Deep Learning Kinematics for Continuum Wire Manipulator

ZHEYU ZHOU, WENXUAN LI, ZHEYUAN ZHANG

Mentor: Dr. David Usevitch,
Professor Iulian Iordachita

Contents

1 Abstract **1**

2 Motivation **1**

3 Background and Prior Work **2**

4 Timeline and Dependencies **3**
4.1 Timeline . . . 3
4.2 Dependencies . . . 3

5 Milestones and Deliverable **4**
5.1 Milestones . . . 4
5.2 Deliverable . . . 4

6 Technical Approach **6**
6.1 Motor Control Design . . . 6
6.1.1 Motor Connection and Controller Test . . . 6
6.1.2 Angle Controller with PID . . . 6
6.1.3 Integration with the Computer Vision System . . . 8
6.2 Dual-Camera Computer Vision Based Data Generation . . . 9
6.2.1 Image Capturing . . . 9
6.2.2 Image Processing . . . 9
6.2.3 Shape from Silhouette (SfS) Algorithm . . . 9
6.2.4 Dual-Camera Calibration . . . 10
6.2.5 Data Generation . . . 10
6.3 Deep Learning for Forward Kinematics . . . 11
6.3.1 Data Collection and Preprocessing . . . 11
6.3.2 Model Development . . . 12
6.3.3 Training . . . 12
6.3.4 Prediction . . . 13
6.3.5 Adaptation for Transfer Learning . . . 13
6.4 Transfer Learning . . . 14
6.4.1 Overview of Transfer Learning . . . 14
6.4.2 Transfer Learning Implementation . . . 14

7 Experiment and Test **15**
7.1 Test Station Setup . . . 15
7.2 Data Collection . . . 16
7.3 Data Processing . . . 17
7.4 Deep Learning Model . . . 17
7.4.1 Conducting the Experiment . . . 17
7.4.2 Assess Model Performance . . . 18
7.4.3 Transfer Learning . . . 19

8 Result **20**
8.1 Deep Learning Model 20
8.2 Transfer Learning 21

9 Discussion **22**
9.1 Deep Learning Model 22
9.2 Transfer Learning 22

10 Future Work and Conclusion **23**
10.1 Future Work 23
10.1.1 Error Validation 23
10.1.2 Deep Learning Improvement 23
10.1.3 Transfer Learning Improvement 23
10.2 Conclusion 24

11 Management Plan **25**
11.1 Team Members and Roles 25
11.2 Meeting Schedule and Communication Platforms 26

12 Reading List **26**

References **28**

1 Abstract

This report presents the result of studying the kinematics of a pre-shaped continuum Nitinol wire manipulator for minimal invasive eye surgery. Nitinol is commonly used as a superelastic metal, and shape memory in a variety of micro-surgical tasks. This work develops a new kind of continuum wire manipulator (CWM) which is made of Nitinol wire through some manufacturing process to form the wire into a shape with a circular loop tip section and crossed ends to actuate the end effector which can be bent by rotating the distal ends of the curved wire. Thus, the configuration is proposed for the steering through small tortuous anatomy in the human body when minimal invasive surgery is trying to reach a desired target. However, unlike to traditional robots, it is difficult to predict soft continuum body kinematics and deformations accurately. Therefore, this proposal presents one possible method using deep learning to find the forward kinematics mapping between motor-actuated input angles to the Nitinol soft robot manipulator output configuration. The general kinematics equations developed from this project could be fine-tuned for many Nitinol continuum wire manipulator designs using transfer learning methods developed here.

2 Motivation

Retinal diseases affect approximately 200 million people worldwide, and degenerative diseases are a leading cause of vision loss [1]. In order to perform a safe and controlled surgery in the confined retinal structure, a soft robot — Continuum Wire Manipulator has been developed and applied in medical procedure. This is where soft robots, such as concentric tube robots, can be particularly useful due to their flexibility and ability to navigate tight spaces [1]. The control of soft robot position and movement is more challenging than rigid body robots. This challenge led to a great deal of research on concentric tube robots kinematics to find out the mapping between the input of motor actuation and the output of soft robot configuration.

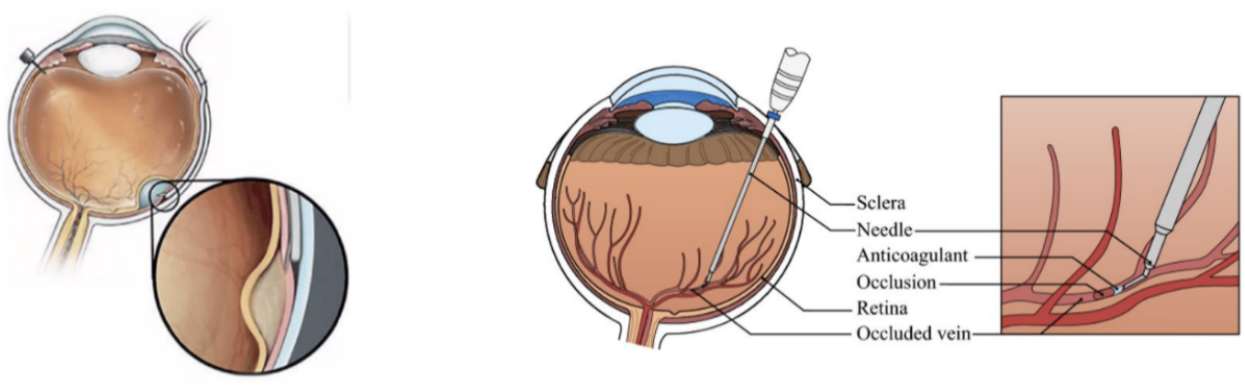

Figure 1: Retinal Disease Symptom Areas and Soft Robot for Vein Cannulation

3 Background and Prior Work

In previous studies, two approaches have been applied to finding the kinematics of concentric tube robots. One of the approaches was examining the mechanics of the concentric tube robot systems and provides general kinematics equations [2]. The other one built a simulated dataset based on previously found mechanics equation for deep learning and then fine-tuned with transfer learning to get the equations for specific concentric tube setups[3].

For the Nitinol CWM, due to its novel geometric configuration and its mechanical property of elasticity and shape memory, it potentially offers more actuation, range of motion, and flexibility than concentric tube robots, can be even smaller in diameter, and requires fewer parts and manufacturing methods. However, no previous studies have been done to find out its general kinematics. So a CWM study cannot be built upon a simulated dataset based on existing data. Instead, a deep learning method was proposed to study the general kinematics equations for Nitinol continuum wires manipulator [4]. A huge number of labeled images will be collected as input data from real-world measurements of the CWM using outputs from data gathered via motor encoder and computer vision algorithms.

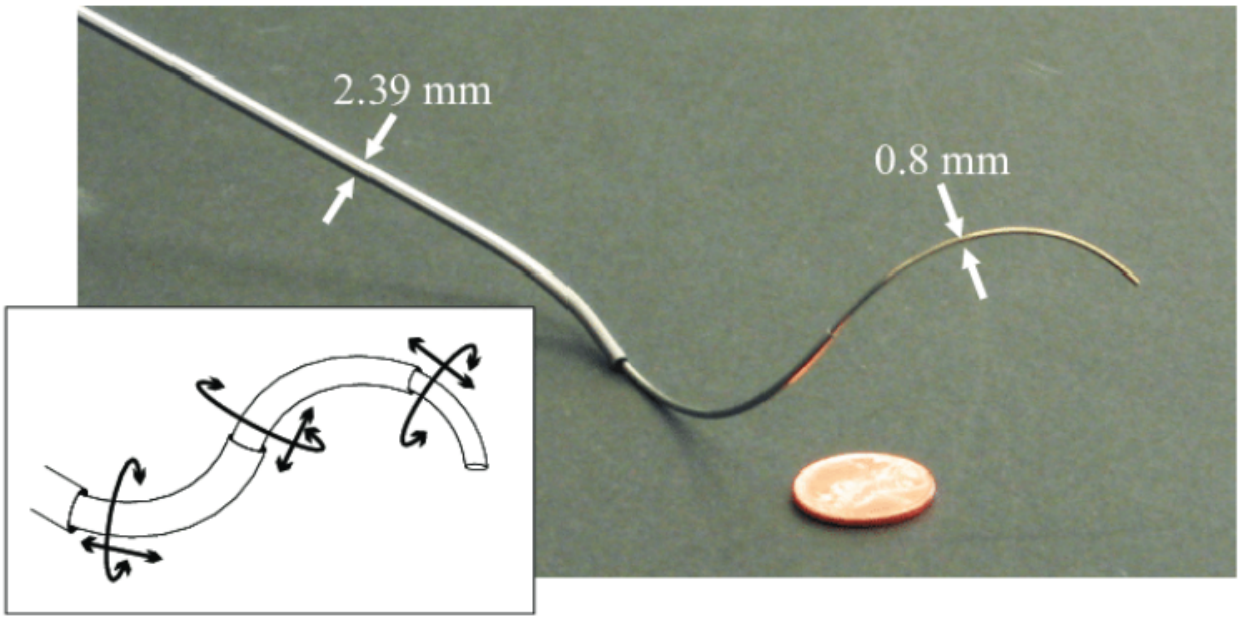

Figure 2: Continuum Wires Manipulator (Bajo et al., 2013).

Once the general kinematics equation is tested and validated through deep learning, it can then be utilized with transfer learning to quickly fine-tune toward some specific CWM designs. These specific designs would have similar but different geometric parameters and their data would be separately collected for transfer learning. Since each Nitinol wire manipulator would have some physical parameter deviations, the transfer-ability from the general deep learning-based model would reduce required data points, therefore saving time in the long run and effort for data collection and training.

4 Timeline and Dependencies

4.1 Timeline

The development schedule and deliverable deadlines are defined in Figure 3. The blue text represents the deliverable needed to meet the minimum product. The purple area represents the expected deliverable, while the bottom pink part represents the maximum deliverable.

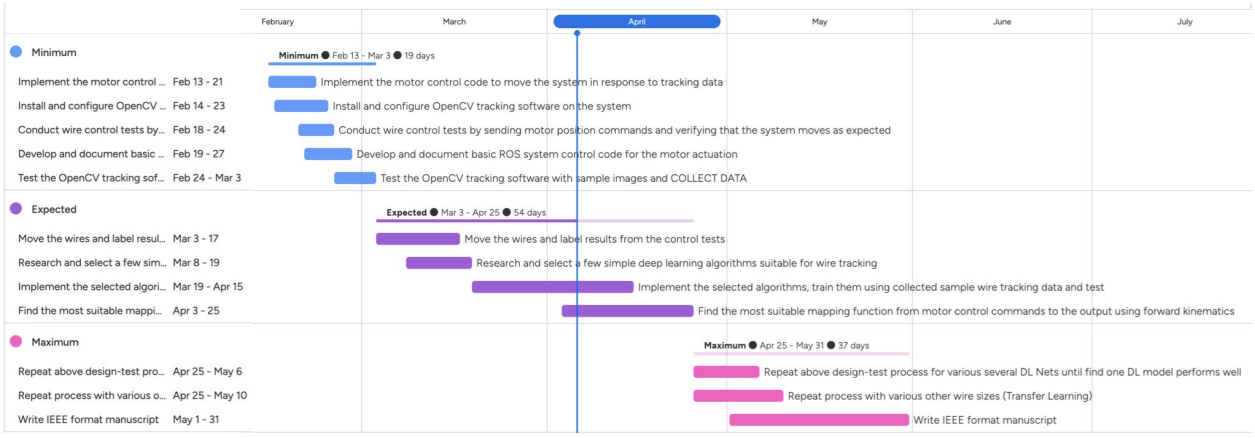

Figure 3: Timeline of the Project

4.2 Dependencies

The project hinges on several key dependencies with various deadlines. Nitinol wire fabrication has been slated for completion by April 8th, 2023, while decisions regarding GPU rental for deep learning training are expected by March 21st, 2023. Camera and motor mounting, as well as the checkerboard for computer vision calibration, involve 3D printing of auxiliary parts, with actual finished dates ranging from March 12nd to March 14th, 2023. Lastly, the control system, which integrates computer vision, motor control, and data collection, was undergoing testing, with full integration anticipated by March 21th, 2023.

Table 1: Project Needs, Status, and Deadlines

	Need	Status	Follow up	Buffer	Actual
Nitinol Wires	Data Collection & Validation	Manufacturing	Fabricate by Dr. Usevitch	4/2/2023	4/8/2023
Computation Resources (GPU)	Deep Learning Training	Comparing Google, Amazon educational grant	Rent GPU	3/16/2023	3/21/2023
Camera Mounting	CV & Data Collection	Camera mount partially installed	3D print auxiliary mounting parts	3/06/2023	3/12/2023
Motor Mounting	3D print auxiliary mounting parts	CAD Designing	3D print auxiliary mounting parts	3/08/2023	3/14/2023
Checker Board	CV calibration	Checker Board Printed, 3D print ongoing	3D print auxiliary mounting parts	3/12/2023	3/18/2023
ROS	CV, Motor, Data Integration	Installed and Testing	Integrate separate system	3/15/2023	3/21/2023

5 Milestones and Deliverable

5.1 Milestones

The project milestones and their respective time periods are outlined in the table provided. Milestone 1 focuses on setting up the OpenCV environment, motor control testing, hardware setup, and data collection. This involves tasks such as image capturing from two cameras, processing images, 3D data output, motor angle control with PID, ROS system setup, encoder data collection, image data matching, nitinol wire manufacturing, motor and camera mounting, as well as creating datasets for deep learning and transfer learning methods. Milestone 2 involves the development of a deep learning model, where a fully connected feed-forward neural network with 2 hidden layers is created using the Adam Optimizer and ReLU activation function. Finally, Milestone 3 is dedicated to developing a transfer learning method, where the pre-trained model is fine-tuned for five specific wires and the performance of transfer learning is tested and validated over the corresponding datasets. After a whole semester's work, except wrapping up all the materials for writing an IEEE format manuscript, we completed all the milestones.

5.2 Deliverable

MINIMAL (Actual Delivery Date: 03/16/2023):
• Setup OpenCV tracking software and confirm track wire (Matlab or Python)
• Develop basic RoboClaw system control code for single motor actuation - and document
• Code Wire control tests (motor positions) to move the system - Simple Code

EXPECTED (Actual Delivery Date: 04/15/2023):
• Move the Wires and label results from the control tests
• Test Simple Deep Learning Algorithms for Tracking the wire
• Map Control of Motor to the output (Forward kinematics)

IDEAL (Actual Delivery Date: 05/09/2023):
• Test and compare various several Deep Learning Nets for control of the system
• Repeat process with various other wire sizes
• Write IEEE format manuscript

Table 2: Project Milestones and Time Periods

Milestone	**Subtasks**	**Time Period**
Milestone 1	OpenCV Setup: create environment, image capturing from two cameras, processing image and shape carving, and 3D data output	02/25/2023 - 03/10/2023
	Motor Control Test: motor angle control with PID, ROS system setup and encoder data collection and image data matching	02/25/2023 - 03/08/2023
	Hardware Setup: nitinol wire manufacturing, motor mounting, and camera mounting and CV auxiliary parts	02/25/2023 - 03/18/2023
	Data Collection: create datasets for Deep Learning and transfer learning method	03/15/2023 - 04/30/2023
Milestone 2	Develop the Deep Learning Model: create a fully connected feed-forward neural network with 2 hidden layers using Adam Optimizer and ReLU activation function	04/02/2023 - 04/25/2023
Milestone 3	Develop the Transfer Learning Method: fine-tune the pre-trained model for five specific wires, test and validate the performance of transfer learning over the corresponding datasets	04/21/2023 - 05/09/2023

6 Technical Approach

6.1 Motor Control Design

6.1.1 Motor Connection and Controller Test

To obtain the required data, it was necessary to set up a microcontroller for managing the electrical connections. Our initial strategy involved using Arduino code to implement a PID (Proportional-Integral-Derivative) control system to manage a motor, which would rotate one end of the CWM (Continuum Wire Manipulator). Simultaneously, the other end would be governed by a motion control system using gears.

The primary step in establishing control of the motor with the Arduino board involved connecting the motor to the board using a suitable motor driver circuit as the controller. This circuit is vital for supplying the necessary power and control signals to the motor. The choice of the motor driver circuit relies on the motor's specifications, such as voltage and current requirements. Connections between the motors and the driver circuit can be established using wires or connectors. The motor driver circuit can then be linked to the Arduino board through digital pins for control, as illustrated in Figure 4.

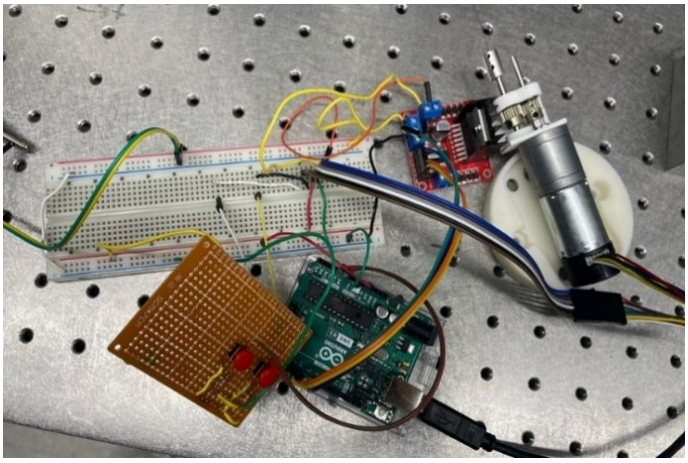

Figure 4: Design of the Arduino Controller

Following the completion of initial testing, the continuous motion of the motor was managed using electrical switches and the Arduino system. Once it was established that the objective was to employ the control algorithm to manipulate the motor's rotation to a specific angle, we consulted our mentor and decided to use RoboClaw as the controller. We then designed the necessary circuitry to manage the motor and refined the entire system by neatly organizing the wires, as depicted in Figure 5.

6.1.2 Angle Controller with PID

The purpose of implementing a Proportional-Integral-Derivative (PID) controller in conjunction with the RoboClaw Package was to control the motor's rotation, enabling it to

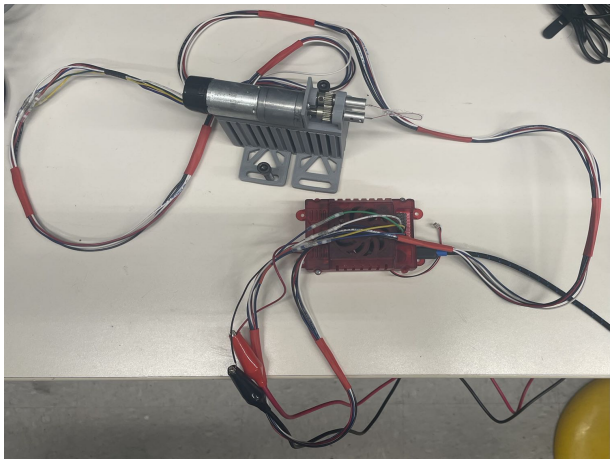

Figure 5: Connection of the RoboClaw Controller with the Motor

reach and maintain a specific angle. The PID controller functions by calculating the error between the desired angle and the motor's current angle, subsequently adjusting the motor's output as needed. The three components of the PID controller—proportional, integral, and derivative—cooperate to minimize error and stabilize the motor's position as shown in Figure 6.

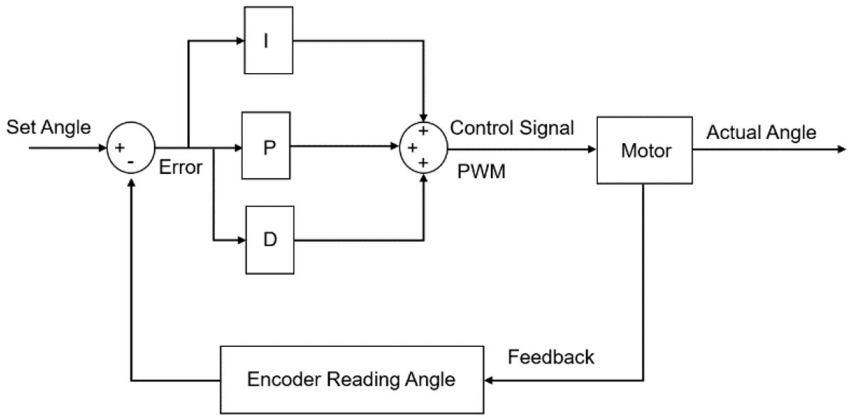

Figure 6: PID Logic Controller

First, the RoboClaw Package and required libraries must be installed and imported into the project. The RoboClaw library simplifies the PID algorithm implementation by providing an interface for communicating with the motor controller. Next, the PID constants (Kp, Ki, and Kd) are defined with the test. These values determine the behavior of the PID controller and need to be tuned for optimal performance based on the specific motor and application. We set Kp, Ki, and Kd with values of 1, 0.5, 0.1.

After defining the constants, the RoboClaw motor controller should be initialized by specifying the appropriate communication port and address. The PID coefficients for the motor are then set using the RoboClaw library functions, the function named SetM1PositionPID

is used to set it. A function is created to move the motor to the desired angle. This function will convert the desired angle to a corresponding motor position value, which depends on the motor's specifications and gearing. The function command the motor to move to the target position. Lastly, to control the motor's rotation and maintain it at a specific angle, the motor-moving function should be called with the desired angle as an argument. As the motor moves towards the target angle, the PID controller will continuously adjust the motor's output to minimize the error and stabilize the motor's position at the desired angle.

6.1.3 Integration with the Computer Vision System

To ensure the successful operation of the entire testbed, we employed two distinct algorithms. The first algorithm's objective was to verify whether the wire had been rotated to the correct angle using the PID-controlled motor. The second algorithm, initiated by the main Python program, captured images of the wire at the corresponding angles and stored both the angle and image information.

During the experimentation phase, the PID control sometimes required a longer execution time. However, given the necessity to collect a large volume of data, often in the thousands, we developed a conditional program to optimize the process. This program assessed whether the encoder readings had reached the target value, and upon achieving the desired value, the system would exit the PID control loop and proceed directly to the computer vision algorithm. This approach significantly improved the efficiency of the data collection process. The Figure 7 shows the system construction.

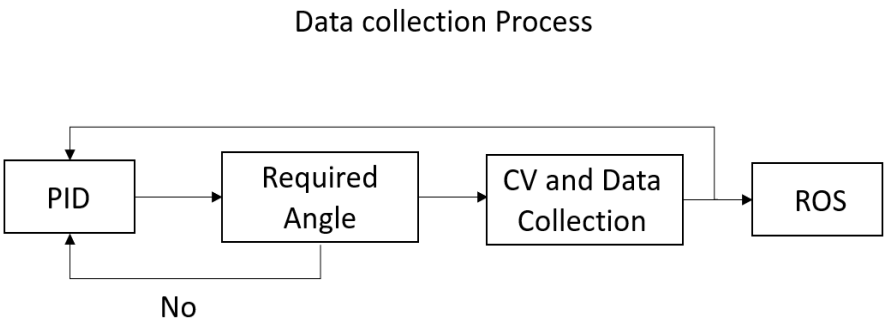

Figure 7: Integration with the Computer Vision System and Control Design

Finally, we utilized the Robot Operating System (ROS) nodes to export and store the collected data. This system enabled seamless integration of the motor control and computer vision components, facilitating comprehensive data analysis and further optimization of the testbed.

6.2 Dual-Camera Computer Vision Based Data Generation

To track the CWM and read the configuration into 3D space, this section provides the technical approaches to accomplish the computer vision task.

6.2.1 Image Capturing

For a project that would apply Deep Learning and Transfer Learning, images are very important for the learning process. All the images would be taken from the Andonstar AD407 digital camera shown in figure 4. It could provide HD 1280x720, 120 FPS images, and could be uploaded to a PC through a USB cable. The "usb_cam" ROS package is required to send real-time images from the USB-connected cameras to the PC. After confirming the PC could receive real-time images from the camera, the OpenCV package could be used to capture the uploaded images for later image processing (Andonstar).

6.2.2 Image Processing

To extract useful information from 2D images, Chroma Key Masking would be applied to convert the green curtain background to black, and red colored wire to white, which is a binary image like figure X shown. That would allow the wire to have a better contrast when compared to the background, therefore later we could easily find the wire pixels in the image frame.

6.2.3 Shape from Silhouette (SfS) Algorithm

The Shape from Silhouette (SfS) algorithm could reconstructs the 3D shape of an object from its 2D silhouette, which is the contour of the object in an image or a series of images. The process involves first extracting the silhouette from one or more images using segmentation techniques. Next, a 3D shape is back-projected from the silhouettes on different image planes to obtain a predicted the 3D shape. The idea of SfS algorithm is shown in Figure8 (Cheung, 2005) (Hartley, 2004).

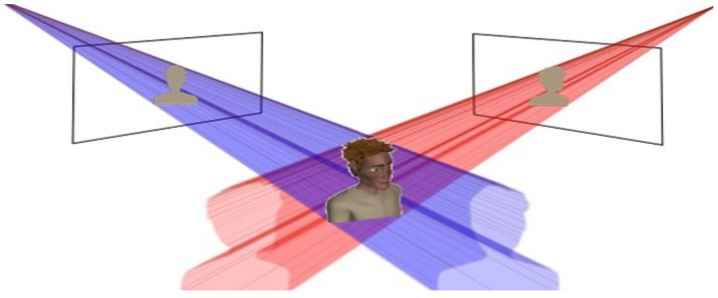

Figure 8: Example of Shape from Silhouette (SfS) algorithm (John, 2011).

6.2.4 Dual-Camera Calibration

Dual-camera calibration is the process of determining the relative positions and orientations of two cameras with respect to each other. These parameters are used to transform the 2D image coordinates into 3D world coordinates through point cloud registration, which is necessary for the back-projection step of the Shape from Silhouette algorithm.

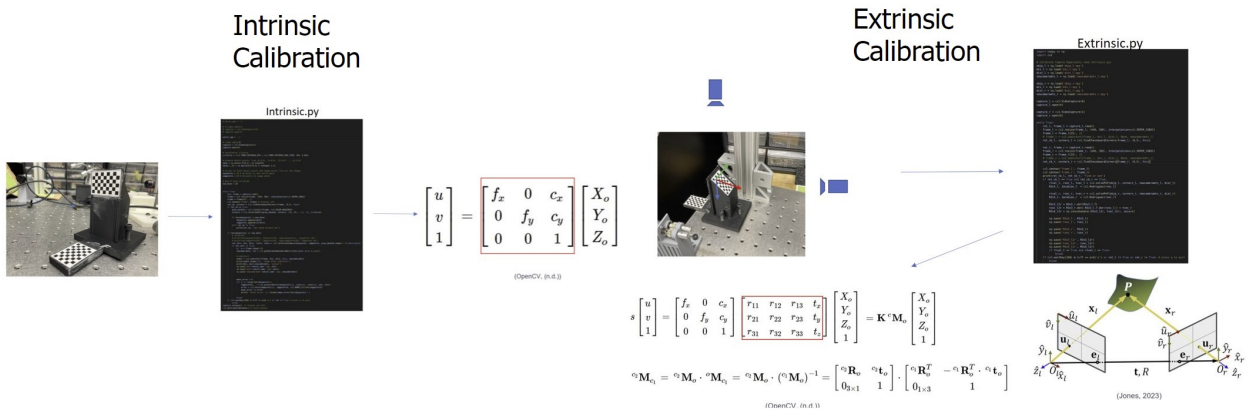

Figure 9: Dual-Camera setup with calibration checker boards.

6.2.5 Data Generation

The data is generated using the Shape from Silhouette (SfS) algorithm with the calibrated camera-world transformation matrices and binary image processed by the Chroma key masking. As Figure10 shows, the SfS algorithm would process the binary image from top and side cameras and find the intersected point clouds that could represent the wire 3D configuration.

The point cloud obtained from SfS algorithm would form a lot of strip-like clusters. The reason is the side-view camera is closer to the object due to the size limitation of the testbed. Thus, the same wire would look a little thicker in the side view. After the binary images are back-projected to the 3D space using the SfS algorithm, the intersected points also be thicker from the side view where the points form strip-like clusters and represent a wire shape.

After that, we applied the Gaussian Mixture Model clustering method which could help to isolate the clustering data and find the mean point of each cluster. The Gaussian Mixture Model has the advantage to find a cluster that has almost the same shape based on "tied" covariance. For each cluster that represents a strip-like point cloud group, the mean points of each cluster could clearly represent the wire configuration in the 3D space.

To find the optimal number of clusters that could represent the wire configuration. We tried Silhouette Analysis which has a score range between -1 to 1. A score close to 1 means

that the clusters are very dense and nicely separated, score of 0 means clusters are overlapping, and a score less than 0 means the data belonging to clusters are incorrect.

$$s(i) = \frac{b(i) - a(i)}{max(a(i), b(i))} \tag{1}$$

Also, we tried Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC) to help us determine the optimal number of clusters. The smaller AIC and BIC result indicates a better choice of cluster number. However, as Figure11 shows, the Silhouette Analysis indicates the smaller cluster number would be greater, on the other hand, the AIC+BIC Analysis says the optimal number is about 130. With this contradiction, we decided to pick the optimal number through visualization shown in the third step and the fourth step in Figure10. Considering the correctness of representing the wire 3D configuration and the easiness of generative Deep Learning where we could not predict a very complex result with a simple lower dimension input, we selected 50 as the number of clusters for mean point extraction.

$$AIC = -2\ln(L) + 2k \tag{2}$$

$$BIC = -2\ln(L) + \ln(n) \cdot k \tag{3}$$

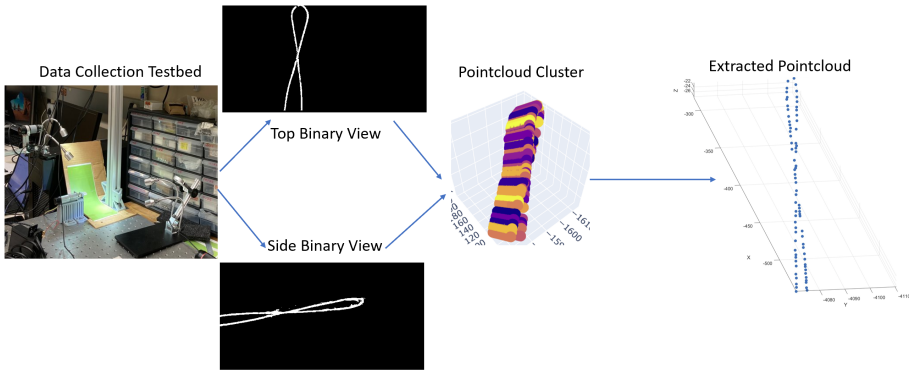

Figure 10: Extracted Wire Configuration from SfS Algorithm + GMM Clustering.

6.3 Deep Learning for Forward Kinematics

6.3.1 Data Collection and Preprocessing

To develop the forward kinematics mapping for the Nitinol soft robot manipulator, 8,752 labeled images were collected using motor encoders and computer vision. These images provided the motor-actuated input angles and the output configuration for the CWM.

The input and output data were loaded and preprocessed by dividing them into training and validation sets (80% for training and 20% for validation, accounting for 3/4 of the

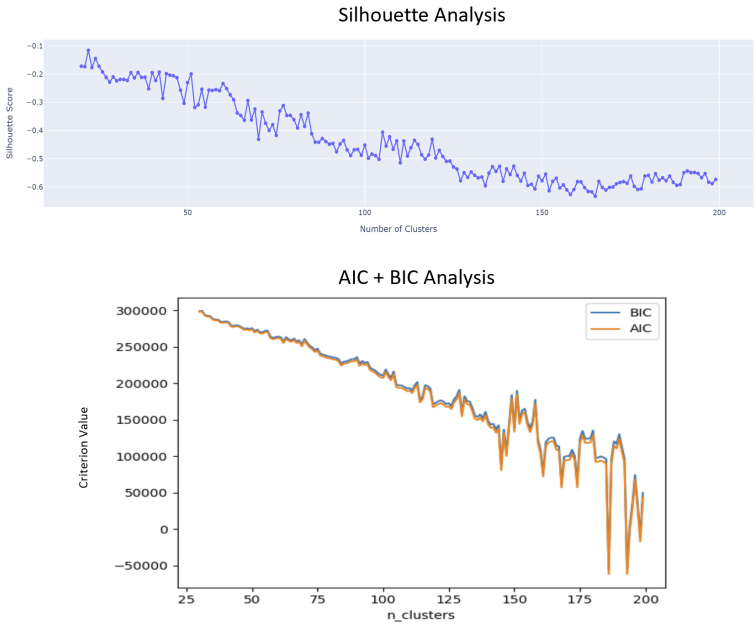

Figure 11: Searching Optimal Number of Cluster using Silhouette Analysis and Akaike information criterion + Bayesian information criterion

total data), while also reserving 1/4 of the rest of data as a test dataset to evaluate prediction performance. The input data were standardized using the StandardScaler from the scikit-learn library. The output data were reshaped and standardized in a similar manner.

6.3.2 Model Development

A multi-layer perceptron (MLP) model was employed as the deep learning model to map the input parameters, which include whether left-top or right-top, wire type, wire thickness, arc length of the wire, and motor input angle, to the output configurations. The model consisted of six hidden layers with 256, 256, 128, 128, 64, and 64 neurons, respectively, and ReLU (rectified linear unit) activation functions for each layer. We selected six hidden layer MLP model by conducting a series of experiments for testing the most suitable MLP design and the six hidden layer wins. The output layer of the MLP model had 150 neurons, corresponding to the 50 principal components and their respective 3-dimensional coordinates in the output data. Figure 12 provides an overview of the deep learning model, which contains 5 neurons in the input layer, 6 hidden layers, and 150 neurons in the output layer.

6.3.3 Training

The MLP model was trained using the mean squared error (MSE) loss function and the Adam optimization algorithm, employing a custom learning rate of 1e-5. The model underwent training for 500 epochs with a batch size of 30. The training process consisted

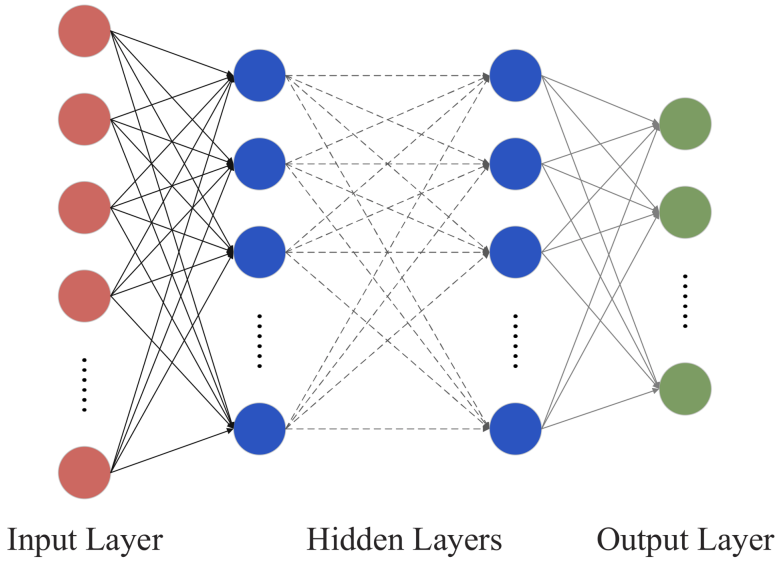

Figure 12: Overview of Deep Learning Model

of forward and backward passes, updating the model's parameters and minimizing the loss.

Throughout the training, the loss metrics, such as mean squared error, mean absolute error, and R-squared score, were computed and recorded for both the training and validation sets. These metrics were utilized to evaluate the model's performance and monitor the progress of training.

6.3.4 Prediction

After training the MLP model, it was employed to predict the output configurations based on the input motor angles. The input data were preprocessed using the input scaler, while the output data were transformed back to their original scale using the output scaler. The predicted output configurations were then compared with the actual output data to assess the performance of the model.

6.3.5 Adaptation for Transfer Learning

The deep learning model methodology presented here involves collecting and preprocessing real-world data, developing and training a deep learning model, and making predictions for the forward kinematics mapping of the Nitinol soft robot manipulator. This deep learning approach can be applied to other Nitinol continuum wire manipulator designs and can be fine-tuned using transfer learning methods for specific applications, such as minimally invasive eye surgery.

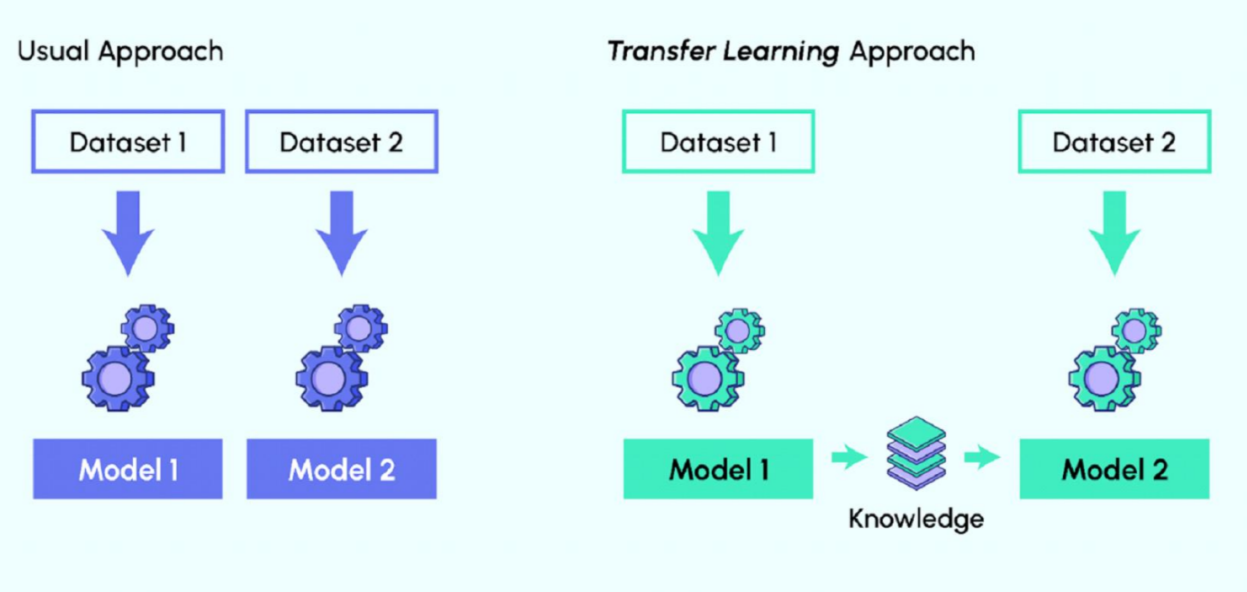

Figure 13: Overview of Transfer Learning

6.4 Transfer Learning

6.4.1 Overview of Transfer Learning

Transfer learning is a machine learning technique where a pre-trained model is used as the starting point for training a new model on a different task, as shown in Figure 13[5]. The concept behind transfer learning is that the knowledge gained by a pre-trained model on one task can be useful in solving a related task. Instead of starting from scratch, the pre-trained model provides a foundation of knowledge that can be built upon and adapted to the new task.

6.4.2 Transfer Learning Implementation

To enhance the generalizability of the developed deep learning model for learning the forward kinematics of various Nitinol continuum wire manipulator designs, a transfer learning approach was employed. Five distinct Nitinol wires were manufactured, and data were collected for each of them. Specifically, 201 data images were gathered for each of the first four wires, totaling 804 data images. The data from the first three wires (603 data images) were utilized to train a generative model for learning the forward kinematics, capturing the shared characteristics of Nitinol manipulators.

Once the generative model was trained, it served as the foundation for the transfer learning process. Only 5 data images from the fourth wire were then used to fine-tune the pre-trained model, adapting it to the specific characteristics of the fourth wire's forward kinematics. Using this fine-tuned model, predictions were made for the shape matrix of the remaining 199 data inputs. The transfer learning accuracy was assessed by cal-

culating the mean absolute error for the entire set of predictions. This transfer learning approach allowed the model to effectively generalize across different Nitinol continuum wire manipulator designs, while also efficiently adapting to the specificities of a new manipulator's forward kinematics with minimal additional data.

7 Experiment and Test

7.1 Test Station Setup

Experimental testing was performed as follows. First we connected the motor to the Arduino board using a suitable motor driver circuit and digital pins. To measure the motor's current position, we then connected an encoder or potentiometer to the Arduino board and use the analog pins for readings. Roboclaw board was then programmed for control signal computing and implement the PID control algorithm to manage the motor's position. For image capturing, we set up the Andonstar AD407 digital camera and install the "usb-cam" ROS package to send real-time images from the camera to the PC. We then utilized OpenCV for image processing, applying Bilateral Filtering and Chroma Key Masking to extract useful information from the images. Implement the Shape from Silhouette (SfS) algorithm to reconstruct the 3D shape from the 2D silhouette and calibrate the dual-camera setup to determine the relative positions and orientations of the cameras.

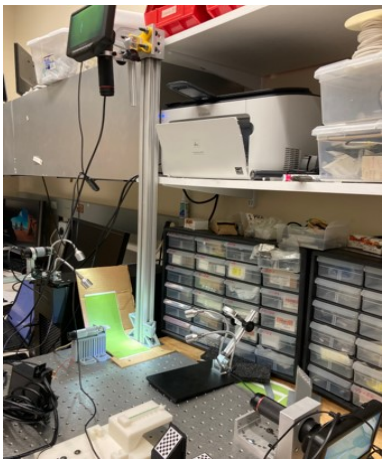

Figure 14: Test Station Overview

Figure 14 shows what the test station should look like. Moreover, throughout the course of the experiment, we discovered that different light sources demand distinct conditions. Therefore, adjusting the lighting to obtain optimal visual images is an indispensable aspect of the process.

7.2 Data Collection

The experiment involved manufacturing five distinct Nitinol wires and collecting data for each of them. Specifically, we gathered 200 data points for each of the first three wires, totaling 600 data points, while only 5 data points were collected for the fourth wire.

Figure 15: Wire Parameters

To prepare the wire for experimental data collection, additional processing steps are necessary. First, the radius of the end crossing of the CWM is controlled through a plastic sealing technique. Subsequently, a fixture is employed to secure the metal ring at the end of the wire, ensuring its stable attachment to the motor. We measured and recorded the parameters of the line as well and Figure 15 shows four wire's parameter for transfer learning. As for the parameters in our report, we encountered a discrepancy in the wire dimensions we measured using vernier calipers compared to the original manufactured dimensions. Our report mentioned 0.5mm, 0.25mm, and 0.2mm wires, whereas the actual manufactured sizes are 0.4064mm, 0.15mm, and 0.1mm square wires.

This discrepancy can be attributed to the red pigment coating that was applied to the outside of the wires. Upon drying, the pigment added a certain thickness to the wires, causing them to appear larger in size than their original manufactured dimensions. For the transfer learning process, we used the manufactured dimensions that take into account the additional thickness caused by the red pigment coating

7.3 Data Processing

We have developed a data collection procedure that tests the motor encoder numbers from -2000 to 2000, corresponding to angles from -39.69 degrees to 39.69 degrees. The procedure consists of implementing a PID control system to precisely control the motor position, allowing it to traverse all positions within the specified range. A detection program continuously monitors the motor's traversal of all positions. When the motor reaches an incremental position, a shooting command is sent to the camera, which is configured to capture images upon receiving the command. The captured images are output and stored using the ROS framework.

The data collection procedure also includes processing the captured images using a detection algorithm to obtain a set of point clouds representing the shape of the soft robot. These point clouds, along with the motor angle, are saved as the result. A clustering algorithm is then employed to convert the point clouds into a simplified 3D matrix representation of the soft robot's shape, which can be utilized for deep learning applications. This comprehensive data collection procedure enables the effective evaluation and control of the soft robot's behavior and performance throughout its range of motion.

7.4 Deep Learning Model

7.4.1 Conducting the Experiment

The Google Colab platform was utilized to leverage computational resources and facilitate collaboration during the entire training process. The experiment aimed to predict the forward kinematics of a Nitinol continuum wire manipulator using a multi-layer perceptron (MLP) model. Essential libraries, such as NumPy, PyTorch, and scikit-learn, were imported to enable data manipulation, model creation, and evaluation. Data was loaded from multiple .npz files and subsequently partitioned into training and validation sets. The data was standardized using the StandardScaler from scikit-learn.

The MLP model consisted of six hidden layers with varying numbers of neurons and ReLU activation functions. The output layer employed a sigmoid activation function to generate the final predictions. The model underwent training for 500 epochs with a batch size of 30 and a custom learning rate of 0.00001. A total number of 8,752 data samples were used during training, validation and testing process. The mean squared error (MSE) was utilized as the loss function, and the Adam optimizer was employed for weight updates. Training and validation losses, as well as mean absolute errors, were recorded and plotted over the epochs.

Upon completion of the training process, the trained model was used to make predictions on a test set of input data. These predictions were compared to the ground truth values, and the differences were analyzed to evaluate the model's performance in predicting the forward kinematics of the manipulator.

Initially, the model was configured with only two hidden layers, while other settings remained the same. The number of layers was altered in subsequent iterations, and the experiment was repeated for predictions of learning forward kinematics. This process continued until the most optimized MLP settings were identified, taking into account both time efficiency and prediction performance.

7.4.2 Assess Model Performance

Mean Squared Error Mean squared error (MSE) was employed as the loss function and as one of the performance assessments for our model. MSE is a widely used performance metric for regression tasks in deep learning and other machine learning models. It quantifies the difference between the predicted values generated by the model and the true values in the dataset. The main idea behind MSE is to penalize larger errors more than smaller errors, as the square of the difference magnifies the impact of larger errors.

MSE is calculated using the following formula:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{4}$$

In this formula, n represents the number of data points (samples) in the dataset, y_i is the true value of the i-th data point, and $\hat{y}_i$ is the predicted value of the i-th data point.

A lower MSE indicates better model performance, as it signifies that the model's predictions are closer to the true values. However, one limitation of MSE is its sensitivity to outliers. The square of the difference can lead to a large value if an outlier is present in the dataset.

Mean Absolute Error Another crucial assessment for our model performance is mean absolute error (MAE). It calculates the average absolute difference between the predicted values generated by the model and the true values in the dataset. Unlike MSE, MAE does not penalize larger errors more than smaller ones, making it less sensitive to outliers, which solves the issue from MSE.

MAE is calculated using the following formula:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Where n is the number of data points (samples) in the dataset, y_i is the true value of the i-th data point, and $\hat{y}_i$ is the predicted value of the i-th data point.

A lower MAE indicates better model performance, as it signifies that the model's predictions are closer to the true values. Since MAE does not square the differences like MSE, it is less sensitive to outliers, making it a better choice for datasets with extreme values or noisy data.

7.4.3 Transfer Learning

Baseline Experiment with 1 Data Image To establish a baseline for comparison, an experiment was conducted in which the model was trained solely on one data image from the fourth wire, without using any pre-trained weights. The same training settings were used, with a learning rate of 0.01, 1,000 epochs, and a batch size of 10. This experiment aimed to evaluate the model's performance when trained exclusively on a single data image without leveraging any prior knowledge from the initial training on the first three wires.

Baseline Experiment with 5 Data Images A second baseline experiment was set up, in which the model was trained only on five data images from the fourth wire, again without the benefit of pre-trained weights. The training settings remained consistent, with a learning rate of 0.01, 1,000 epochs, and a batch size of 10. This experiment aimed to assess the model's performance when given a slightly larger amount of data from the new wire, without the advantage of transfer learning.

Baseline Experiment with 10 Data Images Finally, a third baseline experiment was conducted, training the model exclusively on 10 data images from the fourth wire without utilizing pre-trained weights. The training settings were kept consistent, using a learning rate of 0.01, 1,000 epochs, and a batch size of 10. This experiment aimed to further investigate the model's ability to learn the characteristics of the new wire based on an increased amount of data, while not benefiting from any prior knowledge obtained through transfer learning.

Transfer Learning with 1 Data Image In the transfer learning experiment, the pre-trained generative model was fine-tuned using only one data image from the fourth wire. The model's weights were updated using a different learning rate of 0.01, with 1,000 epochs and a batch size of 10. This fine-tuning process allowed the model to adapt to the specific characteristics of the fourth wire's forward kinematics while leveraging the knowledge gained from the initial training on the first three wires. This experiment aimed to test the effectiveness of the transfer learning approach in adapting the model to new data with minimal additional input.

Transfer Learning with 5 Data Images To further test the model's adaptability, another transfer learning experiment was conducted using five data images from the fourth wire. The same fine-tuning process was applied, with a learning rate of 0.01, 1,000 epochs, and a batch size of 10. This experiment aimed to assess the improvement in the model's per-

formance when given a slightly larger amount of data from the new wire and to evaluate the benefits of transfer learning in this context.

Transfer Learning with 10 Data Images Lastly, a third transfer learning experiment was performed using 10 data images from the fourth wire. The fine-tuning process remained consistent with the previous experiments, using a learning rate of 0.01, 1,000 epochs, and a batch size of 10. This experiment aimed to further explore the model's capacity to adapt to the new wire's characteristics with an increased amount of data, as well as to investigate the scalability of the transfer learning approach for this specific application.

These comparative experiments allowed for the evaluation of the effectiveness of transfer learning in this specific application, by contrasting the performance of the models trained with and without pre-trained weights. The results provide insights into the potential advantages and limitations of using transfer learning for fine-tuning models in the context of Nitinol manipulators' forward kinematics.

8 Result

8.1 Deep Learning Model

The results from our experiments are presented in Table 3, which displays the Mean Absolute Error (MAE) comparison for different deep learning models with varying numbers of hidden layers. It is evident that as the number of hidden layers increases, the MAE significantly decreases, indicating improved performance of the models. The most substantial improvement is observed when moving from a model with three hidden layers to one with four hidden layers, where the MAE reduces from 67.52 mm to 16.27 mm. The model with six hidden layers achieves an MAE of 0.85 mm, while the model with seven hidden layers further improves the performance slightly, resulting in an MAE of 0.83 mm.

Table 3: Comparison of MAE for Various Deep Learning Models with Measurement Precision in Millimeters

	MAE
2 hidden layers	130.79
3 hidden layers	67.52
4 hidden layers	16.27
5 hidden layers	3.92
6 hidden layers	0.85
7 hidden layers	0.83

As depicted in Figure 16, we present a visualization of the real-world wire shape alongside the predicted wire shape. The red lines represent the predictions, while the blue

lines indicate the ground truth. As can be observed, our model closely approximates the ground truth by effectively learning the underlying shape of the real-world data. This demonstrates the success of our approach in accurately predicting the forward kinematics of Nitinol manipulators.

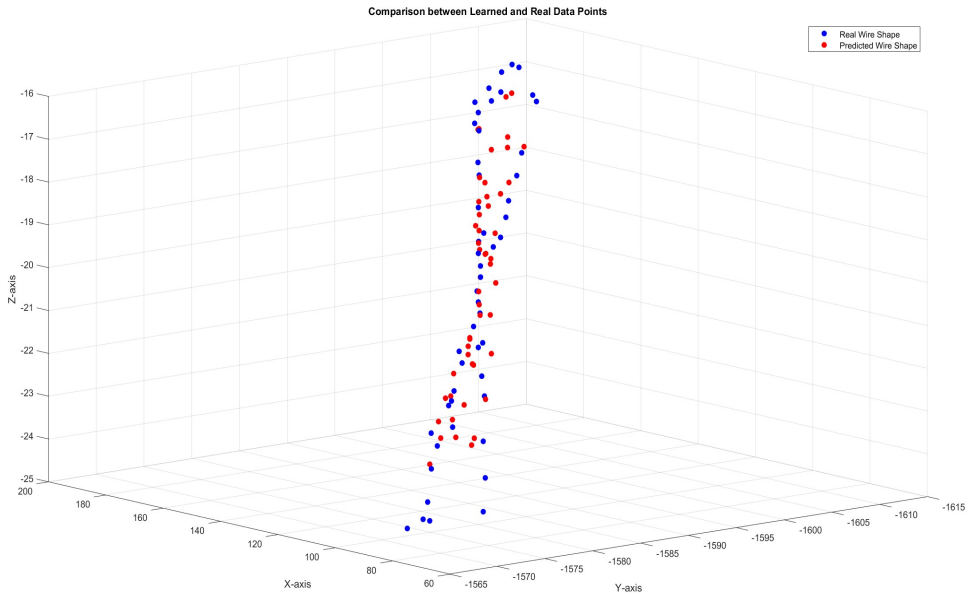

Figure 16: Comparison between Learned and Real Data Points

8.2 Transfer Learning

The results of our transfer learning experiments are presented in Table 4, which compares the errors obtained when training from scratch and when using pre-training followed by fine-tuning for different data samples used as fine-tuning datasets. The table highlights the substantial improvements achieved through transfer learning. From the table, it is evident that the pre-training and fine-tuning process significantly reduces the error compared to training from scratch. The table is divided into two main categories: models trained from scratch and models that underwent a pre-training and fine-tuning process. The columns represent the size of the fine-tuning datasets, including one sample, five samples, and seven samples.

For the models trained from scratch, the MAE values are significantly higher, ranging from 122.248 mm with one sample to 61.793 mm with seven samples. This indicates that without pre-training, the models struggle to capture the underlying relationships in the data, especially when the size of the fine-tuning dataset is limited.

On the other hand, the pre-training and fine-tuning approach demonstrates substantially lower MAE values across all fine-tuning dataset sizes. For one sample, the MAE is 27.255 mm, while for five samples, it is 21.143 mm, and for seven samples, it is 20.789 mm. These

results highlight the effectiveness of transfer learning in improving model performance, even when the fine-tuning dataset is small.

Table 4: Comparison of MAE for Transfer Learning in Different Fine-tuning Datasets with Measurement Precision in Millimeters

	1 Sample	**5 Samples**	**7 Samples**
train from scratch	122.248	81.067	61.793
pre-training + fine-tuning	27.255	21.143	20.789

9 Discussion

9.1 Deep Learning Model

The findings presented in Section 8.1 emphasize the importance of model complexity in achieving accurate forward kinematics predictions for Nitinol manipulators. As the number of hidden layers increases, the model is better equipped to capture intricate and nuanced relationships within the data, resulting in enhanced performance. However, striking a balance between model complexity and computational cost is essential, as excessively complex models can lead to increased computational demands and a heightened risk of overfitting.

In our study, the model with six hidden layers achieves the ideal balance between complexity and accuracy, yielding nearly the lowest MAE among the examined models. This demonstrates that an appropriate level of model complexity can significantly improve prediction accuracy without incurring excessive computational costs. Moreover, by incorporating a dropout layer with a dropout probability of 0.5 before the output layer in an additional experiment, we further optimize our model's performance, achieving an MAE of 0.82. This improvement effectively closes the gap between our current six hidden layer MLP design and the more complex seven hidden layer MLP design, highlighting the effectiveness of regularization techniques in enhancing model performance.

9.2 Transfer Learning

The findings presented in Section 8.2 highlight the success of transfer learning in capitalizing on the knowledge acquired from the initial training on the first three wires (604 data points) to boost the model's performance on the fourth wire, even when using a limited number of samples. The error consistently diminishes as more samples are incorporated into the fine-tuning process, demonstrating the model's capacity to adjust and generalize to the unique characteristics of the fourth wire's forward kinematics. In essence, this analysis emphasizes the importance of transfer learning in augmenting the performance of deep learning models for Nitinol manipulator kinematics prediction, particularly when dealing with scarce data. This approach not only accelerates the learning process but also

contributes to more efficient and accurate predictions, ultimately enhancing the practical applicability of the developed models in real-world scenarios.

10 Future Work and Conclusion

10.1 Future Work

10.1.1 Error Validation

To validate the accuracy of the 3D images generated by the dual-camera setup, we plan to conduct a comparative analysis with computed tomography (CT) scans of the Continuum Wire Manipulator at specific angles in the future. By juxtaposing the resulting CT images and the processed data of our 3D reconstruction, we aim to quantitatively demonstrate, using suitable mathematical methods, that our data accurately represents the shape and configuration of the Continuum Wire Manipulator. This comparative analysis will not only provide valuable insights into the efficacy of our imaging system but also contribute to the development of more reliable and robust 3D reconstructions for applications in various fields, including robotics, medical imaging, and computer vision.

10.1.2 Deep Learning Improvement

In future work, we plan to further improve the deep learning component of our integrated system for the Nitinol continuum wire manipulator (CWM). Specifically, we will focus on enhancing the accuracy, efficiency, and robustness of our model by employing the following strategies:

Hyperparameter optimization: We will perform systematic hyperparameter optimization, using techniques such as grid search and Bayesian optimization, to fine-tune the model's parameters and improve its performance.

Noise reduction and data augmentation: To enhance the robustness of our model, we will investigate noise reduction techniques to minimize the impact of noise in the input data. Additionally, we will explore data augmentation methods to artificially increase the training dataset, potentially leading to improved model generalization.

Even switch to another more efficient or more complex deep learning model to achieve higher accuracy if possible. By investigating and implementing these strategies, we aim to develop a more advanced deep learning model for the CWM control system, ultimately leading to enhanced precision and adaptability in various minimally invasive surgical tasks.

10.1.3 Transfer Learning Improvement

In future work, we aim to enhance the transfer learning component of our integrated system for the Nitinol continuum wire manipulator (CWM) to better adapt to different

parameter wires and obtain a more comprehensive understanding of the wire's angles and configurations. To achieve this, we will focus on the following strategies:

1. **Expanding the pre-trained model library:** We will create a library of pre-trained models that cover a diverse range of wire parameters and manipulator configurations. This will facilitate faster adaptation to new wire types and improve the system's versatility in handling different manipulator designs.
2. **Domain adaptation techniques:** We will investigate domain adaptation techniques to bridge the gap between the source domain (pre-trained models) and the target domain (new wire parameters). This may include methods such as feature alignment, domain adversarial training, and self-supervised learning.
3. **Fine-tuning strategies:** We will explore different fine-tuning strategies to optimize the adaptation process for new wires. This could involve varying the number of layers to be fine-tuned, using learning rate schedules, or applying weight regularization techniques.
4. **Multitask learning:** We will consider employing multitask learning, where the model is trained to perform multiple tasks simultaneously, such as predicting wire angles and configurations. This approach could lead to better transfer learning performance by sharing knowledge across related tasks.
5. **Data-efficient transfer learning:** We will investigate data-efficient transfer learning methods, such as few-shot learning and meta-learning, to enable the model to adapt to new wire parameters with minimal data requirements. This will reduce the time and effort needed to collect and label data for new wire types.
6. **Performance evaluation and model selection:** We will develop a systematic approach to evaluate and compare the performance of different transfer learning models. This will help in identifying the best-performing models and refining the transfer learning process for optimal results.

By implementing these strategies, we aim to enhance the transfer learning component of our system, enabling it to adapt more effectively to various wire parameters and manipulator configurations. This will ultimately contribute to a more versatile and efficient CWM control system, capable of handling diverse applications in minimally invasive surgery.

10.2 Conclusion

In this project, we created an integrated system for controlling and generating data for a Nitinol continuum wire manipulator (CWM) by combining motor control design, computer vision, and deep learning techniques. Key components of the system include:

Motor control design using an Arduino board and PID control algorithm, later integrating the RoboClaw motor controller for enhanced precision. This allowed the motor to reach and maintain desired positions. A computer vision system that captured images of

the wire's configuration using two cameras. Image processing techniques and the Shape from Silhouette (SfS) algorithm were employed to reconstruct the wire's 3D shape. Dual-camera calibration and Gaussian Mixture Model clustering aided in accurately mapping 2D images to 3D world coordinates. A deep learning-based multi-layer perceptron (MLP) model was developed for forward kinematics mapping, predicting the wire's configuration based on motor angles using labeled data from the motor encoder and computer vision system. Transfer learning was utilized to enhance the model's generalizability, using a pre-trained model to learn the forward kinematics of a new wire by fine-tuning with a small amount of wire-specific data.

The resulting system offered a comprehensive solution for CWM control and data generation, with precise control and accurate 3D shape reconstruction. Its flexibility and adaptability, showcased through transfer learning, make it suitable for various CWM designs and tasks in minimally invasive surgery.

11 Management Plan

11.1 Team Members and Roles

The team consist of:
• Wenxuan Li (wli131@jh.edu)
Computer Science master, who is responsible for designing Deep Learning model which involves creating a fully connected feed-forward neural network with 6 hidden layers using Adam Optimizer and ReLU activation function, and Transfer Learning method which involves fine-tuning the pre-trained model, and then conducting experiments.

• Zheyu Zhou (zzhou97@jh.edu)
Mechanical Engineering master, who is responsible for conducting image capturing from two cameras, processing image and shape carving for Computer Vision part and data collection for using in deep learning and transfer learning.

• Zheyuan Zhang (zzhang270@jh.edu)
Robotics master, who is responsible for motor angle control with PID, ROS system setup and encoder data collection and image data matching, as well as motor mounting, and camera mounting.

The whole team will also have some shared responsibilities, which are the data generation, data collection, deep learning model implementation and final manuscript writing.

Mentors The mentors consist of:
• Dr. David Usevitch (usevitch@jhu.edu)
A postdoctoral researcher at Johns Hopkins University developing a variety of new medical tools including improved and smart drilling functionality for a variety orthopedic drilling tasks, and novel methods for microsurgery manipulation using new actuation

methods.

• Prof. Iulian Iordachita (iordachita@jhu.edu)
A research professor at Johns Hopkins University, a member of the Laboratory for Computational Sensing and Robotics, and director of the Advanced Medical Instrumentation and Robotics (AMIRo) Research Lab. His research focuses on medical and surgical robotics; medical instrumentation and smart surgical tools; image-guided and computer-assisted surgery; and mechanisms and mechanical transmissions for robots.

11.2 Meeting Schedule and Communication Platforms

Scheduled meetings for this project occurred on a weekly basis and served to update the mentor and the larger research group.

- **Group Meeting:** Every Tuesday and Thursday, 3:00 pm - 4:00 pm. The purpose of these meetings was to coordinate efforts with the computer vision and hardware teams, hold each other accountable, and address concerns.
- **Meeting with Mentors:** Friday, 2:00 pm - 3:00 pm. The purpose of this meeting was to discuss technical challenges faced during the week, update mentors on progress, and coordinate efforts to move according to the timeline.
- **Code Platform:** GitHub was used for version control and collaboration on code.
- **Communication and File Upload:** Microsoft Teams served as the primary platform for communication and file sharing among team members.
- **Progress Demo and Document Upload:** A Wiki page was used to upload progress demos and important project documents.
- **Remote Meeting Platform:** Zoom was the platform of choice for conducting remote meetings when necessary.

12 Reading List

Bergeles, C., F. Y. Lin, and G. Z. Yang. "Concentric tube robot kinematics using neural networks." Hamlyn symposium on medical robotics. Vol. 6. 2015.

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)," 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 2018, pp. 5125-5132, doi: 10.1109/IROS.2018.8594451.

Reinhard Grassmann, Jessica Burgner-Kahrs. On the Merits of Joint Space and Orientation Representations in Learning the Forward Kinematics in SE(3). In Antonio Bicchi,

Hadas Kress-Gazit, Seth Hutchinson, editors, Robotics: Science and Systems XV, University of Freiburg, Freiburg im Breisgau, Germany, June 22-26, 2019. 2019. [doi]

A. Kuntz, A. Sethi, R. J. Webster and R. Alterovitz, "Learning the Complete Shape of Concentric Tube Robots," in IEEE Transactions on Medical Robotics and Bionics, vol. 2, no. 2, pp. 140-147, May 2020, doi: 10.1109/TMRB.2020.2974523.

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," 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Kyoto, Japan, 2022, pp. 9550-9557, doi: 10.1109/IROS47612.2022.9981719.

References

[1] E. Yasinski, "Implanted chip, natural eyesight coordinate vision in study of macular degeneration patients," *Stanford Medicine News Center*, Feb. 2022. [Online]. Available: https://med.stanford.edu/news/all-news/2022/02/macular-degeneration-retinal-implants.html.

[2] R. J. Webster, J. M. Romano, and N. J. Cowan, "Mechanics of precurved-tube continuum robots," *IEEE Transactions on Robotics*, vol. 25, pp. 67–78, 2009.

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

[4] C. Bergeles, F. Lin, and G. Yang, "Concentric tube robot kinematics using neural networks," in *Hamlyn Symposium on Medical Robotics*.

[5] M. Komendyak. "How to choose the best pre-trained model for your convolutional neural network." Accessed: 2022-04-11. (2022), [Online]. Available: https://data-science-blog.com/blog/2022/04/11/how-to-choose-the-best-pre-trained-model-for-your-convolutional-neural-network/.