PolypGripper
A Novel Robotic System for Colonoscopies
Project Report

MIE1076 Artificial Intelligence for Robotics II
Instructor: Prof. Andrew Goldenberg

Prepared by:

Chen, Ryan	Teichman, Matthew	Ali, Noman
1003912992	1002640476	1003602657

University of Toronto

April 27, 2023

Contents

1 Problem Statement

Colorectal cancer is a disease in which abnormal growths, or polyps, form in the colon or rectum and progress to cancer over time [1]. Currently, colorectal cancer is the second and third most common cause of cancer-related deaths among men and women, respectively [2]. However, the implementation of colorectal cancer screening, specifically colonoscopy, has contributed to a decline in the incidence and mortality rates of the disease in recent years [3]. Colonoscopy is a procedure performed either by a general surgeon or a gastroenterologist to visualize the interior of the colon and rectum using a colonoscope, which is a long and flexible tube equipped with a camera at the end [4]. The colonoscope is inserted through the anus into the rectum and colon, and specialized instruments can be passed through to perform biopsies or remove potentially malignant polyps. Despite the efficacy of colonoscopy in preventing colorectal cancer, access to the procedure remains inequitable, particularly among minority and Medicaid populations. In a study conducted by Sanchez *et al.* [5], it is revealed that Black and Hispanic populations have a lower likelihood of receiving colonoscopy compared to non-Hispanic White populations. Furthermore, patients with Medicaid coverage had 35% lower odds of undergoing surveillance colonoscopy. Therefore, while colonoscopy represents a valuable tool in the fight against colorectal cancer, ensuring equitable access to the procedure remains a critical challenge.

1.1 Design Overview

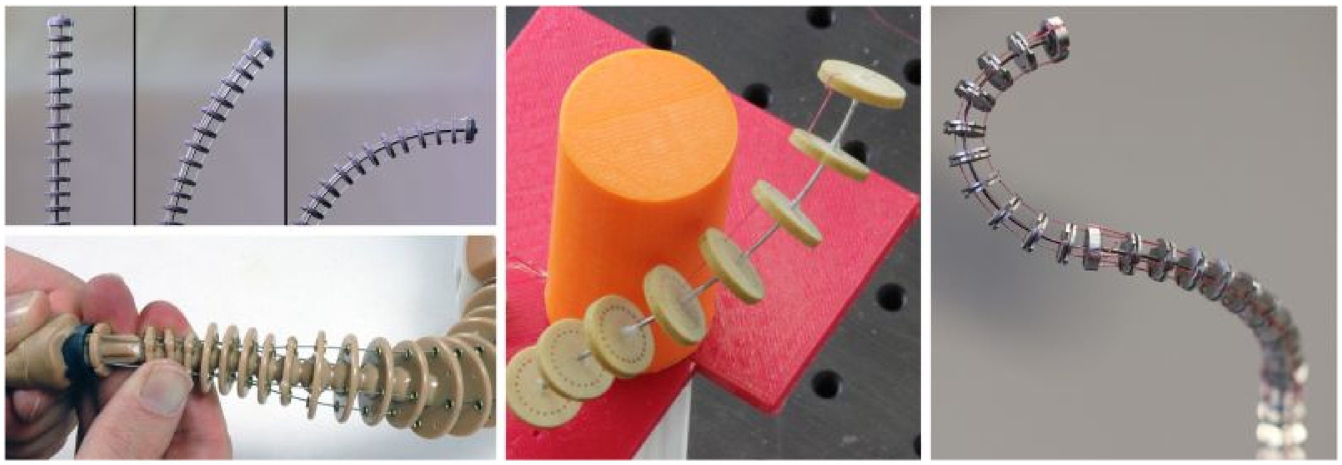


Figure 1: Examples of tendon-driven continuum robots.

To address the above issue, we propose a cooperative surgical system, PolypGripper, that assists physicians with detecting and manipulating cancerous polyps within the colon. PolypGripper is designed to be a tendon-driven continuum robot that leverages the benefits of compliance, flexibility, and high dexterity. Continuum robots are characterized by their slender and soft structure, which allows them to easily navigate through confined environments [6]. Tendon-driven continuum robots are a specific type of continuum robot that utilize cables or tendons to transmit motion and force to the end effector [7]. The inherent flexibility, compliance, and ability to follow non-linear trajectories of tendon-driven continuum robots make them a suitable choice for minimally invasive surgery. Therefore, we chose to utilize the tendon-driven continuum robot for our application of robotic colonoscopy.

To further facilitate its operation, PolypGripper is equipped with several additional hardware features. A 1080p wide-angle camera is attached to the end effector of the robot to enable perception-based tasks. Furthermore, a remote-controlled snare hook is attached to the end effector of the robot for manipulation-related surgical operations.

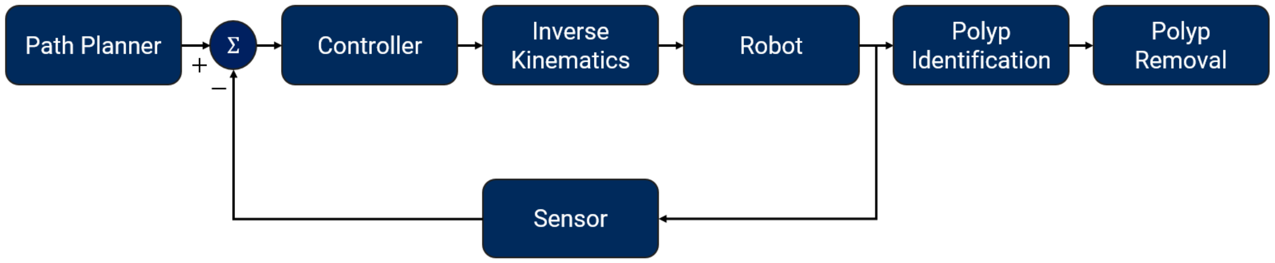


Figure 2: System architecture and workflow of PolypGripper.

The overall system architecture and workflow of PolypGripper is shown in Figure 2. The system operates by first obtaining a magnetic resonance imaging (MRI) scan of the patient's colon to locate the site of potentially cancerous polyps. PolypGripper then generates the shortest path between the entry point and the desired target location within the colon using its path planner. Subsequently, the current desired location is passed into a controller and the inverse kinematics of the robot to determine the desired input tendon forces at each step along the path. The robot's motion is then guided by these forces, and the resulting end effector position is tracked using external sensors. Positional feedback is incorporated into the control loop to adjust the input forces as necessary for accurate movement along the path. Once the robot has successfully navigated to the target site, a deep learning-based algorithm is employed for visual polyp detection. Finally, if a potentially cancerous polyp is identified, the surgeon can remotely employ the onboard snare hook attachment to grasp and remove it.

PolypGripper aims to reduce the surgical procedural complexity and overall physical effort for surgeons. With a reduced learning curve of the surgical procedure, PolypGripper can allow for more life-saving procedures at a reduced cost, compared to traditional endoscopic surgical techniques. This medical robotic system intends to address the issue of inequitable access to colonoscopy and enhance the quality of care for patients at risk of developing colorectal cancer.

1.2 Design Focus

PolypGripper is composed of a variety of different components, as depicted in Figure 2. However, the primary focus of our design lies in the planning, navigation, and perception modules. Specifically, we aim to accomplish the following tasks: path planning within the colon, deep learning-based kinematic modeling of the tendon-driven continuum robot, controlling the tendon-driven continuum robot, and visual perception and identification of cancerous polyps using deep learning. In this report, we will only discuss the main focus of the project, which encompasses the aforementioned tasks of **path planning**, **kinematic modeling**, **robot control**, and **polyp identification**. By focusing on these core components, we aim to develop a reliable and effective system that can assist physicians in detecting and manipulating cancerous polyps in the colon.

1.2.1 Path Planning

Path planning is a critical component of PolypGripper's navigation capabilities within the colon. Given that the target site is predetermined, PolypGripper employs an offline approach to path planning in order to reduce its computational load. This involves generating a set of feasible paths prior to performing the task, with the aim of optimizing for criteria such as path length and obstacle avoidance. The chosen path is then executed during the task by following a sequence of pre-defined waypoints. The efficiency and accuracy of the path planning algorithm are crucial to ensuring the successful completion of the task while minimizing the risk of complications.

1.2.2 Kinematic Modeling

The kinematic modeling task encompasses two subtasks that are critical for the successful operation of PolypGripper:

- **Forward Kinematics:** In order to simulate the behaviour of PolypGripper and validate control strategies, an accurate forward kinematics model is required. Several methods have been proposed in the literature for modeling the forward kinematics of tendon-driven continuum robots. The most suitable model is selected for this application based on accuracy and computation cost.

- **Inverse Kinematics:** The inverse kinematics model of PolypGripper is needed to determine the corresponding tendon actuation input based on the desired end effector position, which is a key requirement for implementing control strategies. As there are no existing analytical solutions for the inverse kinematics of tendon-driven continuum robots, a deep learning-based approach is employed to address this task.

1.2.3 Robot Control

Robot control is an important task in the operation of PolypGripper. In order to achieve precise and accurate control of the robot's motion, a controller is required to regulate the errors induced by potential inaccuracies in the inverse kinematics model and disturbances in real-world scenarios. Therefore, the controller must be designed to be robust to disturbances and uncertainties in the system, such as changes in the colon environment and variability in the robot's physical properties. In addition, the control parameters must be thoroughly tuned to achieve optimal performance in different scenarios.

1.2.4 Polyp Identification

A key element of PolypGripper is the ability to identify polyps in the large bowel. If these polyps are missed and left untreated, the polyps could metastasize into cancer, resulting in a life-ending illness. To address this, PolypGripper incorporates a tissue identification program that analyzes real-time images captured by the camera mounted on the robot's end-effector. The program uses advanced deep learning-based image processing techniques to identify differences between healthy and unhealthy colonic tissues, producing a binary mask of the location of the polyp in the image. The resulting mask enables PolypGripper to determine a path and navigate toward the polyp for removal or further analysis.

2 Background

In this section, a comprehensive literature review regarding the four main tasks previously presented in Section 1.2 is provided.

2.1 Path Planning

The task of path planning and traversing through the colon is heavily linked to robotic path planning in general. Existing robotic path planning approaches can be broadly summarized into two types: online path planning and offline path planning.

Online path planning methods provide real-time information about the robot's location relative to the environment and its assigned goal. Examples include utilizing simultaneous localization and mapping (SLAM) [8], a technique used to map unknown environments, or magnetic manipulation [9]. However, these are computationally expensive, prone to sensor noise, unintuitive, and may not be feasible for the continuum robot [9].

In contrast, generating maps beforehand and utilizing offline path planning algorithms are more computationally efficient, provide higher optimality, and enable path errors to be accounted for before the operation. Examples of such planning algorithms include A* [10], rapidly exploring random tree (RRT) [11], and RRT* [12]. These methods rely on pre-determined maps of the environment and thus generally do not react to dynamic environments, nor do they take sensor noise into account.

A* is a heuristic-based planner used to find optimal paths between two points. Depending on the quality of the heuristic, it can have lower computation cost in a 2D environment [10, 13, 14]. However, being a search-based algorithm, it is prone to the problem of dimensionality. When we extend the problem to a 3D space (or even higher dimensions if we take the orientation of the end effector into account), A* may perform worse. On the other hand, RRT and RRT* are random sampling-based algorithms that are simpler to use with higher dimension configuration spaces, at the cost of sub-optimal paths [11, 12]. The process of randomly sampling across the configuration space also leads to the easier implementation of kinematic or dynamic constraints in the environment (e.g., robot dimensions, turn radius, orientation). However, it also leads to variable path lengths due to its randomized nature.

2.2 Kinematic Modeling

2.2.1 Forward Kinematics

In traditional robotic manipulators, forward kinematics refers to the process of computing end-effector coordinates (task space) based on the joint angles (joint space). However, this process is much more complex in the case of tendon-driven continuum robots. These robots exhibit significant flexibility and deformation due to the tension and compression forces acting on their tendons, making them more challenging to model analytically. Rao *et al.* [7] provides an overview of the state-of-the-art kinematic (based on robot geometry) and static (based on material properties and tendon interactions) models, which include:

- **Piecewise Constant Curvature Model:** A static modeling approach assuming constant curvature bending for each subsegment [15].
- **Pseudo Rigid Body Model:** An approach which assumes and models the same subsegment as pseudo rigid bodies with virtual discrete joints [7].
- **Variable Curvature Model:** An approach which models each subsegment as slender, flexible structures (wires, cables, etc.) and accounts for torsion and shearing [16].

As depicted in Figure 3, these models take the tendon forces as inputs and generate the 3D end effector coordinates as outputs. These models allow for a more accurate representation of the tendon-driven continuum robot's behaviour and can be used to improve control and manipulation capabilities.

2.2.2 Inverse Kinematics

In robotics, the inverse kinematics problem refers to constructing the mathematical mapping from task space to joint space. This mapping is a key component in building functional open-loop and close-loop control schemes to operate the robot with high accuracy. For tendon-driven continuum robots, constructing an analytical solution to the inverse kinematics problem has proven to be a complex task due to their continuous structure and kinematic redundancy [17]. Although several physics-based analytical models have been proposed to calculate the forward kinematics of such robots, existing approaches have not been successful in formulating an analytical solution to the inverse kinematics problem [7, 15, 16, 17, 18].

In recent years, machine learning-based approaches, specifically those that utilize artificial neural networks, have shown great potential in addressing the inverse kinematics problem for both traditional rigid-link robots and continuum robots [19, 20, 21, 22, 23]. These learning-based models have the advantage of accounting for unmodeled factors such as friction and gravity, which often lead to inherent modeling errors in physics-based methods. In particular, deep learning methods have demonstrated notable improvements in both accuracy and computation time for the inverse kinematics of tendon-driven continuum robots [24, 25].

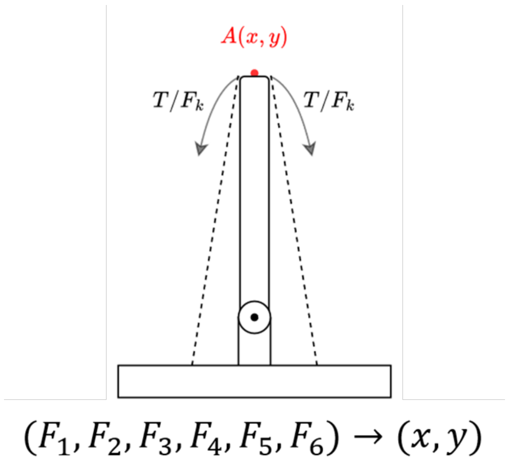


Figure 3: Forward kinematics of tendon-driven continuum robots. The forward kinematics model maps the tendon actuation forces (input) to end effector positions (output).

2.3 Robot Control

Robot control is a crucial aspect of robotics that deals with regulating the behaviour and motion of robots. A successful control system should be able to handle various sources of uncertainty, including modeling errors, disturbances, and sensor noise, to achieve the desired behaviour of the robot. One of the most common approaches to robot control is the use of classical control techniques, such as proportional-integral-derivative (PID) controllers [26]. These controllers have been widely used in industrial robotics due to their simplicity and ease of implementation. However, they have limited capabilities to handle nonlinearities and uncertainties, which are inherent in many robotic systems. To address the limitations of classical control techniques, researchers have developed various advanced control strategies, including adaptive control, robust control, and model predictive control (MPC) [27, 28, 29].

Furthermore, recent advances in machine learning have also led to the development of learning-based control strategies. These approaches use data-driven models and optimization algorithms to learn the optimal control policies directly from the data. Reinforcement learning (RL), in particular, has emerged as a popular approach in learning-based control, where an agent learns through trial and error to maximize a cumulative reward signal [30]. Deep RL, which combines deep neural networks with RL, has also shown impressive results in several robotics domains, including manipulation, locomotion, and navigation [31].

2.4 Polyp Identification

Accurately generating masks of polyps is a crucial requirement for the polyp identification function of PolypGripper. There are two prominent neural network architectures that generate masks for semantic segmentation tasks: fully convolutional networks (FCNs) [32] and U-Net [33], also known as encoder-decoder networks.

FCNs are neural networks that are comprised solely of convolutional layers [32]. These networks take input images and pass them through multiple convolutional hidden layers, with the output being a set of dense prediction masks. However, the use of FCNs is limited by several issues, including the potential for vanishing gradients between hidden layers and the downsizing of prediction masks compared to the original image size.

In contrast, U-Net is an encoder-decoder network, in which the first half of the network encodes image features by applying convolutional down-sampling, while the second half decodes the features by up-sampling the hidden features to generate a set of prediction masks [33]. U-Net incorporates skip connections between each sequential layer, which copy features and merge them with the decoder to prevent vanishing gradients commonly observed in FCN-style networks. Originally developed for medical image segmentation, U-Net has been used in numerous academic papers for applications such as skin disease detection, organ segmentation, and polyp segmentation.

3 Methodology

In this section, the methods utilized to tackle the tasks outlined in Section 1.2 are introduced.

3.1 Path Planning

Offline path planning methods are chosen for PolypGripper due to their computational efficiency and ability to generate and account for paths before the operation, ensuring patient safety. Since the colon map is obtained beforehand, the environment is assumed to contain no dynamic obstacles or interferences during the operation. Although the A* algorithm could potentially serve as a feasible path planner, incorporating kinodynamic constraints into search-based planners poses challenges. The generated path may not be kinematically feasible due to sharp turns and edges, as demonstrated in Figure 4. Considering the high dimensionality of our environment and the fact that optimality is not a major requirement, RRT* is selected as the path planning algorithm for PolypGripper.

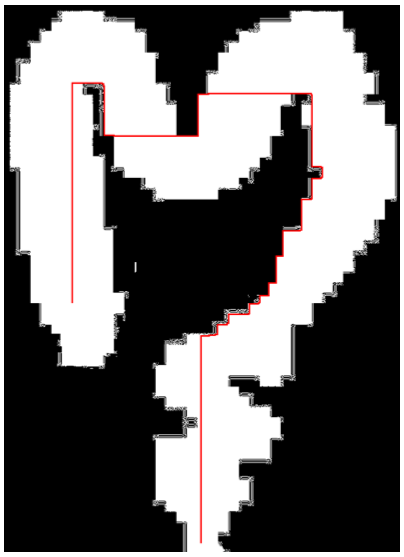


Figure 4: A path generated by the A* path planner in a 2D map of the colon.

Before implementing the algorithm, several constraints and considerations are added to the path planning algorithm. These can be broken down into the following steps: kinodynamic constraints, collision checking, and finally, the algorithm's approach.

3.1.1 Kinodynamic Constraints

As discussed in Section 3.1, directly generating paths which comprise of nodes and coordinates of the shortest path with sharp turns would not be ideal for tendon-driven continuum robots. They have a radius of curvature which must be considered in order to ensure safe and effective navigation.

One approach to account for this constraint is by using curved trajectories such as splines or Bezier curves. We will impose a similar constraint on the path by using Dubins paths. A Dubins path is described as the shortest curve that connects two points in a 2D plane, while also considering their initial and final orientation [34]. Although they are typically used for planning paths for wheeled robots or cars, Dubins paths can be used for continuum robots since they share similar constraints. Figure 5 shows an example of a Dubins path generated for an object with initial and final configurations.

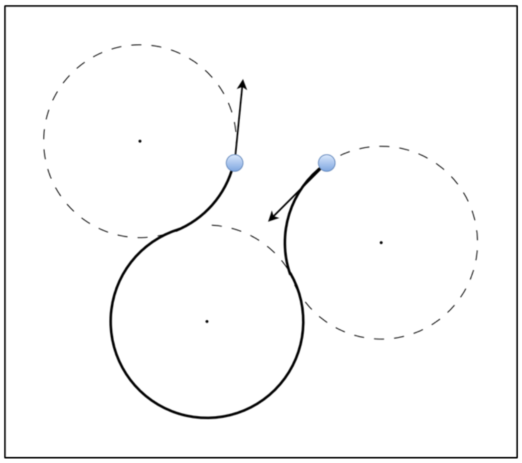


Figure 5: Dubins path from an initial position and orientation to the goal position and orientation.

3.1.2 Collision Checking

Collision and obstacle avoidance is another important aspect of path planning and is generally an extremely computationally expensive procedure. To lower the computation cost, we discretize the obstacles in the map into equivalent circular obstacles. This reduces the complexity of the map and leads to easier collision checking. Figure 6 shows a discretization of an environment's obstacles into circular equivalents.

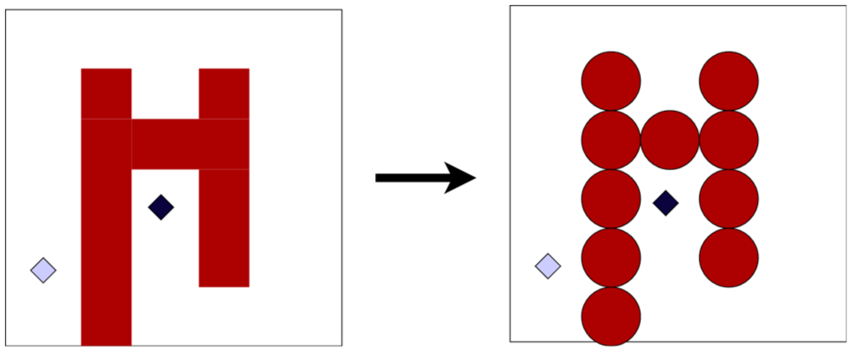


Figure 6: Discretization of obstacles in an environment. The original map on the left is transformed into the map on the right through the discretization process.

The effectiveness of the newly generated map can be increased by using more circles (and increasing computation cost) or increasing their size (and decreasing the accuracy of the map). To check for collisions between the straight paths, we use geometry to measure the shortest distance between the line and the center of the circular obstacle. If the distance is smaller than the radius, the path is considered unviable. This process is as follows:

$$ax + by + c = 0 \;\rightarrow\; \text{Straight Line Segment of Path}$$

$$C_o(y_0, x_0) \;\rightarrow\; \text{Center of Circular Obstacle with Radius } r_0$$

$$d = \frac{|ax_0 + by_0 + c|}{\sqrt{a^2 + b^2}}$$

$$d > r_o \text{ For Path to Be Viable} \tag{1}$$

This is not as straightforward for calculating collision for the circular path, since there are numerous edge cases where measuring the distance between the two circle radii may not account for a large radius of curvature. We can check for individual points in the circular path and their distance to the circular obstacles, or a safer (yet sub-optimal) route would be to ensure that the distance between the center of the curvature circle and the circular obstacle is greater than the sum of their radii.

$$C_o(y_0, x_0) \rightarrow \text{Center of Circular Obstacle with Radius } r_0$$

$$C_r(y_r, x_r) \rightarrow \text{Center of Radius of Curvature with Radius } r_r$$

$$d = \sqrt{(x_0 - x_r)^2 + (y_0 - y_r)^2}$$

$$d > r_o + r_c \text{ For Path to Be Viable} \tag{2}$$

In simulation, the individual points of the circular path are checked. However, the sub-optimal route is also viable. Figure 7 shows examples of the above methods.

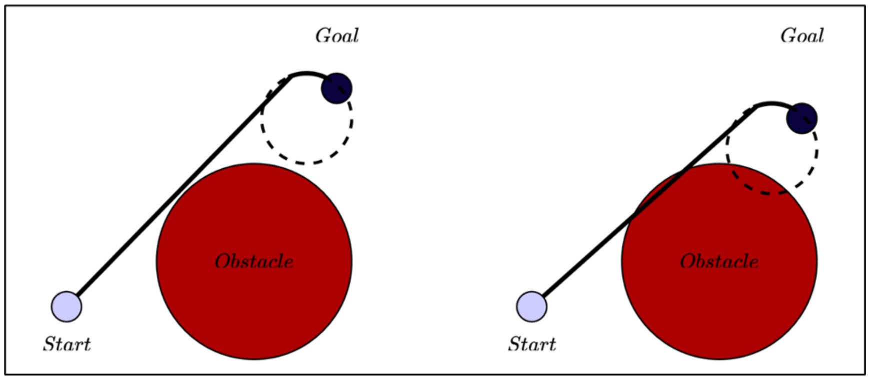


Figure 7: Satisfying straight and curved path collision checks (left) and failing both collision checks (right).

3.1.3 RRT* Search

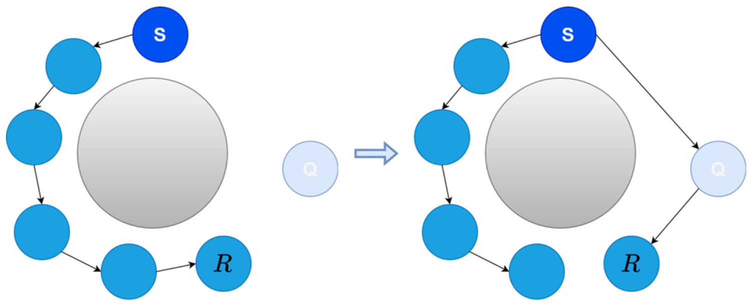


Figure 8: Connecting nodes via the RRT* algorithm.

RRT* is an extension of the RRT algorithm, which is also sampling-based. RRT generates random data points (nodes) and links them based on proximity, forming a tree structure. On the other hand, RRT* considers the cost (or total distance travelled) of each path and tries to find a path with the lowest cost. It achieves this by rewiring the tree structure by considering the new path and changing the parent of nodes if it reduces the overall cost of the path. This rewiring process can result in a more optimal path and can converge to the optimal solution given enough time. Figure 8 shows an example of how RRT* works.

Despite being closer to node R, the new node Q is connected to the starting node S because of its lower cost. Furthermore, RRT* restructures the tree by rewiring the parent of R to Q, resulting in a decreased overall cost for node R. Figure 9 shows two different trees generated by RRT (left) and RRT* (right), showing the different paths generated by each algorithm [35].

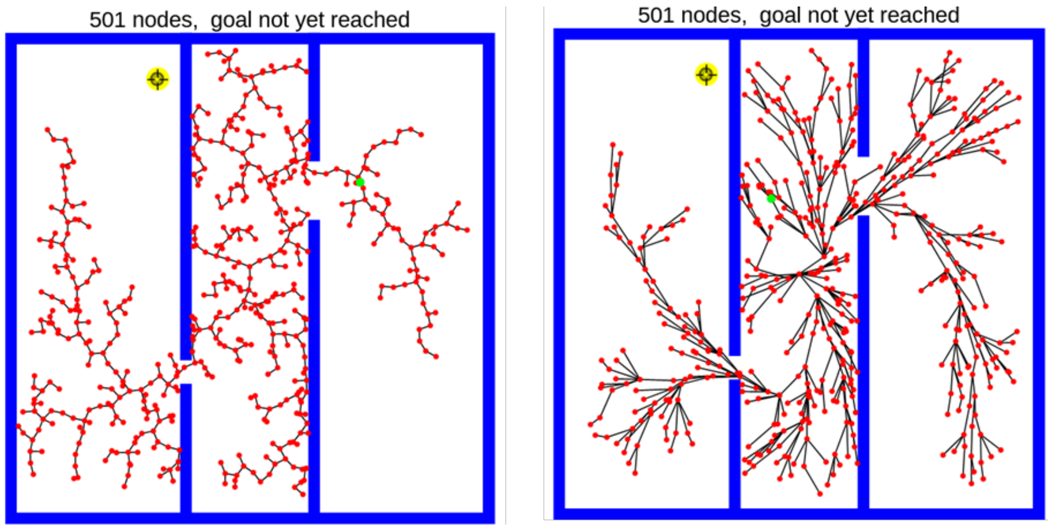

501 nodes, goal not yet reached
501 nodes, goal not yet reached

Figure 9: Generated trees using RRT (left) and RRT* (right).

3.2 Kinematic Modeling

3.2.1 Forward Kinematics

Using the models mentioned in Section 2, we compared the different models with randomized tendon forces based primarily on their computation time, as well as their accuracy [7]. Out of all the models, the pseudo rigid body model (PRBM) demanded the most computational resources and delivered the highest level of precision. However, using the PRBM outputs as the baseline, we discovered that the piecewise constant curvature model (CCsub) and variable curvature model (VCsub) were significantly faster, with CCsub being the most efficient. Although their outcomes were less precise than the PRBM, their accuracy levels proved acceptable, with a maximum difference of 1% in our experiments. With this in mind, we opted for the CCsub model as the forward kinematics model of PolypGripper.

Table 1: Forward Kinematics Dataset Generation

Data Point	T_1	T_2	T_3	T_4	T_5	T_6
1	0	0	0	0	0	0
2	0	0	0	0	0	0.5
3	0	0	0	0	0	1
...	...	...	...	...	...	...
15,623	2	2	2	2	2	1
15,624	2	2	2	2	2	1.5
15,625	2	2	2	2	2	2

We decided to utilize the CCsub model selected earlier to create a dataset with sufficient size to develop an accurate learning-based inverse kinematic model. Our approach involved defining the workspace limits from 0 newtons, indicating no force, to 2 newtons, which offered a reasonable deflection. We then chose to increment the six tendon forces $\boldsymbol{T} = [T_1\ T_2\ T_3\ T_4\ T_5\ T_6]^\top$ uniformly from 0 to 2, in fixed increments of 0.5. This method generated a total of 15,625 data points. This data generation procedure is outlined in

Table 1. We preferred fixed increments instead of random increments over the workspace to avoid any bias towards specific regions in the generated dataset. The resulting end effector position based on the input tendon forces are recorded to create the kinematics dataset.

3.2.2 Inverse Kinematics

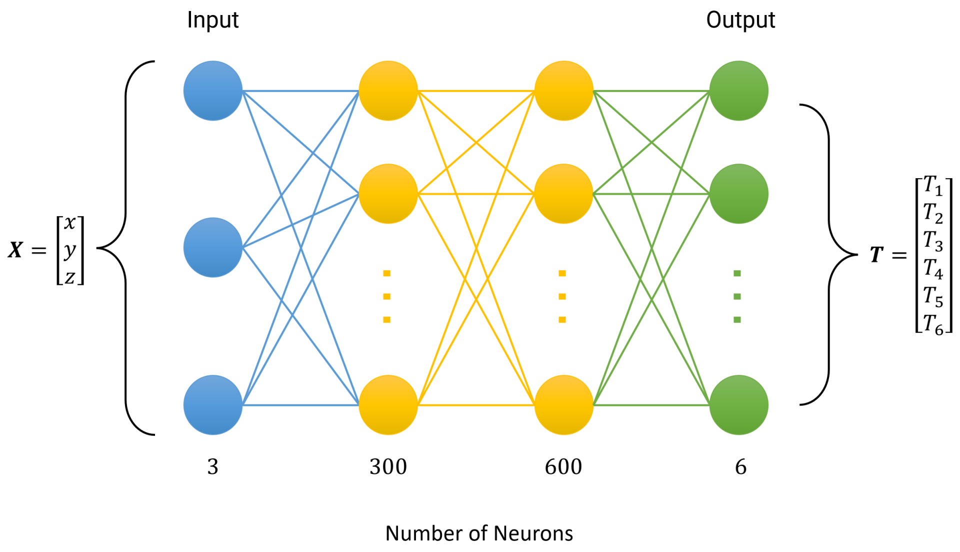

Input
Output
$X = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$
$T = \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \\ T_6 \end{bmatrix}$
3
300
600
6
Number of Neurons

Figure 10: Inverse kinematics neural network architecture. The input layer consists of 3 neurons corresponding to the Cartesian position of PolypGripper's end effector. The output layer consists of 6 neurons corresponding to PolypGripper's input tendon forces. There are two intermediate hidden layers between the input and output layers with 300 and 600 neurons, respectively.

To model the inverse kinematics of PolypGripper, a deep learning-based approach is utilized where the direct inverse mapping between the end effector position and tendon actuation input is learned using a neural network. The previously generated kinematics dataset based on the forward kinematics model is used to train the neural network. The data is split with a ratio of $8 : 1 : 1$, with 12,000 data points for training, 1,500 data points for validation, and the remaining 1,500 data points for testing.

The neural network architecture consists of an input layer with 3 neurons corresponding to the Cartesian position of the robot end effector and an output layer with 6 neurons representing the input tendon forces. The network has two intermediate hidden layers with 300 and 600 neurons, respectively. The overall architecture of the neural network model is illustrated in Figure 10.

To introduce non-linearity into the model, the rectified linear unit (ReLU) activation function is employed between the hidden layers and the output layer. The Kaiming Initialization technique is used to initialize the neural network weights [36], and the mean square error (MSE) loss function is employed to measure the model's performance during each training epoch. The Adam optimizer is utilized to optimize the trainable parameters, with a mini-batch size of $N_{bs} = 128$ [37]. The network is trained for 1000 epochs with a learning rate of `1e-3`.

3.3 Robot Control

With an inverse kinematics model in place, the next step is to design a control scheme to operate PolypGripper. The simulation control loop is depicted in Figure 11. The PID controller is implemented in a closed-loop control system, with the learned inverse kinematics model producing the input tendon forces to the robot. The controller continuously monitors the error between the desired and actual end effector positions and makes updates accordingly. In simulation, the forward kinematics model serves as a proxy for the robot model in the real-world scenario. The output of the forward kinematics model is considered the

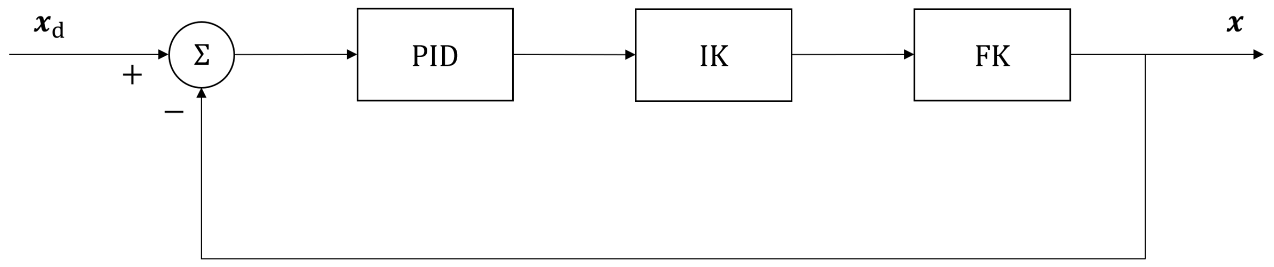


Figure 11: Control scheme of PolypGripper in simulation. The PID controller is chosen for its robustness in various robotic applications. The learned inverse kinematics model provides input tendon forces to the robot. The forward kinematics model serves as a surrogate for the robot model. The output of the forward kinematics model is considered to be the ground truth end effector position.

ground truth end effector position and used as feedback in the control loop, rather than relying on sensor feedback in the physical robot case. The PID controller is chosen in this application based on its robustness in a wide range of robotic applications. The PID controller accounts for the integral and derivative of the error over time, which helps to eliminate steady-state error and improve the response speed. The gains of the PID controller are manually tuned through trial and error.

3.4 Polyp Identification

For polyp identification, our group decided to use the UPolySeg model proposed by Mohapatra *et al.* [38]. We choose the UPolySeg model due to its success at learning to identify polyps with a dataset of fewer than 1000 images. As a result, we replicated the work utilizing the neural network architecture described in the paper. Figure 12 shows an overview of the model's architecture.

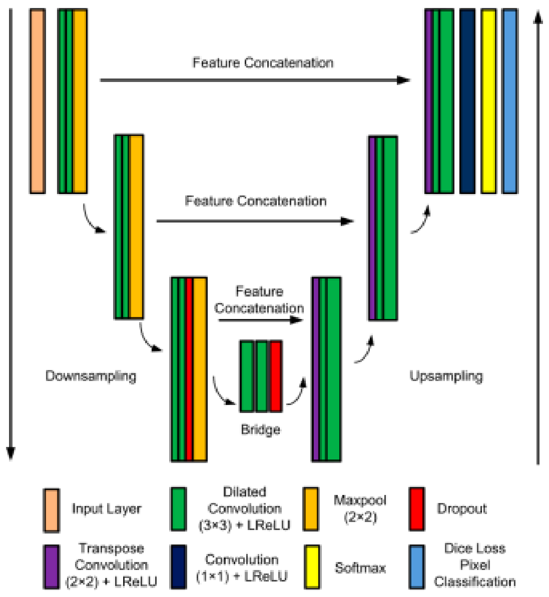


Figure 12: Neural network architecture of UPolySeg. UPolySeg is a U-Net-based polyp segmentation network proposed by Mohapatra *et al.* [38].

The UPolySeg model consists of two main components: the encoder side and the decoder side. The encoder side down-samples the input image into smaller feature sizes, while the decoder side up-samples the features into a larger size. Between each layer of the network, features are concatenated by copying and merging them with the decoder side. The feature concatenation enables the U-Net-style model to learn from small datasets and avoids the problem of vanishing gradients. The resulting output of UPolySeg is a binary mask the same size as the input image.

Each layer of the encoder side consists of double convolution blocks, which are two dilated convolutions followed by a leaky-ReLU activation function in series. After the double convolutional block, the image is down-sampled by a factor of two using a max pool operator. At each resulting layer of the encoder, features are stored to be used on the decoder side. At the bottom of the encoder side is a bridge block, which is once again a double convolutional block followed by a dropout layer. On the decoder side, features are up-sampled using transpose convolution. The resulting feature from transpose convolution and copy features from the decoder are merged using a double convolution block. The process for the decoder is repeated until the output feature matches the same size as the input image. Lastly, the network applies 1×1 convolution to reduce the feature channel size to a binary mask, where a Sigmoid activation function is applied to output the mask.

The loss function used was the dice coefficient loss function. The main advantage of using the dice coefficient loss over binary cross-entropy pixel loss was the dice loss's ability to award mask spatial accuracy. As a result, training convergence became more efficient, and the gradients were smoother. The loss function is as follows:

$$\boldsymbol{L}_{dice} = \frac{2 * \Sigma p_{true} * p_{pred}}{\Sigma p_{true}^2 + \Sigma p_{pred}^2 + \epsilon}, \tag{3}$$

where the two terms were squared to provide better training stability against noise, and an epsilon constant was added with a small decimal number to avoid issues with zero division.

In our application, the UNet polyp identification model is trained for 100 epochs. Training is stopped once the training loss is no longer monotonically decreasing.

4 Results

In this section, the performance of each task outlined in Section 1.2 is individually evaluated in simulation and discussed.

4.1 Path Planning

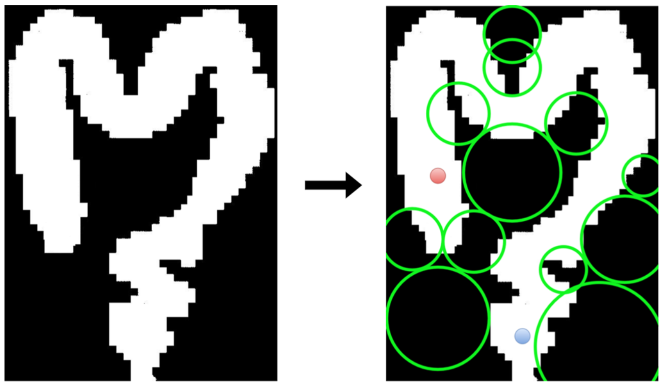


Figure 13: Conversion of the grayscale environment to the circular obstacle environment.

For the path planning simulation, we first performed manual image segmentation on a 2D colon map used in a similar planner to convert the environment into circular obstacles to avoid [39]. The converted map is shown in Figure 13. We then search this environment using RRT*, with a visualizer and Dubins path solver provided by Professor Jonathon Kelly [40] with the starting node (blue) as our entry point, and the goal node (red) situated at the farthest end of the colon.

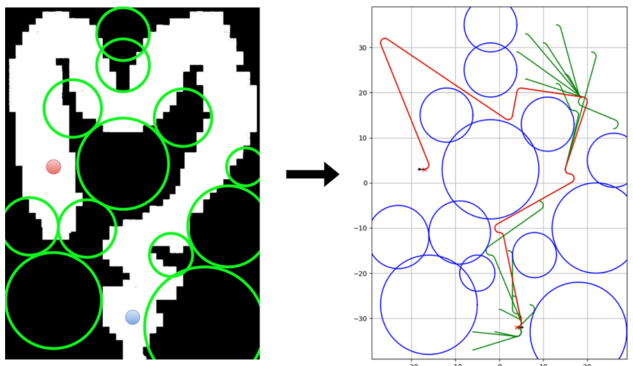


Figure 14: Path generated within the colon using the RRT* algorithm.

The resulting generated path is depicted in Figure 14. In this 2D space, RRT* took an average of 240 iterations to find a path with different seed values, and an average time of 4 seconds. However, the environment is not an accurate representation with some obstacle areas not being covered after conversion. A higher resolution obstacle discretization would help in acquiring a more accurate map, at the cost of a higher processing time.

Overall, these results show the benefits of using RRT* as an offline planning tool for PolypGripper, and extending the environment into a 3D space is considerably straightforward, as the circular obstacles will transform into spherical ones, while the Dubins paths generated will be equivalent to 3D Dubins paths [41].

4.2 Kinematic Modeling

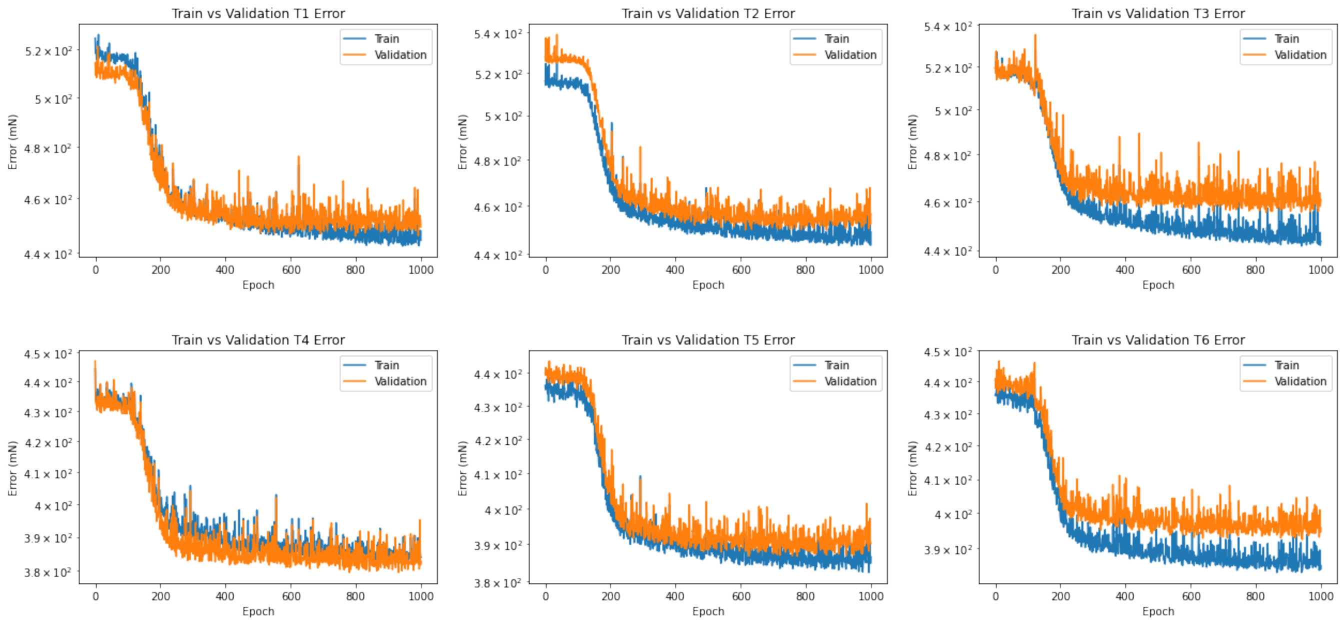


Figure 15: Training and validation error curves of the inverse kinematics neural network model.

The training results of the deep learning-based inverse kinematics model are shown in Figure 15. The model with the lowest validation loss during the training process is selected as the final model for the robot control task. It can be seen that the training errors closely resemble the validation errors with minimal over-fitting occurring in the obtained model.

To evaluate the performance of the selected model, it is tested on the 1,500 data points in the test set. The model's ability to accurately map end effector positions to corresponding tendon actuation forces is

Table 2: Tendon Actuation Force Errors of Learned Inverse Kinematics Model in Newtons.

Errors	T_1	T_2	T_3	T_4	T_5	T_6
Values	0.454	0.451	0.448	0.392	0.385	0.379

measured by calculating the error between the predicted and ground truth tendon actuation forces. The results are presented in Table 2.

From the table, it can be observed that the trained model can achieve an accuracy of less than 0.5 newtons in predicting the required tendon actuation forces for a given end effector position. This level of accuracy is considered sufficient for the intended robot control tasks. Furthermore, the results indicate that the model generalizes well to unseen data, as the testing error is comparable to the validation error during training.

Overall, the deep learning-based inverse kinematics model provides a robust and accurate solution for determining the corresponding joint space values based on task space configurations. These results demonstrate the potential of the proposed model to be implemented in robotic control applications.

4.3 Robot Control

To evaluate the performance of the proposed control scheme, the end effector of PolypGripper is controlled to move to various target positions in simulation, as illustrated in Figure 16. The resulting x, y, and z components of PolypGripper's end effector position during the experiment are plotted in Figure 17. The results show that the robot is able to efficiently reach the desired target locations with minimal overshoot, indicating the effectiveness of the proposed control scheme.

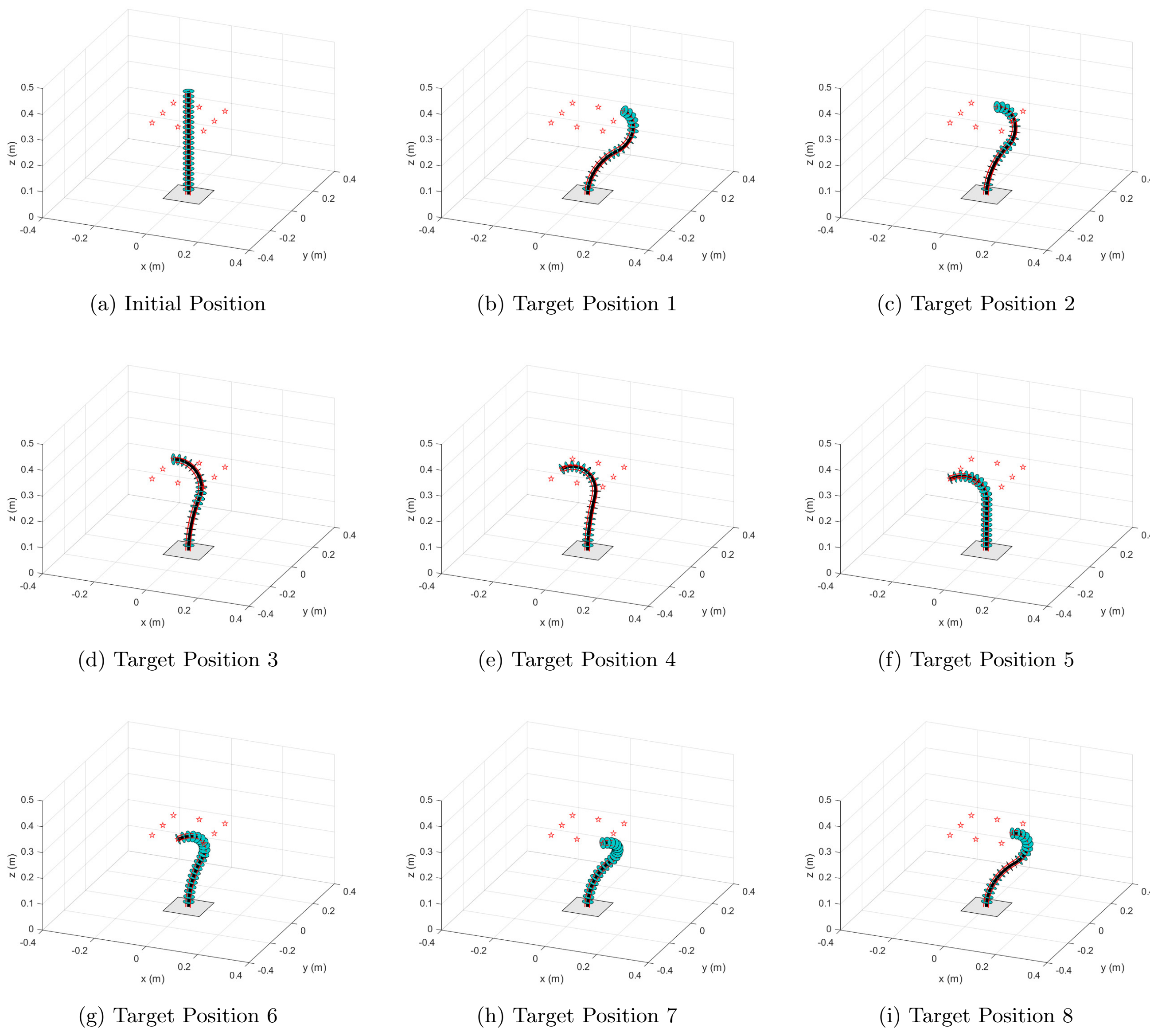
In addition to the simple position control demonstration, the control scheme is further utilized to control the end effector of PolypGripper to follow various complex paths and patterns. The results of the path following experiments are presented in Figure 18. In the first experiment, PolypGripper's end effector is controlled to follow a circular path with a radius of 5 cm in the $x - y$ plane centred at the origin. In the following experiment, the end effector of PolypGripper is controlled to follow a maple leaf pattern in the $x-y$ plane centred at the origin. In both experiments, the desired end effector position is constantly changing between each iteration. The paths are generated through a discretization procedure where PolypGripper is controlled to follow several points along the designed paths in sequence. The results demonstrate that the proposed control scheme enables PolypGripper to accurately follow the desired paths.

Overall, the control scheme based on the PID controller and neural network model for inverse kinematics provides a framework for precise and efficient control of PolypGripper in the simulation environment. These results serve as a proof of concept for the proposed control scheme and show its potential for controlling PolypGripper in real-world scenarios.

4.4 Polyp Identification

The polyp identification model exhibited excellent performance, as demonstrated by its dice coefficient score of 94.7% and intersection over union (IoU) score of 90.6%. These results are comparable to those obtained in the UPolySeg paper, which reported a dice score of 96.86% and an IoU score of 87.91%. However, it is worth noting that our dataset was larger, which may have contributed to the better performance of our model. Additionally, we employed batch normalization between convolutional layers, which likely served as an additional form of regularization, contributing to the high scores.

The pixel mask precision and recall scores were 94.8% and 92.9%, respectively, indicating high accuracy in the identification of unhealthy colonic tissue. Notably, the model demonstrated particular difficulty in correctly identifying the border margins of the sessile polyp, which has the highest potential for cancerous growth. While the model performed well overall, a medical practitioner noted that the model's ability to identify the polyp borders could be improved.



(a) Initial Position (b) Target Position 1 (c) Target Position 2

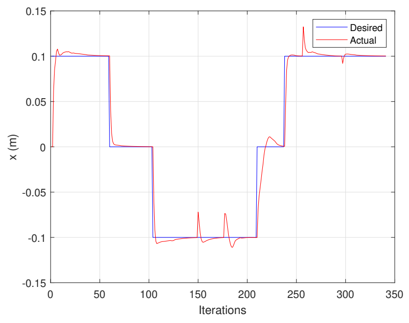
(d) Target Position 3 (e) Target Position 4 (f) Target Position 5

(g) Target Position 6 (h) Target Position 7 (i) Target Position 8

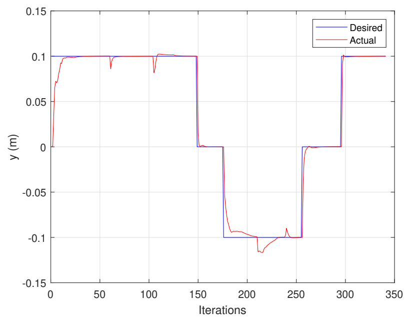
Figure 16: Position control of PolypGripper's end effector in simulation. (a) depicts the initial position of the robot. (b), (c), (d), (e), (f), (g), (h), (i) demonstrate precise control of PolypGripper's end effector to desired target locations indicated by red stars.

To enhance the performance of the polyp identification model, the training dataset could be expanded to include a greater number of sessile polyps, which would address the imbalance between normal and sessile polyps in the current dataset. Additionally, reducing noise in the dataset by increasing the batch size and decreasing the learning rate may improve performance and stability. Finally, the inclusion of a buffer mask around the borders of the mask may help ensure that all polyp tissue is detected and excised during the operation.

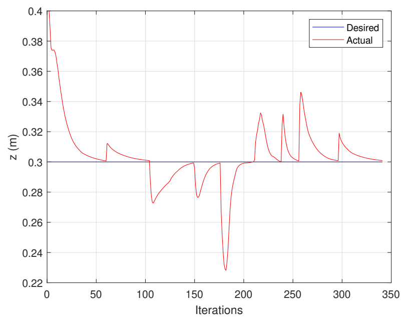
Sample inferences generated by the polyp identification model are presented in Figure 19, demonstrating the model's efficacy in detecting differences between healthy and unhealthy colonic tissue. Overall, the results suggest that the UPolySeg model is a robust model to be utilized in the polyp identification task of PolypGripper.



(a) x Direction

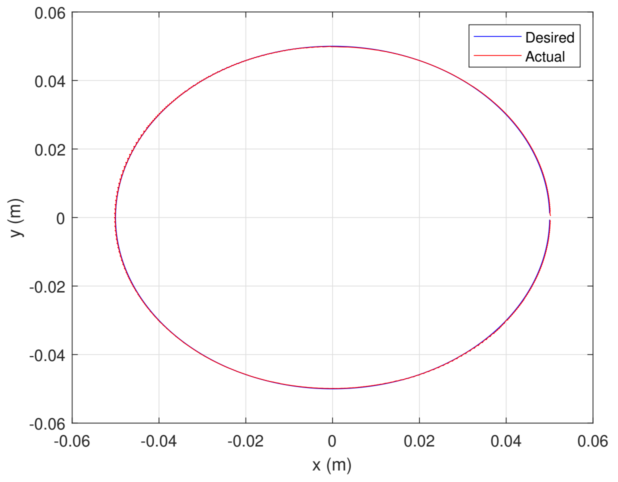


(b) y Direction

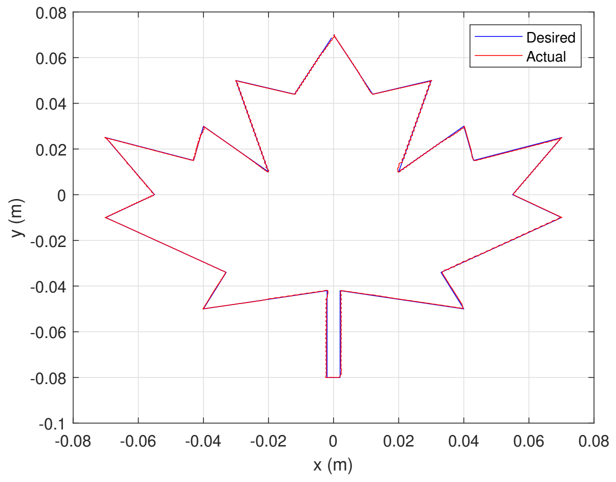


(c) z Direction

Figure 17: Plotted end effector positions in the position control experiment. (a), (b), (c) represent the x, y, and z components of PolypGripper's end effector in Cartesian coordinates, respectively.



(a) Circular Path



(b) Maple Leaf Path

Figure 18: Simulated PolypGripper end effector trajectories following various paths. The robot end effector is controlled to follow a circular path in (a) and a maple leaf pattern in (b).

5 Future Work

After successfully completing simulations, our team began constructing a prototype of the PolypGripper robotic device. We designed and engineered each individual component in Solidworks, creating a system assembly as shown in Figure 20. Our PolypGripper design consists of eight stepper motors that connect to a simple spur gear, driving the spool containing the robot's tendon. The tendons run along the robot's body and control its deflection. The eight stepper motors are divided into two separate levels, with each level controlling a segment of the robot. For example, the first level of four motors controls the first half of the robot, while the second level of four stepper motors controls the end-effector segment. At the end-effector, we have a 1080p camera and 9-axis accelerometer/gyro, with electronics wiring running through a flexible rubber tube along the robot's backbone to the base. During construction, we determined that DC motors would provide better control of tension in the robot's tendons than stepper motors. Consequently, we halted construction to incorporate DC motors into the design. Figure 21 shows some of the 3D components constructed using our 3D printer, with all parts designed using PLA plastic material.

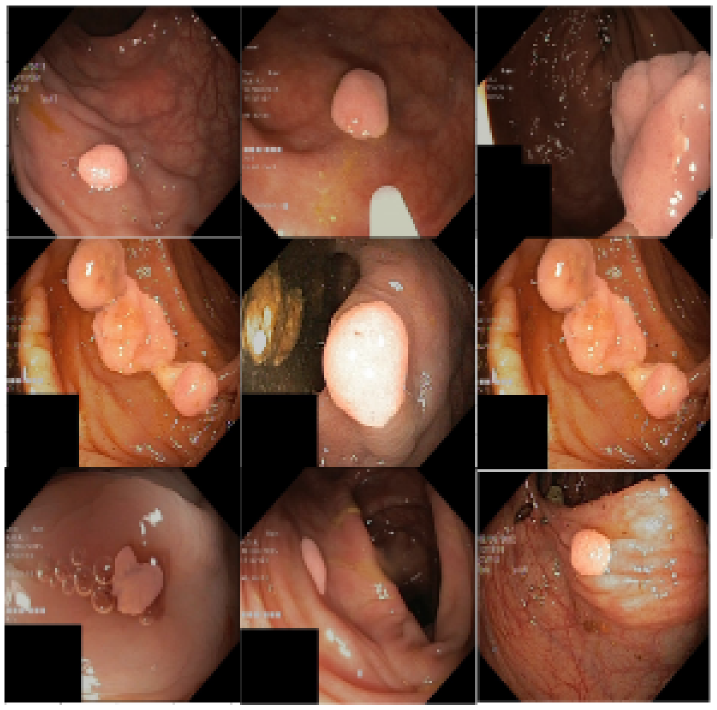


Figure 19: Sample inferences generated by the polyp identification model on the test dataset. The highlighted regions indicate the presence of polyp tissue while non-highlighted regions represent normal healthy colonic tissue.

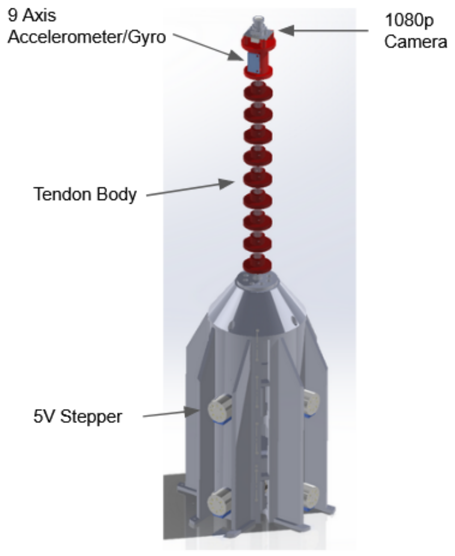


Figure 20: 3D CAD model of PolypGripper produced in Solidworks.

Moving forward, we plan to continue improving our initial PolypGripper design and making systematic improvements to our robotic systems. These improvements include adding depth perception to navigate through the colon, introducing noise to simulation, expanding our path planning to include three-dimensional anatomy, incorporating resistance feedback to control, adding 6D poses to our inverse kinematic model, and researching complex reinforcement learning methods for polyp removal.

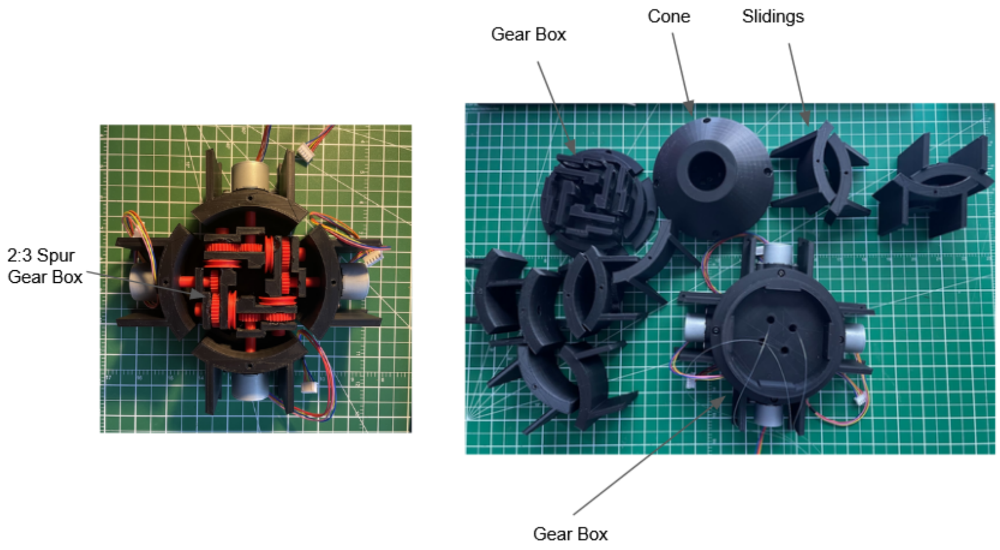


Figure 21: 3D printed components of PolypGripper produced during robot construction.

To achieve depth perception, we plan to use a stereo camera, as other methods such as LiDAR and ultrasonic sensors would not be suitable for a colonic environment. To ensure our control algorithm operates in the colon, we will add random noise to our sensor feedback to simulate uncertainty in the environment.

Our current planning algorithms use 2D maps, but for the robot to navigate through the colon, we need to expand our planning algorithm to 3D. Moreover, we must incorporate an efficient online real-time element into our planner to ensure that it follows the offline path, responds to sensor/actuator noise, and accounts for dynamic changes in the environment.

Tactile feedback from medical instruments is essential for surgeons during polyp removal procedures. To improve our control algorithms, we plan to include resistance/force feedback in our control loop. Additionally, we intend to enhance our inverse kinematic model to account for orientation, not just positional translation, thereby allowing the robot to navigate around tight bends in the colon.

Finally, mimicking a surgeon's ability to extract and manipulate polyps is not possible with traditional robotic methods and may require advanced deep learning techniques. As a result, our team will be actively researching various reinforcement learning methods for future work.

6 Conclusion

In conclusion, this study presents a novel robotic system, PolypGripper, for efficient detection and removal of cancerous polyps in the large colon. Our work successfully demonstrates the use of a neural network to model the inverse kinematics of the tendon-driven continuum robot. Moreover, through motion simulations, the effectiveness of classical PID control strategies in regulating the position of the robot end-effector is demonstrated. Additionally, our work shows that it is possible to generate a path through the colon using MRI images, which can aid in the efficient navigation of the robot. The use of deep learning-based computer vision algorithms also produced an accurate polyp detection network despite the limited medical data and computing power available. Lastly, the PolypGripper prototype model showcases the feasibility of constructing a tendon-driven continuum robot with 3D printed materials.

References

[1] "What is colorectal cancer?" *Centers for Disease Control and Prevention*, 2023. [Online]. Available: https://www.cdc.gov/cancer/colorectal/basic_info/what-is-colorectal-cancer.htm

[2] "Colorectal cancer statistics," *Canadian Cancer Society*, 2022. [Online]. Available: https://cancer.ca/en/cancer-information/cancer-types/colorectal/statistics

[3] "Colorectal cancer statistics | how common is colorectal cancer?" *American Cancer Society*, 2023. [Online]. Available: https://www.cancer.org/cancer/colon-rectal-cancer/about/key-statistics.htmls

[4] "What is colonoscopy? | how is a colonoscopy done?" *American Cancer Society*, 2023. [Online]. Available: https://www.cancer.org/treatment/understanding-your-diagnosis/tests/endoscopy/colonoscopy.html

[5] J. I. Sanchez, V. Shankaran, J. M. Unger, M. M. Madeleine, S. R. Selukar, and B. Thompson, "Inequitable access to surveillance colonoscopy among medicare beneficiaries with surgically resected colorectal cancer," *Cancer*, vol. 127, no. 3, pp. 412–421, Feb. 2021.

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

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

[8] J. M. Santos, D. Portugal, and R. P. Rocha, "An evaluation of 2d slam techniques available in robot operating system," in *2013 IEEE international symposium on safety, security, and rescue robotics (SSRR)*. IEEE, 2013, pp. 1–6.

[9] J. W. Martin, B. Scaglioni, J. C. Norton, V. Subramanian, A. Arezzo, K. L. Obstein, and P. Valdastri, "Enabling the future of colonoscopy with intelligent and autonomous magnetic manipulation," *Nature machine intelligence*, vol. 2, no. 10, pp. 595–606, 2020.

[10] J. K. Goyal and K. Nagla, "A new approach of path planning for mobile robots," in *2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2014, pp. 863–867.

[11] S. M. LaValle, "Rapidly-exploring random trees : a new tool for path planning," *The annual research report*, 1998.

[12] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011. [Online]. Available: https://doi.org/10.1177/0278364911406761

[13] S. M. LaValle, *Planning algorithms.* Cambridge University Press, 2014.

[14] C. Zammit and E.-J. Van Kampen, "Comparison between a* and rrt algorithms for 3d uav path planning," *Unmanned Systems*, vol. 10, no. 02, pp. 129–146, 2022.

[15] H. Yuan, L. Zhou, and W. Xu, "A comprehensive static model of cable-driven multi-section continuum robots considering friction effect," *Mechanism and Machine Theory*, vol. 135, pp. 130–149, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0094114X1831468X

[16] 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.

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

[18] A. Gao, Y. Zou, Z. Wang, and H. Liu, "A General Friction Model of Discrete Interactions for Tendon Actuated Dexterous Manipulators," *Journal of Mechanisms and Robotics*, vol. 9, no. 4, 06 2017, 041019. [Online]. Available: https://doi.org/10.1115/1.4036719

[19] C. Bergeles, F.-Y. Lin, and G.-Z. Yang, "Concentric tube robot kinematics using neural networks," in *Hamlyn Symposium on Medical Robotics*, 2015, pp. 13–14.

[20] R. Grassmann, V. Modes, and J. Burgner-Kahrs, "Learning the forward and inverse kinematics of a 6-dof concentric tube continuum robot in se(3)," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018, pp. 5125–5132.

[21] R. Grassmann and J. Burgner-Kahrs, "On the merits of joint space and orientation representations in learning the forward kinematics in se(3)," in *Robotics: Science and Systems*, 2019.

[22] N. Liang, R. M. Grassmann, S. Lilge, and J. Burgner-Kahrs, "Learning-based inverse kinematics from shape as input for concentric tube continuum robots," in *IEEE International Conference on Robotics and Automation*, 2021.

[23] R. M. Grassmann, R. Z. Chen, N. Liang, and J. Burgner-Kahrs, "A dataset and benchmark for learning the kinematics of concentric tube continuum robots," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 9550–9557.

[24] *A Feed Forward Neural Network for Solving the Inverse Kinetics of Non-Constant Curvature Soft Manipulators Driven by Cables*, ser. Dynamic Systems and Control Conference, 10 2013. [Online]. Available: https://doi.org/10.1115/DSCC2013-3740

[25] S. Garg, S. Dudeja, and V. Rastogi, "Inverse kinematics of tendon driven continuum robots using invertible neural network," in *2022 2nd International Conference on Computers and Automation (CompAuto)*, 2022, pp. 82–86.

[26] C. Zheng and Y. Su, "Pid control of robot manipulators in task space," in *2010 8th World Congress on Intelligent Control and Automation*, 2010, pp. 1794–1799.

[27] T. Hsia, "Adaptive control of robot manipulators - a review," in *Proceedings. 1986 IEEE International Conference on Robotics and Automation*, vol. 3, 1986, pp. 183–189.

[28] M. Spong, "On the robust control of robot manipulators," *IEEE Transactions on Automatic Control*, vol. 37, no. 11, pp. 1782–1786, 1992.

[29] P. Poignet and M. Gautier, "Nonlinear model predictive control of a robot manipulator," in *6th International Workshop on Advanced Motion Control. Proceedings (Cat. No.00TH8494)*, 2000, pp. 401–406.

[30] 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. [Online]. Available: https://doi.org/10.1177/0278364913495721

[31] L. Tai, J. Zhang, M. Liu, J. Boedecker, and W. Burgard, "A survey of deep network solutions for learning control in robotics: From reinforcement to imitation," 2018.

[32] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, jun 2015, pp. 3431–3440. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/CVPR.2015.7298965

[33] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, Eds. Springer International Publishing, 2015, pp. 234–241.

[34] "Wolfram demonstrations project." [Online]. Available: https://demonstrations.wolfram.com/ShortestPathForTheDubinsCar/

[35] "Wolfram demonstrations project." [Online]. Available: https://demonstrations.wolfram.com/RapidlyExploringRandomTreeRRTAndRRT/

[36] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *IEEE International Conference on Computer Vision*, 2015, pp. 1026–1034.

[37] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations*, 2015.

[38] S. Mohapatra, G. K. Pati, M. Mishra, and T. Swarnkar, "Upolyseg: A u-net-based polyp segmentation network using colonoscopy images," *Gastroenterology Insights*, vol. 13, no. 3, pp. 264–274, 2022. [Online]. Available: https://www.mdpi.com/2036-7422/13/3/27

[39] H.-E. Huang, S.-Y. Yen, C.-F. Chu, F.-M. Suk, G.-S. Lien, and C.-W. Liu, "Autonomous navigation of a magnetic colonoscope using force sensing and a heuristic search algorithm," *Scientific reports*, vol. 11, no. 1, p. 16491, 2021.

[40] [Online]. Available: https://starslab.ca/people/prof-jonathan-kelly/

[41] Y. Lin and S. Saripalli, "Path planning using 3d dubins curve for unmanned aerial vehicles," in *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2014, pp. 296–304.