Rapidly-exploring Random Trees for multi-section Continuum Arms

Abstract

Continuum Arms are devices inspired by the biology of physical appendages that serve as a compromise between soft and rigid robots. These manipulators are constructed using both compliant and rigid components such as a pneumatic muscle actuators and backbones. By chaining several of these robots serially, a multi-section arm can be constructed that has many degrees of freedom and, by extension, a wide number of shapes and uses. This design, though useful, presents several challenges.

We study the problem of path planning for multi-section continuum arms. This task requires that the tip of a robotic arm move from some starting position to a goal position without colliding with obstacles and while obeying physical constraints of the arm. Continuum arms are a recent area of study, so there does not exist extensive work covering this problem. However, the hyper-redundancy of these manipulators causes issues when implementing conventional path planning approaches such as trajectory tracing via inverse kinematics. Such approaches run into issues of knotting and convergence to local minima. Additionally, the quality of such paths is important as little perturbation in the motion of an arm could lead to unreliable movement that is difficult to track. Many attempts have been made to solve this problem, but it remains a largely open problem. Some have attempted AI based approaches that require low resources on simple continuum arms, but these methods have not been shown to extend well to more complicated models. Others have attempted deterministic algorithms, but these are often prohibitively slow and rely on precomputed data.

We propose a path planning scheme based on Rapidly-Exploring Random Trees (RRT) to produce high quality paths quickly. Conventional RRT does not behave well for continuum arms, as the hyper-redundancy makes exploring all configurations of the arm slow. We hypothesize that the use of Jacobian-based velocity approximations paired with a procedure that "steers" the exploration in the direction of the goal could produce high-quality paths that are reliable with minimal computation time. Such methodologies have been applied to potential field approaches before, as they are used as an attractive force between a tip and a target. For our approach, this would allow us to add a heuristic to RRT and produce a combination has not been attempted for continuum arms. This would not only speed up exploration significantly but would create smooth paths and would maintain the flexibility of not using precomputed configurations. We validate the smoothness of these path by passing the results into a dynamic simulator. Using this simulator, we can determine the quality of a path based on its overall running time. A path that runs faster is smoother and therefore higher quality. We will also run experimental simulations on the prototype arm to verify that they proposed and standard approaches behave normally on a physical continuum arm. Each arm will replicate a trajectory produced in simulations and the difference between expected and produced trajectories will be recorded.

Introduction and Overview

Robotic Model

Rigid robots, most known for their application in manufacturing, have a high degree of precision and can handle large payloads. However, these devices, typically made of metals, are not safe around humans as their great strength and non-flexible build can prove fatal when uncontrolled. Furthermore, they have difficultly handling fragile materials, and are ill-suited for human-robot collaborative projects. Additionally, these devices are limited in their shape, as rigid materials cannot deform. Soft robots, on the other hand, are a style of robotic devices that rely on soft material construction to complete tasks. Compliant materials are notable for their ability to deform into novel shapes and for their relative safety around humans (as it is far harder to be injured by a non-rigid body). By using compliant materials, these devices are well suited for human interaction. Additionally, they are often designed to complete delicate tasks like fruit picking. However, they are notorious for their low payload as it can be difficult to carry much weight with flexible construction. Each style has clear downsides that prevent them from being general purpose manipulators.

Continuum arms are a style of robotic manipulators that are "bio-inspired". The purpose of such devices is to replicate the anatomies of physical appendages like an elephant's trunk. In pursuing this mimicry, continuum arms can serve as a middle ground between soft and rigid robotics. Usually, these devices combine rigid components for strength with compliant components for safety. One can think of this combination as the usage of muscle and bone. This provides them with a great deal of payload capabilities while also being safe enough to use around humans. Continuum arms are in this way perfect to serve as collaborative robots or co-bots. Their application ranges wildly from assistive manufacturing (such as welding) or in the medical field as an assistive hand. Additionally, smaller devices could be mounted to wheelchairs to provide increased mobility.

Path Planning and RRT

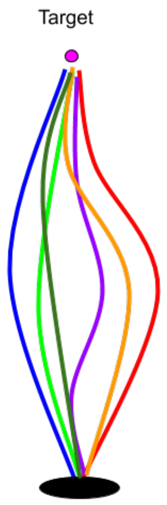

Figure 1

Robotic path planning is an arm from one place to another without colliding with obstacles and often to accomplish some task. For example, the problem of moving an arm towards and item, picking up the item, and then moving the item somewhere else is a path planning problem. This problem is also known as the piano mover's problem. We study this problem for a continuum arm in 3D space. Put more formally, we provide a target end position for the tip of the arm and a beginning shape. A path planning algorithm must then output a series of shapes or *configurations* of the arm that move it from the starting to ending position. A configuration is a set of numbers that defines the current state of the arm.

An open and challenging problem is path planning for continuum arms. Given their design, these arms have a high degree of redundancy in their shape. Figure 1 shows an example of this redundancy. This can make path planning difficult as there can be many "correct" positions to use at any moment. In finding path, we must make certain that the redundancy does not lead to low-quality paths. For example, if the arm oscillated between two wildly different shapes while moving, the device would quickly become unstable and would not be performing the task in a practical manner. As such, paths that are devised for continuum arms must consider the quality or "smoothness" of the path. Additionally, algorithms can be unreliable, meaning that it may not find a path even if one exists. For lower-complexity arms consisting of only 1 section, path planning is achievable using standard approaches. However as more sections are added, the arm becomes increasingly hyper-redundant and the problem becomes more difficult. To explore this topic, we must focus on a specific robotic model. Robotic models are an important piece of path planning background, as they can greatly influence the implementation details. For the purposes of our research, we focus on a 3-section implementation of a continuum arm with both 6 degrees of freedom and hyper-redundant shapes.

Rapidly-Exploring Random Trees (RRT) is a common path planning algorithm for mobile robots. This approach randomly explores the space that the arm occupies outwards from a starting position. Through this exploration, a “tree” of potential configurations is generated. Once the random exploration finds the desired endpoint, the arm can simply traverse this tree from the starting to ending position. As an added benefit, the exploratory nature of means that it may find higher-quality paths if given more exploration time. In all applications, RRT is highly efficient, can execute exceedingly quickly, produces high-quality paths, and is very reliable, so it is widely used and studied in robotics.

In its standard implementation, RRT is not suited for continuum arms. The complexity of these devices means that exploration can be prohibitively slow. Our hypothesis is that RRT can be modified with Jacobian-based guiding procedure to produce both high-quality and reliable paths for continuum arms.

A Jacobian-based steering method would allow for rapid, guided exploration towards a desired target. In such a scheme, RRT would explore outwards towards the target, but we could use the Jacobian to base this search in a lower complexity setting. This would allow us to utilize the benefits of RRT (efficiency, reliability) without any of its issues (slow exploration). Jacobians are efficient to calculate and it would eliminate the need for consideration of many spatial shapes, as the shape that best suited the current movement of the arm would be selected. We hypothesize that the combination of RRT with a Jacobian-based steering method could reduce complexity and improve smoothness in exploration while also maintaining the benefits of efficiency and versatility already seen in RRT.

To test our hypothesis, we intend to create an implementation of this modified RRT procedure. Once this procedure has been properly developed, we can test its performance to verify our claims. Often, path planning algorithms do not behave expectedly when first implemented, so it would likely take trial and error to eliminate minor issues with our scheme that are not yet evident. After implementing modified RRT, several other algorithms would also need to be implemented as a basis for comparison without approach. There already exists a simulated test environment that can be used to demonstrate each of these algorithms and compare their performance. A simulated test would involve tasking the arm to move from one tip position to another. Next, these algorithms would be run though a dynamic simulator. This simulator adds forces like gravity, momentum, and robotic controller to simulate the behavior of the arm more closely on a physical device. At this moment, there are no existing robotic controllers that allow for real-time control of an arm. However, in order to show that our proposed path produces feasible results, the arm will simulate the produced paths very slowly on the physical arm. Each path will have its motion recorded and it will be compared against the simulated results to see how well our theory matches up to reality.

Related Work

Continuum Arms are new devices and, as such, they are a very hot topic of research. Much time is being spent into developing controllers for these devices so that they can be controlled in real-time. The current model of continuum arm we use was first devised in [1] by Godage et al. They devised modal kinematic to allow for angle-based parameterization of the arm's configuration. This model was extended to include multi-section arms in [2]. Lastly, the dynamics used in a dynamic simulator for this robotic model was created in [3].

However, the purpose of this paper is algorithmic, so we must focus on the path planning contributions. RRT is a very foundational algorithm, so the proposed modification to RRT could be foundational to the field. There are many existing algorithms for solving the path planning problem. However, given the diversity of robotic models, each algorithm is suited to different needs. Additionally, algorithms that work for mobile robots like self-driving cars may not work well for arm-shaped robots. For our purposes, we focus on path planning algorithms that are suited for arm-shaped robots in 3D space. Common approaches include inverse kinematics trajectory tracing, potential fields, and A*. Potential field is a numerical process by which a figurative magnet is placed on the tip of the arm and a magnet of opposite polarity is placed on the target. Obstacles have figurative magnet inside that have the same polarity as the arm's tip, so the arm is pulled towards the target while repelling obstacles. This approach was demonstrated in [4] but did so in a highly restrictive case. In other research such as [5], the authors demonstrated that potential fields did not work well in more complicated test cases. A* is an exploration algorithm that is commonly used for solving mazes. In this approach, a search occurs outward that is guided to the goal by some meaningful metric like distance. Inverse kinematics is the process of finding a configuration that places the tip of the arm in some desired location. To achieve path planning with this approach, a set of tip points is produced, and inverse kinematics is tasked with outputting all the necessary arm shapes to touch each point. In the case of each of these algorithms, a path can be quite easily produced that is technically correct. However, they often produce extremely low-quality paths as the algorithms lack the ability to distinguish between good and bad shapes.

There have also been attempts at path planning for multi-section continuum arms by more novel means in recent years. The authors in [6] propose to do path planning through a graph-theoretic approach that relies on an offline lookup table. However, this approach required significant time and space resources, as every shape that the robot could assume had to be stored on this. This led to a RAM requirement that was almost 32GB in addition to slow computation time. In [7], the authors proposed an improvement to this graph-theoretic algorithm that used inverse kinematics to reduce the RAM requirement and increase flexibility in the number of shapes used during calculation. Though effective, the runtimes of such algorithms were in the magnitude of hours which is far too slow for real time control.

Lastly, there has been some research into single section arms, but this has not been extended to multiple sections. Fang et al. [8] created a Jacobian-based approach to inverse kinematics that allowed for faster and more accurate mappings. Though this was shown to be effective, it was only demonstrated on a single section. Additionally, the authors in [9] demonstrated a reinforcement learning scheme for control of a continuum arm. Again, this was only applied to a single section and has not yet been shown to work for a multi-section arm.

Overall, much work struggles to handle multi-section continuum arms. Either an approach is created that works well for single sections, or it works well for multiple sections with major downsides (runtime, storage, etc). We aim to find the "best of both worlds" by combining the efficiency and versatility of the standard RRT while also adding the ability to handle hyper-redundancy and smoothness considerations.

Research Design and Methodology

Overview

Summary of Metrics:

1. **Calculation Runtime**
2. **Kinematic Change in Shape**
3. **Kinematic Change in Configuration**
4. **Dynamic Time-To completion**
5. **Experimental Shape Error**

Figure 2

A significant amount of time will first need to be spend working on an implementation of RRT. There are several components to making this work that will take significant time. Following the implementation of RRT, much work needs to be done to prove the performance of our algorithm. At this moment, real time control of a continuum arm is not feasible. Some have attempted to do so, but at very low speed. This is useful as a theoretical lower bound, but it is far from real-time control. Journals are often very critical of algorithms that are not validated sufficiently, so a large amount of comparison needs to be performed. Since very few path planning algorithms exist for continuum arms, we will need to implement several commonplace path planning algorithms as a baseline for comparison. As discussed, Potential fields, trajectory tracing, A*, regular RRT, and more (discussed below) can be implemented for a multi-section continuum arm. Following these implementations and thorough debugging, we can set out to compare the algorithms. There are 3 phases of testing with #2 being the most important. A summary of our metrics is provided in Figure 2. First, these algorithms will be discussed in a simulated, kinematic environment. The kinematic simulator is a basic comparison between algorithms that simply simulated the changing state of the arm over time. This can be a useful baseline of comparison as issues in this stage are compounded in the second. Next, these algorithms will be tested using a dynamic simulator. As discussed, this dynamic simulator was created in [3] and will be used to compare the output of each algorithm when forces are applied to a simulated arm. When forces are added, sequences of configurations that were feasible in a kinematic simulator can be made completely unusable. Often, unusable sequences are the result of low-quality paths, so this is the state of the art when it comes to comparison. Lastly, to add physical validation, these algorithms will be simulated on a prototype arm at a significantly reduced speed. Since real-time control is not possible, we will use the state of the art in controlling to run precalculated paths at the greatest speed possible. These paths will be tracked using sensors and fed into a comparison. This can serve as a third comparison, further demonstrating the versatility of certain algorithms. We now discuss each phase in greater detail.

Robotic Model

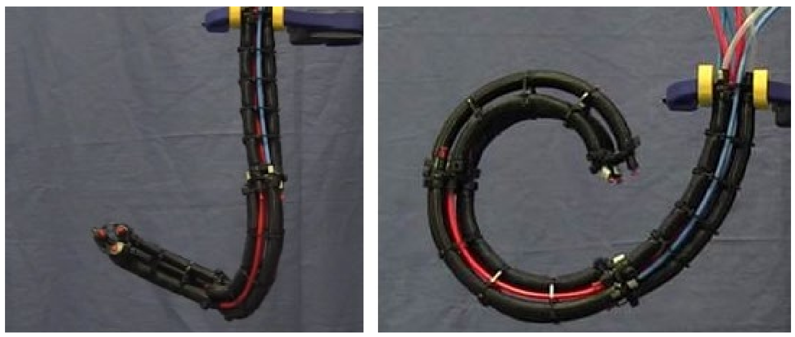
Figure 3

Our prototype model is a multi-section continuum arm. These devices are made of small "sections" consisting of a bundle of pneumatic muscle actuators joined around a rigid backbone. This backbone is made of plastic and is designed to look like a spine a maximum bend angle. Several sections can be joined together serially to create more interesting and useful forms but at the cost of complex control. Our prototype has 3 sections as it is an interesting mix of utility and complexity. These pneumatic muscle actuators are filled with a liquid or with air. As they are filled, they expand, causing arcing motions. As a result, these devices can achieve complex shapes. Figure 3 above shows an example of our prototype actuated into two different shapes.

Algorithm

What follows is a high-level description of our proposed algorithm. First, a starting configuration and target tip position are selected. We then begin the construction of random tree with disconnected nodes inserted for the start and target. These nodes consist of a configuration and the corresponding tip position. Now, we start growing the tree. Every other node in this tree is placed randomly. First, we select a random point within reach of the arm's tip. Next, we find the nearest node to this random point that is already in the tree. We now place a new node that is within some small distance of the tree in the direction of our random point. Upon placing this random point in the tree, we use Jacobian methods to find the corresponding configuration. This Jacobian considers the previous position and configuration of the arm along with the intermediate tip position. We then use the Jacobian to find the new configuration for intermediate tip position. The node in the tree is updated slightly if needed. Since every other node is placed randomly, the non-random nodes are placed in the direction of the target. We use the same Jacobian methods to find the corresponding configuration. Whenever the tree is connected to the node corresponding to the target tip position, a path is found, and we follow the tree to output a sequence of configurations.

Implementations

All our algorithms will be implemented in C++ using the same libraries and tools (where relevant). C++ is a highly performant language that is widely used for robotics. Our modified RRT and any other algorithms will need to be implemented before any testing can occur. Given the inherently exploratory nature of our RRT implementation, we expect that many bugs or unexpected behaviors will pop-up during the development of the algorithm. In any case, we will slightly modify our algorithm until a sufficiently stable one is found.

Following a successful implementation of our modified algorithm, we will also be implementing standard and novel algorithms as a basis of comparison. For standard algorithms, we will be implementing **regular RRT**, **potential fields, trajectory tracing** (using inverse kinematics), and **A***. Additionally, we will be implementing the **Brute-Force**, **Sequential-IK**, and **Global-IK schemes** from [7]. These algorithms are the best for our current model and serve as an important benchmark. Though some of these algorithms have pre-existing implementations, we will be implementing each from scratch. We do so in order to be able to make fair claims about the runtime of each. By creating every algorithm on our own, we can minimize confounding variables. In this way, we can use the same compiler, library tools, and optimization techniques. We will also ensure that tests are run on identical hardware, again to minimize differences between algorithms.

Simulators

A kinematic simulator uses the sequence of configurations in a path to draw the motion of the continuum arm. This is done without forces like gravity, inertia, or friction. From this simulation, two useful metrics can be found. First, we can calculate the overall movement of the arm (by comparing the change in shape over time). This is a useful metric as less change in the shape of the arm is a high-quality trait. Additionally, we can compare the total change in configuration over time. A lower value for this metric indicates overall less movement.

The dynamic simulator for our robotic arm was created in [3] will be used to validate our algorithms. These simulators attempt to control the arm using a kinematic controller in a setting with forces like gravity, friction, inertia, and velocity. The goal of the dynamic simulator is "track" or follow (and correct) the movement of the arm. Since velocity is the easiest way to determine if a path is easily tracked, we use the overall time-to-completion of a path as a metric for smoothness, quality, and ease of tracking. We will run each algorithm through the dynamic simulator and modify velocity until the tracking error is within an acceptable margin of 0.1cm.

Tests

To verify the validity of our proposed algorithm, simulators must be used as a basis of comparison. There are two types of simulator: kinematic and dynamic. To test our algorithm, sample cases will be created. At first, 10,000 sample scenarios will be created using random generation. Each case will consist of a starting configuration, a target tip point, and 1 to 10 (randomly decided) spherical obstacles placed randomly. Each algorithm will be tasked from moving the arm from the initial shape until the tip of the arm is in the desired location. If the arm enters an invalid state or it collides with and obstacle, the trial will be considered a failure. Paths will be generated with each algorithm and then run through our two simulators. Relevant metrics will be captured and recorded. We will also record the amount of time that it takes each algorithm to produce output after being given input. This metric speaks to the efficiency of each algorithm, and we expect that our version of RRT will be among the fastest.

Physical Experimentation

Most robotics journals and conferences require some form of experimental validation of algorithms. As discussed earlier, this is not yet feasible in real-time given the state of the art. However, there do exist controllers for experimentation at a slow speed. To physically validate the output from our algorithms, 10 of our test cases will be selected at random, and each algorithm's output for those cases will be physically simulated on our prototype arm.

Sensors will be placed at the tip and midpoint of each section. The arm will then be tasked with replicating the pre-generated path. During the arm's movement, the sensors will record the difference between the requested path and the produced path. Each algorithm's performance will be recorded. Figure 4 shows an example of the experimental setup.

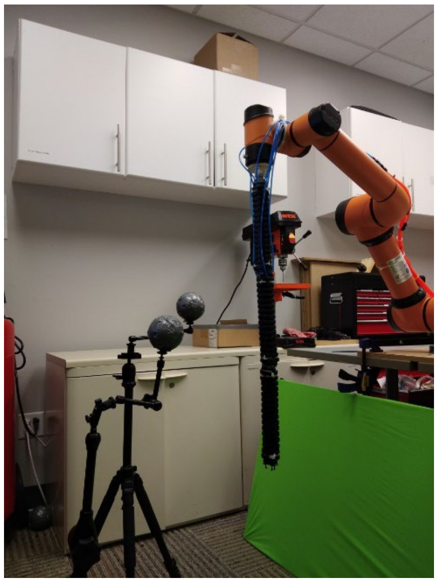
Figure 4

Basis of Comparison

See Figure 2 above for a summary of the metrics used for comparison. Our algorithm may or may not be the top performer metrics. Some metrics are only presented to demonstrate acceptable or non-acceptable performance. Runtime, for example, is not the most important metric being measured. However, if our proposed method is the slowest, we would have reasons to doubt the efficacy of our approach. The most important metrics are the dynamic simulator (4) and physical experimentation (5). This is followed by runtime (1) and then kinematic simulations (2,3). If there is some discrepancy between the results in the dynamic simulator and the kinematic simulator (for example, if modified RRT performs well in one or not the other) some sort of explanation must be provided. This is true too for the dynamic simulator and the physical experiment. If our algorithm can have the best dynamic and experimental smoothness while also having a low runtime, we will consider our hypothesis correct and near real-time calculation of smooth paths will have been achieved.

Plan of Work and Outcomes

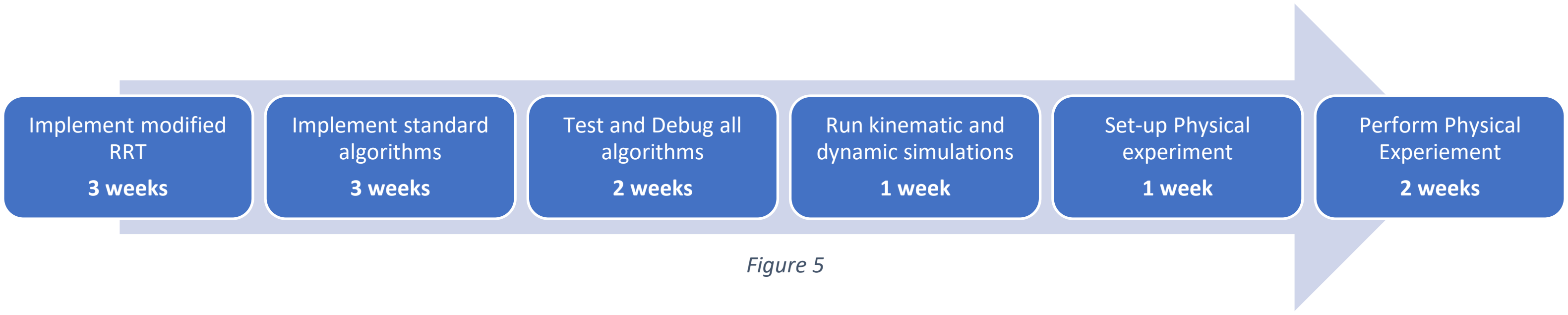
Implement modified RRT
3 weeks
Implement standard algorithms
3 weeks
Test and Debug all algorithms
2 weeks
Run kinematic and dynamic simulations
1 week
Set-up Physical experiment
1 week
Perform Physical Experiement
2 weeks

Figure 5

Figure 5 shows our expected timetable for this project. Implementation of the algorithms should be given 8 weeks, as the bulk of the work must be done in this step. This will be broken up into 3 segments: implementing our algorithm, implementing existing algorithms, and debugging. They will be given 3, 3, and 2 weeks, respectively. Next, we dedicate 4 weeks to testing. Since the simulators are already written, it should take only 1 week to get results. The remaining 3 will be used to set up the physical experiment (1 week) and run the experiments (2 weeks). We leave 1 week at the end as buffer time in case any other section runs long. **In total, we expect 13 weeks will be needed.**

We anticipate that our modified RRT will perform the best in all metrics except for runtime. Given the simplicity of algorithms like A*, it would be unreasonable to expect our modified RRT to outperform them. However, our approach should outperform Sequential-IK, global-IK, and Brute-Force, schemes noted for their slow performance.

Showing that our modified RRT could outperform all other algorithms in terms of path quality, while also being among the top performers in runtime would be a substantial outcome. Currently, no path planning scheme runs fast enough to be useful in real-time settings. RRT is a very efficient algorithm, so if our modified version produced the highest quality paths in a relatively short amount of time, we would be much closer to the goal of real-time calculation. By simulating our algorithms in each of the 3 settings, we can prove definitively the quality and reliability of our path planning scheme. Physical experimentation, in particular, goes a long way in proving the viability of our scheme. However, given the limitations of this test, the simulations are an effective way to demonstrate superiority. We will consider our results a success if our algorithm is at least 20% better than any other algorithm in the dynamic simulator metric and at least the median (or lower) in terms of the runtime metric.

Conclusion and Future Work

Following a successful result from our modified RRT, future work could focus on improving the runtime further either though distributed computing or parallelism (or both). Additionally, more modifications to RRT will likely become apparent during implementation that could be further explored. We also expect that future breakthroughs in controller technology for multi-section continuum arms would necessitate a revisiting of these tests to re-validate their performance. Though controller design is outside of the scope of this research, we may also attempt path planning schemes that are more directly tailored to a working controller. We would also like to attempt more challenging tasks like dynamic obstacles and object manipulation. Looking forward, we see many opportunities for multi-section continuum arms in healthcare and manufacturing, and we hope to continue improving path planning schemes so that these arms one day become a practical reality.

References

1 - I. S. Godage, E. Guglielmino, D. T. Branson, G. A. Medrano-Cerda and D. G. Caldwell, "Novel modal approach for kinematics of multisection continuum arms," 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2011, pp. 1093-1098, doi: 10.1109/IROS.2011.6094477.

2 - Godage, Isuru & Medrano-Cerda, Gustavo & Branson, David & Guglielmino, Emanuele & Caldwell, Darwin. (2015). Modal kinematics for multisection continuum arms. Bioinspiration & biomimetics. 10. 035002. 10.1088/1748-3190/10/3/035002.

3 - I. S. Godage, D. T. Branson, E. Guglielmino, G. A. Medrano-Cerda and D. G. Caldwell, "Dynamics for biomimetic continuum arms: A modal approach," 2011 IEEE International Conference on Robotics and Biomimetics, 2011, pp. 104-109, doi: 10.1109/ROBIO.2011.6181270.

4- I. S. Godage, D. T. Branson, E. Guglielmino, and D. G. Caldwell,"Path planning for multisection continuum arms," in 2012 IEEEInternational Conference on Mechatronics and Automation, 2012, pp.1208–1213.

5 – B.Meng, D. Arachchige, J. Deng, I. S. Godage, I. Kanj, "Anticipatory Path Planning for Continuum Arms" to appear in 2021 IEEE International Conference on Robotics and Biomimetics, 2021

6 - J. Deng, B. H. Meng, I. Kanj and I. S. Godage, "Near-optimal Smooth Path Planning for Multisection Continuum Arms," 2019 2nd IEEE International Conference on Soft Robotics (RoboSoft), 2019, pp. 416-421, doi: 10.1109/ROBOSOFT.2019.8722778.

7 - B.Meng, I. S. Godage, I. Kanj, "Improved Smoothness for Continuum Arms" to appear in 2021 IEEE International Conference on Robotics and Biomimetics, 2021

8 - Fang, Guoxin & Tian, Yingjun & Yang, Zhi-Xin & Geraedts, Jo & Wang, Charlie. (2020). Jacobian-based learning for inverse kinematics of soft robots.

9 - S. Satheeshbabu, N. K. Uppalapati, G. Chowdhary and G. Krishnan, "Open Loop Position Control of Soft Continuum Arm Using Deep Reinforcement Learning," 2019 International Conference on Robotics and Automation (ICRA), 2019, pp. 5133-5139, doi: 10.1109/ICRA.2019.8793653.

Budget and Budget Narrative

Much of the supplies needed are already included in the ROME lab's funding. We already have an existing prototype arms along with any needed sensors.

- $2,500 to offer as a stipend to 2 students in the ROME lab's research group.
- $500 in HatchBox resin
- $500 Polyhemus G4 Wireless Motion Tracker and Digitizing Stylus
- $600 Pneumatic Regulators
- $400 Compliant Tubes
- $300 Miscellaneous Parts
- $200 Air Compressor

First, we must pay additional students to assist with construction. They will be paid to dedicate 20 hours a week to the implementation and testing. All other materials are either in the lab (the arm, sensors, computers) or could be recreated using DePaul's IRL lab (replacement parts, additional sections, etc).

The Resin, Tubes, and Miscellaneous parts will be used to construct additional prototype arms. Simulation, especially at this stage, is likely to cause wear and potential breaking, so it will be useful to have spares. Additionally, it will likely be prohibitively slow to run experiments serially, so additional arms will expedite this process. The resin is used for 3D printing of rigid components. The remaining parts cannot be produced with maker tools, so they must be purchased.

The regulators are used to manage air moving into and out of the arm. This, paired with the air compressor, is mandatory for implementing an additional arm. Furthermore, they could act as spares if one of our arms were to break.

The Polyhemus sensors are specialty and necessary to perform the manual tracking mentioned in the Experiment section. These sensors are highly sensitive and can monitor the accuracy of our arms down to millimeters.