CIS II: Deep Learning Kinematics for Continuum Wire Manipulator

Student: Zheyu Zhou, Wenxuan Li, Zheyuan Zhang

Mentor: Dr. David Usevitch, Prof. Iulian Iordachita

Table of contents

Abstract

This proposal presents an ideation of studying the kinematics of a pre-shaped continuum Nitinol wire manipulator for minimal invasive eye surgery. Nitinol is commonly used as a superplastic, and shape memorize in a variety of microsurgical tasks. This work develops a new kind of continuum manipulator, the continuum wire manipulator (CWM) made of Nitinol wire bent and heat treated 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. Unlike traditional robots, however, it is difficult to predictsoft continuum body kinematics and deformations accurately. Therefore, this proposal presents one possible method using deep learning to find the general 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.

Clinical Motivation

Retinal diseases affect approximately 200 million people worldwide, and degenerative diseases are a leading cause of vision loss (Usevitch, 2023). In order to perform a safe and controlled surgery in the confined retinal structure, a soft robot — CWM have 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 (Runciman, 2019). The control of soft robot position and movement is more challenging than rigid body robots. This challenge led to a great deal of researches on concentric tube robots kinematics to find out the mapping between the input of motor actuation and the output of soft robot configuration.

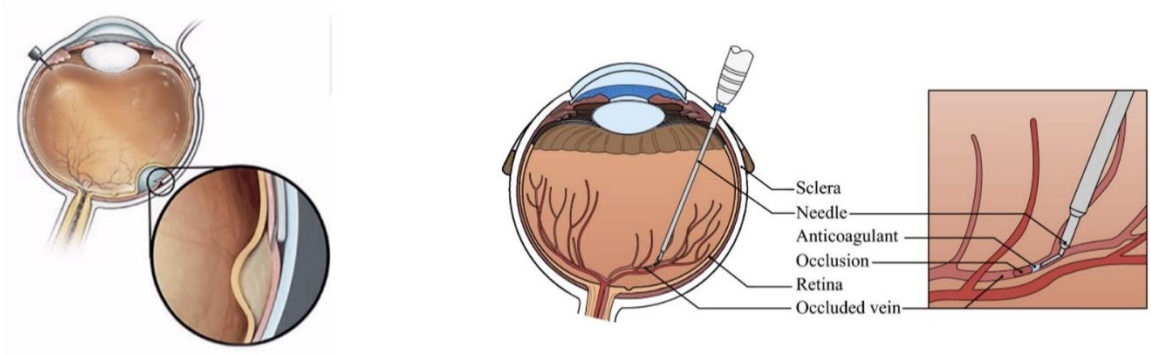


Figure 1. Suggested route for subretinal injection using the continuum manipulator.

Background

In previous studies, two approaches have been applied to finding the kinematics of concentric tube robots, one of the closest works related to continuum wire manipulators proposed in this work. One of the approaches was a study examining the mechanics of the concentric tube robot systems and provides general equations (Webster, 2009). 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 (Grassmann, 2018).

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. 10,000 labelled 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.

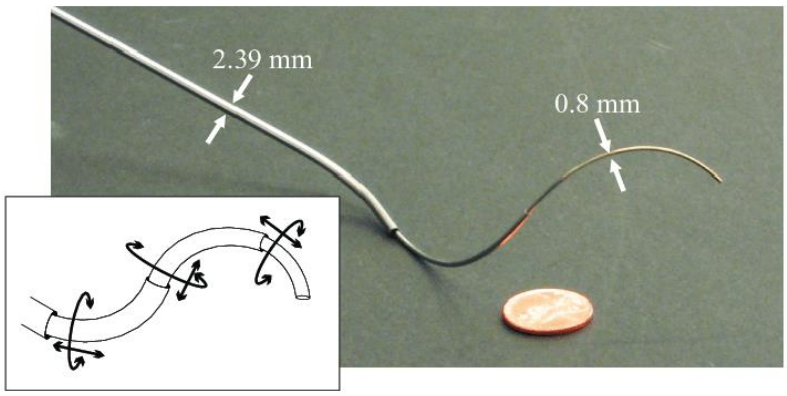


Figure 2. Continuum Wires Manipulator (Bajo et al., 2013)

Once the general kinematics equation is tested and validated using deep learning, it can then be utilized with transfer learning to quickly fine-tune towards 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 transferability 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.

Technical Approach

In this section, the preliminary steps for data collection and the testbed setup procedure are illustrated. The implementation of learning techniques for kinematics is also presented.

A. Motor Control

1. Motor Setting and Connecting

To collect data, we must first setup a motor to control the wires. We plan to do this using Arduino code implementing PID control to a motor which will twist the two ends of the CWM (one end with motion controlled via gearing. The first step in implementing PID control with an Arduino to control a motor is to connect the motor to the Arduino board using an appropriate motor driver circuit as a motor controller. The motor driver circuit is responsible for providing the necessary power and control signals to the motor. The selection of the motor driver circuit is based on the motor's specifications, such as its voltage and current requirements. The motor can be connected to the driver circuit using wires or connectors. Finally, the motor driver circuit can be connected to the Arduino board using digital pins.

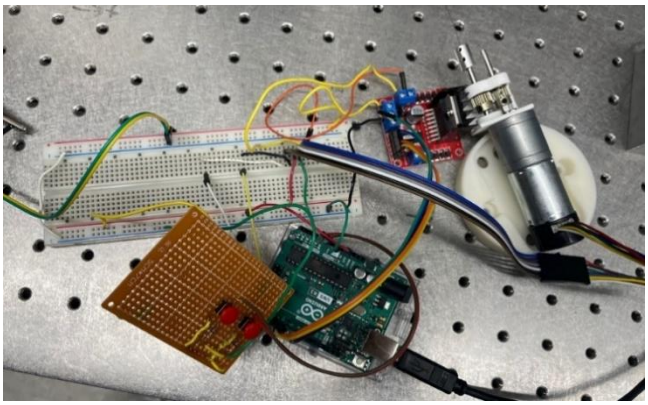


Figure 2. Motor Connection

2. Current Position measuring of the Motor

After connecting the motor to the Arduino board, we would need to program the board to read and control the current position of the motor. This is done using an encoder or a potentiometer. An encoder is a device that converts rotary motion into electrical signals that can be read by the Arduino board. A potentiometer, on the other hand, is a variable resistor that can be used to measure the position of the motor. The current position of the motor can be read using analog pins on the Arduino board, and more importantly, control of a desired turn of the wire ends can be made.

3. Control Signal Computing

Once the current position of the motor is read, the Arduino board can calculate the error between the current position and the desired position. The desired position can be set by the user, or it can be obtained from a sensor or a control system. The error signal is then used as the input to the PID controller. The PID algorithm calculates the Proportional, Integral, and Derivative components of the error signal to compute the control signal that would be sent to the motor driver to adjust the motor's position. The Proportional component provides a quick response to reduce the error, the Integral component eliminates any steady-state error, and the Derivative component improves the system's stability and reduces overshoot. To ensure stable and accurate control of the motor, the Proportional, Integral, and Derivative gains would need to be tuned based on the system's characteristics. Once the gains are set, the controller continuously monitors the motor's position and adjusts the control signal as necessary to maintain the desired position. This approach allows for precise and reliable control of the motor, making it ideal for

applications such as robotics, automation, and industrial control systems. The logic figure as shown below.

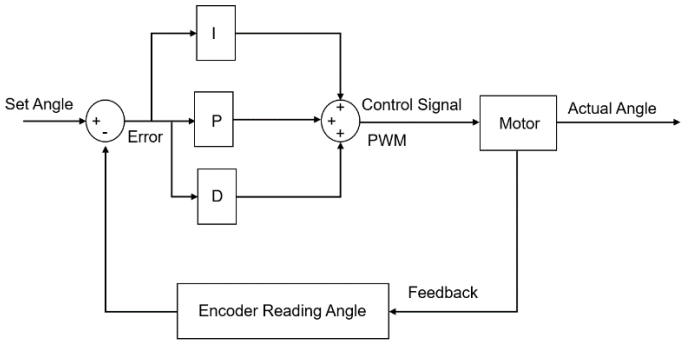


Figure 3. PID Logic of the Control

B. Computer Vision

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

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 PC through 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).

Figure 4. Andonstar AD407 digital camera (Andonstar)

2. Image Processing

To extract the useful image from 2D images, processing methods of Bilateral Filtering and Chroma Key Masking would be applied. First, we would use Bilateral Filtering shown in figure 5 to smoothen the image, reduce the noise, and preserve the edges.

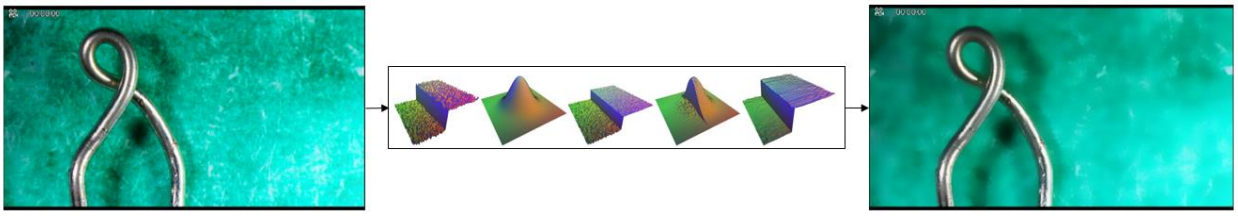


Figure 5. Original image and Bilateral filtered image

After the image is smoothened and noise is reduced, a Chroma key mask would be created to green curtain background to fully black like figure 6 shown. That would allow the silver coloring wire has a better contrast when comparing to the background, therefore the code could easily to read the wire pixels in the image frame.

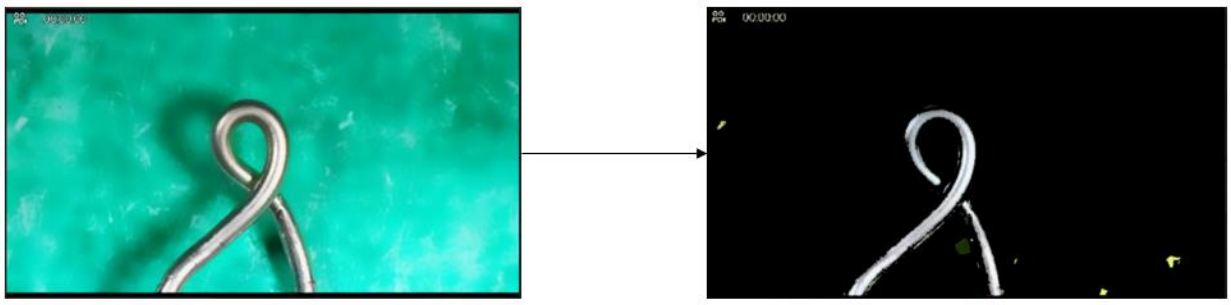


Figure 6. Bilateral filtered image and Chroma Key masked image

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 projected onto the image plane to obtain a predicted silhouette using a process called back-projection. The 3D shape is then adjusted to minimize the difference between the predicted silhouette and the observed silhouette. By iterating this process, the algorithm estimates the 3D shape of the object (Cheung, 2005) (Hartley, 2004).

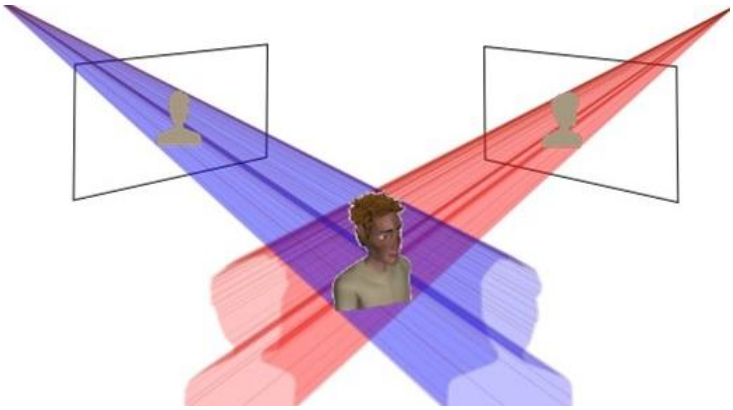


Figure 7. Example of Shape from Silhouette (SfS) algorithm (John, 2011)

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.

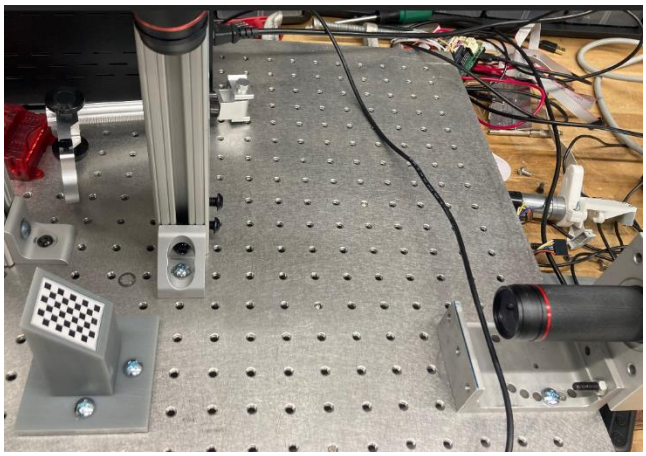


Figure 8. Dual-Camera setup with calibration checker boards

C. ROS Integration

1. Environment Setup and Installing Necessary Libraries

The installation process for ROS packages and libraries involves running commands on the terminal, which can be found on the ROS wiki (ROS.org, n.d.). The rosserial library provides a simple and efficient way to communicate between a microcontroller and a ROS environment, allowing for easy integration of sensors, actuators, and other hardware components.

To use the rosserial library with an Arduino board, the rosserial_arduino library must be installed on the Arduino IDE. This library provides the necessary tools for converting ROS messages into a format that can be understood by the microcontroller, enabling communication between the microcontroller and the ROS environment. With the library installed and the microcontroller connected to the computer, the development process can proceed to the next step of creating ROS nodes and configuring the system.

2. Developing ROS Nodes and Testing the System

The next step is to develop a ROS node that subscribes to motor commands and publishes the motor angle using the ROS publisher and subscriber API. The node should also have a callback function that receives the motor commands and sends them to the microcontroller, which then controls the motor using the appropriate signals. Another ROS node should be developed to subscribe to the motor angle and publish it to the image processing node. The

image processing node uses the motor angle to match the image data with the motor angle, which can be processed using libraries like OpenCV to extract object location and orientation.

To test the motor control and encoder reading, wire control signals are sent to the microcontroller, and the motor's movement is observed. Multimeters or oscilloscopes can be used to measure the signals and compare them with the expected values. The following is the ROS node logic: The motor control node receives the desired position from the user interface and sends the control signal to the motor driver. The encoder node reads the motor angle and publishes it to the image processing node. The image processing node processes the image data and publishes the object location and orientation to the user interface.

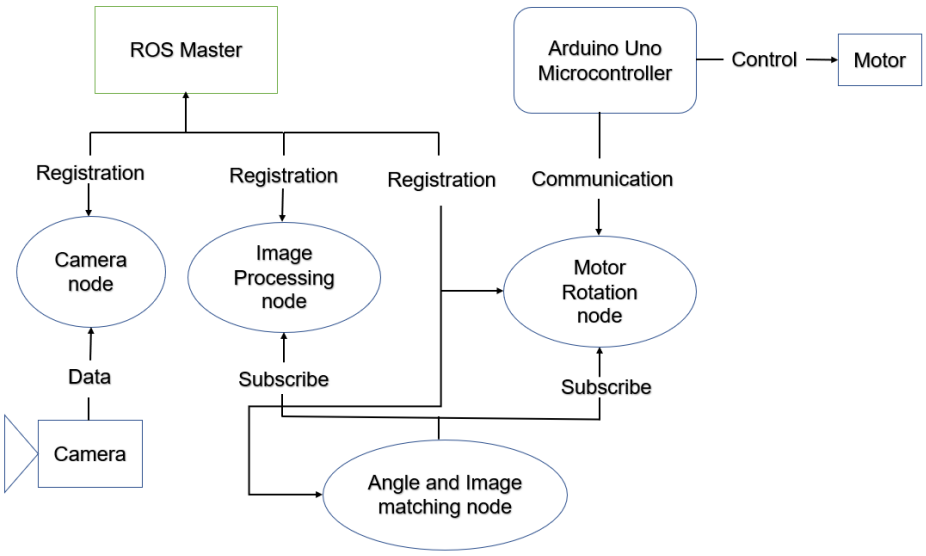


Figure 9. ROS System

D. Deep Learning

1. An Overview of Our Deep Learning Model

In this part, we will introduce the deep learning model we are going to use, which is a fully connected feedforward neural network with 2 hidden layers and also makes use of Adam optimizer and ReLU activation function. We will use this model to make a prediction of the wire manipulator configuration in 3D space.

A fully connected feedforward neural network with 2 hidden layers is a type of artificial neural network that consists of an input layer, two or more hidden layers, and an output layer. The input layer is responsible for accepting input data, while the output layer generates

predictions or outputs. The hidden layers are where the neural network processes and learns the underlying patterns in the data. Each neuron in the input layer is connected to every neuron in the first hidden layer, and each neuron in the first hidden layer is connected to every neuron in the second hidden layer, and so on until the output layer. This means that each neuron in a given layer is connected to every neuron in the next layer, hence the term "fully connected." The connections between neurons are represented by weights, which the neural network adjusts during training to improve its performance.

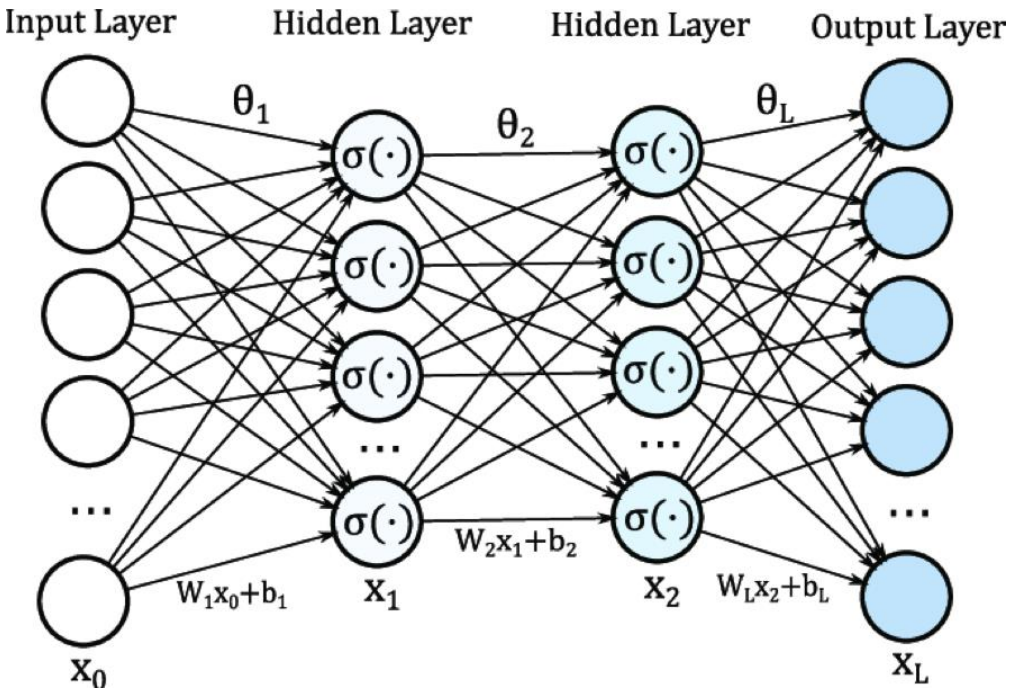


Figure 10. Fully Connected Feedforward Neural Network with 2 Hidden Layers (Tonello, Letizia, Righini, & Marcuzzi, 2019)

2. The Process of How Our Deep Learning Model Works

The process of making a prediction in a fully connected feedforward neural network with 2 hidden layers is as follows:

1. Input data is fed into the input layer, and each neuron in the input layer is activated by the corresponding feature in the input data.
2. The activated neurons in the input layer then feed their outputs forward to the first hidden layer.

3. Each neuron in the first hidden layer takes a weighted sum of the inputs it receives, adds a bias term, and applies an activation function to the result. The activation function determines whether the neuron should "fire" or not based on the weighted sum of inputs it receives, which we choose to use ReLU in our model.
4. The outputs of the neurons in the first hidden layer are then fed forward to the next hidden layer in the same way, and the process is repeated until the final hidden layer.
5. The outputs of the neurons in the final hidden layer are then fed forward to the output layer, where the network produces its final output which is a prediction of the wire manipulator configuration in 3D space.
6. The output layer's output is compared to the desired output, and the difference between the two is used to update the weights of the network using an optimization algorithm.
7. The updated weights are then used to make a new prediction on the next input, and the process repeats until the network has been trained to achieve its desired level of performance.

E. Transfer Learning

1. An 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. The idea 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.

2. The Process of How Transfer Learning Works

First, a pre-trained model is selected as the starting point. This would be our deep learning model that was trained on a large and diverse dataset for different geometry wires. The pre-trained model is then adapted to a smaller, domain-specific dataset which refers to a new specific wire, perhaps by adding or modifying some layers at the end of the model. These layers are specific to the new task (new specific wires) and are responsible for producing the output. After the new layers are added to the pre-trained model, then, we will train this model on the domain-specific dataset for the new task. The pre-trained layers are typically frozen during this

training process, which means that their weights are not updated. This allows the new layers to learn from the pre-trained layers without disrupting the knowledge they have already acquired.

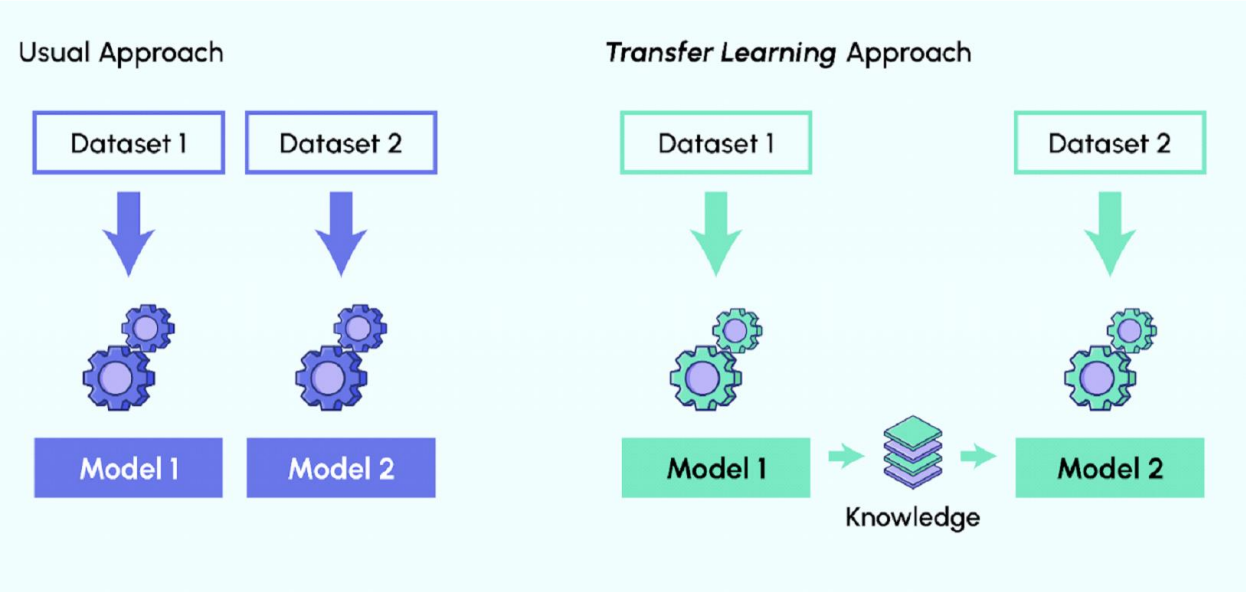


Figure 11. How Transfer Learning Works(Komendyak, 2022)

Dependencies

	Need	Status	Follow up	Buffer	Deadline
Nitinol Wires	Data Collection & Validation	Manufacturing	Fabricate by Dr. Usevitch	3/2/2023	3/8/2023
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	2/28/2023	3/2/2023
Motor Mounting	3D print auxiliary mounting parts	CAD Designing	3D print auxiliary mounting parts	2/28/2023	3/4/2023

Checker Board	CV calibration	Checker Board Printed, 3D print ongoing	3D print auxiliary moun ting parts	2/26/2023	3/2/2023
ROS	CV, Motor, Data Integration	Installed and Testing	Integrate separ ate system	3/2/2023	3/5/2023

Deliverables

MINIMAL (Expected delivery: 03/02/2023):

- Setup OpenCV tracking software and confirm track wire (Matlab or Python)
- Develop basic ROS system control code for single motor actuation - and document
- Code Wire control tests (motor positions) to move the system - Simple Code (ROS based)

EXPECTED (Expected delivery: 03/31/2023):

- 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 (Expected delivery: 04/14/2023):

- Test and compare various several DL Nets for control of the system
- Repeat process with various other wire sizes
- Write IEEE format manuscript

Test plan

Motor Control Test

1. Encoder Reading Feedback and PID Angle Control Test

To test the encoder reading feedback and PID angle control, wire control signals are sent to the microcontroller, and the motor's movement is observed. Multimeters or oscilloscopes can be used to measure the signals and compare them with the expected values. The feedback signal

from the encoder should be stable and accurate, allowing the PID controller to adjust the motor's position accordingly. The PID controller's gains must be tuned to ensure stable and accurate control of the motor. The tuning process involves adjusting the Proportional, Integral, and Derivative gains until the desired performance is achieved.

2. ROS System Optimizing and Conducting Tests

To optimize the system performance, parameters such as the PID gains and the encoder filter settings are tuned to minimize the position error and improve the response time. Documentation of the code and the system design is essential. UML diagrams and sequence diagrams can be used to illustrate the system architecture and the interaction between the nodes. Finally, the system performance is validated by conducting a series of tests and experiments that simulate real-world scenarios. Metrics such as the root mean square error (RMSE), the response time, and the failure rate can be used to assess the system performance and compare it with the design specifications. The system's performance should be evaluated under various conditions to ensure its reliability and robustness.

Computer Vision Test

This test is designed to generate accurate wire 3D configuration for data collection. The Nitinol wire manipulator would be actuated to different positions and orientations within the camera's field of view, and multiple images would be captured for each position. The images would be processed using the Bilateral Filtering and Chroma Key Masking techniques, and the resulting wire 2D images would be used as input to the SfS algorithm.

Additionally, the system would be tested under different lighting conditions and background environments to evaluate its robustness. Data from these tests would be used to fine-tune the SfS algorithm and improve the system's performance. The performance of the system would be evaluated based on the accuracy and robustness of the 3D shape estimates, as well as the processing speed and computational efficiency of the overall system. The goal of this computer vision system is to provide real-time feedback of the wire manipulator's position and orientation, which is critical for safe and effective surgical procedures.

Deep Learning Model Validation

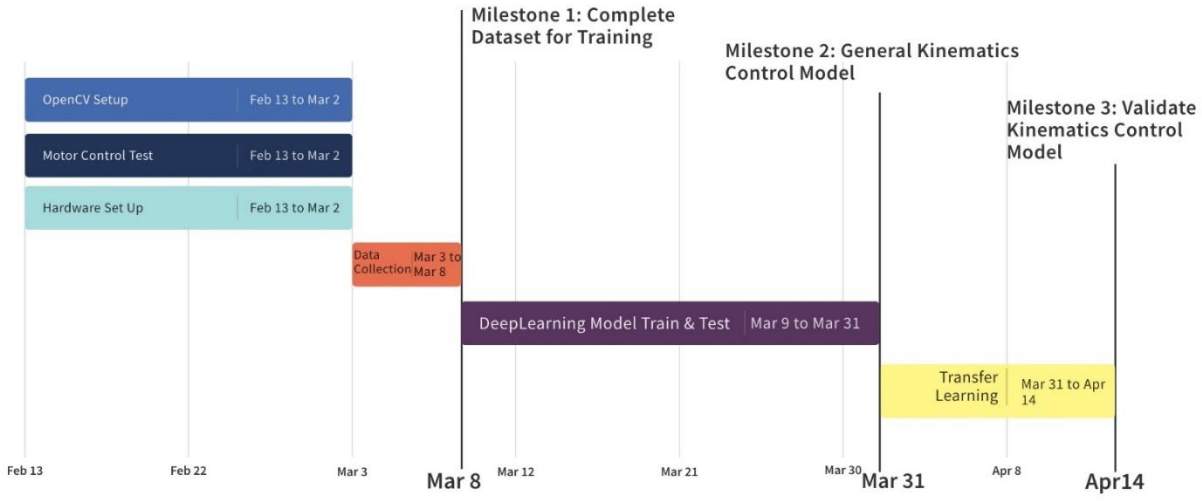
To test a fully connected feedforward neural network with 2 hidden layers involves loading the trained model, preparing the test data, running the test data through the model, and evaluating the model's performance. First, we will load the trained neural network model. Then, prepare the test data that we want to use to evaluate the performance of the neural network. This data should be separate from the training data used to train the model and should be representative of the real-world data that the model will be applied to. Next, run the test data through the model and generate predictions for each input. Finally, evaluate the performance of the model, which involves calculating various performance metrics, such as accuracy, precision, recall, and F1 score, and comparing them to a predefined threshold or benchmark.

Depending on the performance of the model, we may need to repeat the process of tweaking the model architecture or hyperparameters and retesting until the desired level of performance is achieved.

Transfer Learning Validation

To test a transfer learning model involves similar steps to test a fully connected feedforward neural network which are just mentioned above, but with some additional considerations, for example, we need to load the pretrained model and pretrained weights as the base for our transfer learning model and we may need to freeze the pre-trained layers so that they do not get updated during training.

Timeline



Milestone

Milestone	Subtasks	Time Period
Milestone 1	OpenCV Setup: create environment, image capturing from two cameras, processing image and shape carving, and 3D data output	Ongoing - 03/02/2023
	Motor Control Test: motor angle control with PID, ROS system setup and encoder data collection and image data matching	Ongoing - 03/02/2023
	Hardware Setup: nitinol wire manufacturing, motor mounting, and camera mounting and CV auxiliary parts	Ongoing - 03/02/2023
	Data Collection: create datasets for Deep Learning and transfer learning method	03/03/2023 - 03/08/2023
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	03/09/2023 - 03/31/2023
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	03/31/2023 - 04/14/2023

Management Plan

1. **Meetings:**

Weekly mentor meetings will be held every Friday at 2 PM with Prof. Iordachita's Lab. In addition, group meetings will be held twice a week. Daily communication and updates will be made through Teams.

2. **Platforms:**

Various platforms will be utilized for communication, documentation, file-sharing, and report writing.

- Communication: Teams (chat) with Prof. Iordachita's Lab will be used, along with email and Zoom meetings.
- Platform Construction and Test Lab: Prof. Iordachita's Lab

- Code: Code files will be maintained on GitHub, on a private repository.
- Data & Filesharing: The project wiki and Teams (files) will be utilized for sharing data and files. The functionality of file in Teams is secure and Teams is an encrypted platform.
- Report Writing: Teams (files) will be used for basic report writing, and LaTeX (Overleaf) will be used for the final manuscript preparation.

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 2 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.

Reference

[1] Andonstar AD407 HDMI Digital USB microscope. Andonstar. (n.d.). Retrieved February 23, 2023, from https://andonstar.com/?product=andonstar-ad407-hdmi-digital-usb-microscope

[2] Cheung, G.K., Baker, S., & Kanade, T. (2005). Shape-From-Silhouette Across Time Part II: Applications to Human Modeling and Markerless Motion Tracking. *International Journal of Computer Vision, 63*, 225-245.

[3] Grassmann, R.M., Modes, V., & Burgner-Kahrs, J. (2018). 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)*, 5125-5132.

[4] Hartley, R., & Zisserman, A. (2004). Multiple view geometry in computer vision (2nd ed.). Cambridge University Press. Chapter 12 discusses the SfS algorithm in the context of reconstructing 3D shapes from multiple views.

[5] John, C. (2011). Volumetric Hand Reconstruction and Tracking to Support Non-Verbal Communication in Collaborative Virtual Environments.

[6] Komendyak, M. (2022, April 11). How to Choose the Best Pre-Trained Model for Your Convolutional Neural Network. Data Science Blog. https://data-science-blog.com/blog/2022/04/11/how-to-choose-the-best-pre-trained-model-for-your-convolutional-neural-network/.

[7] Runciman, M., Darzi, A., & Mylonas, G. P. (2019). Soft robotics in minimally invasive surgery.

[8] *Soft Robotics*, 6(4), 423–443. https://doi.org/10.1089/soro.2018.0136.

[9] ROS.org. (n.d.). rosserial_arduino/Tutorials. Retrieved from http://wiki.ros.org/rosserial_arduino/Tutorials.

[10] Tonello, A. M., Letizia, N. A., Righini, D., & Marcuzzi, F. (2019). Machine Learning Tips and Tricks for Power Line Communications. IEEE Access, 7, 1-1. doi:10.1109/ACCESS.2019.2923321.

[11] Usevitch, D. (2023, January 23). *16 - Deep Learning Kinematics for Continnum Wire Manipulator* [CIS II Lecture, Johns Hopkins University].

[12] Webster, R.J., Romano, J.M., & Cowan, N.J. (2009). Mechanics of Precurved-Tube Continuum Robots. *IEEE Transactions on Robotics, 25*, 67-78.