. Available: https://store.arduino.cc/usa/due

[122] O. S. R. Foundation. (2020, February) Gazebo: Robot simulation made easy. Http://gazebosim.org/.

[123] M. A. Diftler, J. Mehling, M. E. Abdallah, N. A. Radford, L. B. Bridgwater, A. M. Sanders, R. S. Askew, D. M. Linn, J. D. Yamokoski, F. Permenter *et al.*, "Robonaut 2-the first humanoid robot in space," in *2011 IEEE international conference on robotics and automation.* IEEE, 2011, pp. 2178–2183.

[124] M. Hannan and I. Walker, "Kinematics and the implementation of an elephant trunk manipulator and other continuum style robots," *J. Robot. Sys.*, vol. 20, no. 2, pp. 45–63, Feb. 2003.

[125] F. Riewe, "Nonconservative lagrangian and hamiltonian mechanics," *Physical Review E*, vol. 53, no. 2, p. 1890, 1996.

[126] J. Y. S. Luh, M. W. Walker, and R. P. C. Paul, "On-line computational scheme for mechanical manipulators," *ASME. J. Dyn. Sys., Meas., Control*, vol. 102, no. 2, p. 69–76, 1980.

[127] S. Davis and D. G. Caldwell, "Braid effects on contractile range and friction modeling in pneumatic muscle actuators," *The International Journal of Robotics Research*, vol. 25, no. 4, pp. 359–369, 2006.

[128] I. S. Godage, D. T. Branson, E. Guglielmino, and D. G. Caldwell, "Pneumatic muscle actuated continuum arms: Modelling and experimental assessment," in *2012 IEEE International Conference on Robotics and Automation.* IEEE, 2012, pp. 4980–4985.

[129] C. H. Papadimitriou and J. N. Tsitsiklis, "The complexity of markov decision processes," *Mathematics of operations research*, vol. 12, no. 3, pp. 441–450, 1987.

[130] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction.* MIT press, 1998.

[131] M. Tonapi, I. S. Godage, A. M. Vijaykumar, and I. Walker, "A novel continuum robotic cable aimed at applications in space," *Advanced Robotics*, vol. 29, no. 13, pp. 861–875, Jul. 2015.

[132] P. Kormushev, Y. Demiris, and D. G. Caldwell, "Encoderless position control of a two-link robot manipulator," in *Proc. IEEE Int. Conf. Robot. Autom.*, Seattle, Washington, 2015, pp. 943–949.

[133] ——, "Kinematic-free position control of a 2-dof planar robot arm," in *Proc. IEEE/RSJ Int. Conf. Intel. Robot. Syst.*, Hamburg, Germany, 2015, pp. 5518–5525.

[134] A. AlAttar and P. Kormushev, "Kinematic-model-free orientation control for robot manipulation using locally weighted dual quaternions," *Robotics*, vol. 9, no. 4, p. 76, 2020.

[135] Microsoft. (2020, October) Kinect for windows. Https://developer.microsoft.com/en-us/windows/kinect/.

[136] A. Melingui, O. Lakhal, B. Daachi, J. B. Mbede, and R. Merzouki, "Adaptive neural network control of a compact bionic handling arm," *IEEE/ASME Transactions on Mechatronics*, vol. 20, no. 6, pp. 2862–2875, 2015.

[137] M. Wooten, C. Frazelle, I. D. Walker, A. Kapadia, and J. H. Lee, "Exploration and inspection with vine-inspired continuum robots," in *2018 IEEE International Conference on Robotics and Automation (ICRA).* IEEE, 2018, pp. 1–5.

[138] 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, 2019.

[139] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[140] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," *arXiv preprint arXiv:1801.01290*, 2018.