MRI Compatible Position Sensing for Soft Continuum Surgical Robots

MECH5845 Professional Project
Author Name: Neeraj Krishnan Nambudiri
SID: 201887240
Supervisor Name: Dr. James Avery
Examiner Name: Dr. George Jackson-Mills
Date of Submission: 14/08/2025
MRI Compatible Position Sensing for Soft Continuum Surgical Robots

SCHOOL OF MECHANICAL ENGINEERING

UNIVERSITY OF LEEDS

MECH5845M Professional Project

TITLE OF PROJECT

MRI Compatible Position Sensing for Soft Continuum Surgical Robots

PRESENTED BY

Neeraj Krishnan Nambudiri

If The Project Is Industrially Linked Tick This Box And Provide Details Below ☐

Company Name and Address:

INDUSTRIAL MENTOR:

This project report presents my own work and does not contain any unacknowledged work from any other sources.

Signed Date: 14/08/2025

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Dr. James Avery, for his continued guidance and support through the duration of this project. His advice has been invaluable throughout the course and has made it possible for me to achieve the project results.

I would also like to thank Sam Wilson for his guidance and cooperation in using the lab space and equipment to conduct the project.

Table of Contents

Abstract

Minimally Invasive Surgery (MIS) increasingly employs soft continuum robots for their dexterity, reduced-traumatic interaction, and ability to navigate complex anatomical pathways. In procedures guided by Magnetic Resonance Imaging (MRI), sensing the robot's position is critical for accurate manipulation and safety. However, many established tracking technologies face challenges of MRI incompatibility, size constraints, or high cost.

This project presents the design and evaluation of an MRI-compatible, low-cost sensing system for real-time curvature and tip position estimation in a cable-driven soft continuum robot. The approach uses non-metallic tendons routed through PTFE channels, with one end fixed at the robot tip and the other marked for visual detection outside the MRI environment. A computer vision algorithm processes video frames to track cable displacement, which is correlated to bending angle through calibration.

The development process included iterative refinement of image processing pipelines from static to dynamic tracking, integration with a single-tendon cable-driven actuation system based on the Constant Curvature Model (CCM), and calibration using physical bending trials. Validation was conducted by comparing sensor-derived angles with both manually measured angles and CCM predictions.

The system demonstrated reliable performance for moderate to large bends, meeting real-time frame rate targets and maintaining MRI safety by keeping all sensing electronics outside the scanner field. The results indicate that optical cable-displacement tracking is a viable and scalable solution for MRI-guided continuum robot sensing, with potential for integration into closed-loop surgical control systems.

Chapter 1

Introduction

1.1 Background

Modern-day healthcare has been revolutionised by the introduction of Minimally invasive surgery (MIS). It has enabled precision-driven procedures, aimed at reducing patient trauma. In the applicable surgical domains, Robotic MIS has started replacing open surgeries, due to its significant benefits such as greater patient safety, faster recovery, lower post-operative problems, and reduced in-patient duration [1][2]. However, long and rigid instruments are used in traditional robot-assisted MIS, which can have restrictions due to the fulcrum effect and some collateral damage to surrounding tissue. This has resulted in prominent research developing soft robotics for MIS, focused on combining rigid robotic control with flexibility and safety of soft materials [2]. Figure 1.1 showcases the evolution of MIS and introduction of flexible instruments in the field.

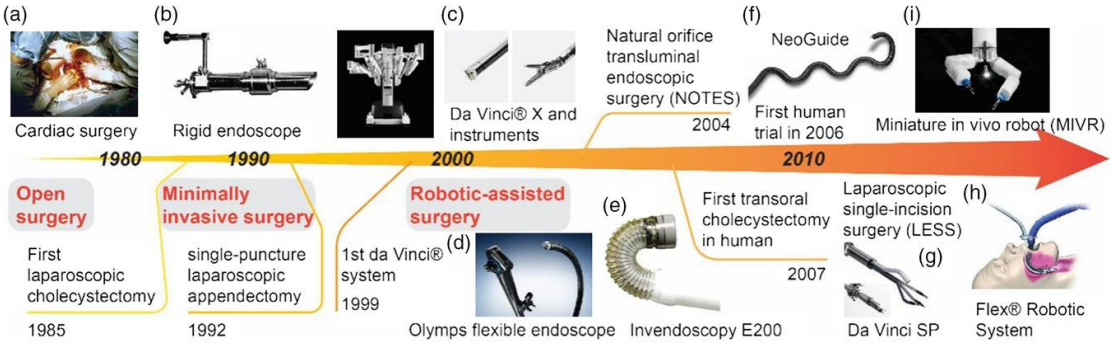


Figure 1.1: Development of MIS with typical surgical tools and products. Captions for individual images included within [3].

Simultaneously, the intervention of advanced imaging techniques, such as Computed Tomography (CT), Ultrasound, X-ray, and Magnetic Resonance Imaging (MRI), with MIS is also being developed. Among these, only MRI provides the data without any ionising radiation. Hence, comprehensive study for integrating MRI during MIS procedures is ongoing [4]. However, it poses considerable challenges due to the presence of strong magnetic field, thus prohibiting the use of conventional metallic or electrical sensing mechanisms for robotic control. Hence, development of alternative proprioceptive sensing solutions is paramount. The most widely investigated sensing technology is optical-based sensing, such as Fibre-Bragg Grating (FBG), fibre-scope, and Fabry-Perot interferometer (FPI) [5]. They have exhibited high accuracy but also have relatively higher stiffness which reduces the flexibility of the soft robotic

manipulators and are very expensive, which limits their integration in single-use soft surgical robots.

Accordingly, this project involves the design and evaluation of an MRI-compatible sensing system tailored for the position tracking and motion control of a soft continuum robot. It utilises a low-cost, flexible sensing system based on cable transmission. The displacement of this cable, induced by robotic movement, is captured using a camera setup and employing Computer Vision (CV) algorithm. It is correlated to the bending angles for accurate position tracking, and its performance is evaluated to analyse potential of offering a non-intrusive method for accurate motion tracking without interference with MRI functionality. Figure 1.2 shows the reference design of a soft-continuum robot.

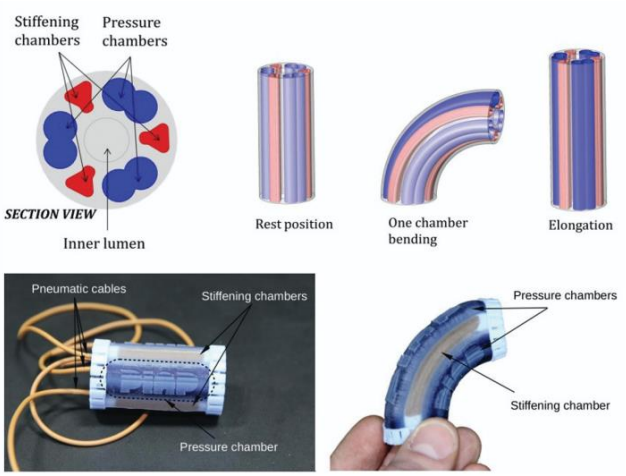


Figure 1.2: Example of a Soft Robot [6].

1.1 Project Aim

This project aims to develop and evaluate a size and stiffness tuneable, metal-free, MRI-compatible sensing system for real-time position tracking of a soft continuum surgical robot used in minimally invasive procedures.

1.2 Project Objectives

The key objectives to achieve the project aim are as follows:

1. Review relevant literature to understand the benefits and limitations of existing technologies and finalise the methodology.
2. Design and implement the sensing system utilising Polytetrafluoroethylene (PTFE) tubing and cable-displacement mechanism for motion tracking.
3. Integrate the sensor with the cable-driven continuum robot after preliminary testing.

4. Develop a CV algorithm for detecting and quantifying edge displacement and establish a robust correlation between detected cable-displacement and robotic motion parameters.
5. Calibrate and perform validation to ensure system reliability, repeatability of results, and account for hysteresis errors.
6. Assess accuracy, latency, speed, and precision of the proposed sensing system.
7. Compare results with established sensing technologies such as Fiber-Bragg Grating (FBG) and EM-tracker.
8. Target accuracy within 0.2 mm, delay of 0.1 seconds and frame rate of 30 fps.

1.3 Project Report Layout

- Chapter 1 – Introduction: Establishes the background of the project and specifies the corresponding aims and objectives.
- Chapter 2 – Literature Review: Provides a critical review of relevant literature for all aspects of the project, from MRI-guided MIS to Fabrication methods.
- Chapters 3 to 5 – Project Development: Describes in detail the methodologies of all the aspects – CV system, Fabrication, and Actuation Control.
- Chapter 6 – Validation Experiment: Describes the experimental setup and methodology for evaluation of the sensing system, along with the results and analysis of its performance.
- Chapter 7 – Conclusion: Summarises the key achievements, results, discusses significance and limitations of the findings, and outlines potential pathways for future work.

Chapter 2
Literature Review

2.1 Introduction

The convergence of soft robotics and MRI-guided intervention in MIS procedures have amplified the need for developing robust, accurate, and reliable soft sensing technologies and unveils a new, critical frontier in the field of medical robotics research. They are capable of navigating complex anatomical structures, providing additional dexterity and accessibility [7]. The robotic systems also elevate the skills of a surgeon, while delivering higher precision and accuracy [8]. At the same time, it also presents interdisciplinary challenges, due to the multiple constraints – clinical and technical. To overcome these challenges, adaptation and re-imagining of the feedback mechanisms for robot position is required. This literature review will explore the current benchmarks in the domain. Furthermore, validation of the proposed sensing system requires the development of a working prototype of the soft continuum robot, which is also discussed briefly.

2.2 MRI Guidance in Minimally Invasive Surgery

MRI stands as the benchmark for soft tissue visualisation and hence adds considerable value when used as guiding intervention for MIS procedures. It offers unparalleled contrast resolution, continuous and real-time monitoring without any cumulative radiation exposure – critical for patient safety. But it also necessitates specific surgical instrumentation, based on the constraints exemplified due to the presence of strong magnetic fields in these environments.

2.2.1 Regulations for MRI Compatibility

The electromagnetic compatibility of components and devices used in a surgical procedure are critical in an MRI environment. Based on their safety in MRI environment, the American Society for Testing and Materials (ASTM) F2503 [9] categorises the systems into three: *MR Safe*, *MR Conditional*, and *MR Unsafe*. MR Safe devices are neither affected nor influence the magnetic field in an MRI environment. These instruments are made of non-metallic, non-conductive, and non-magnetic materials. MR Conditional stipulates instruments which are demonstrated to be safe in an MRI environment, but only under specific conditions. MR Unsafe devices are classified as hazardous for functioning in an MRI environment [9].

Based on this classification by ASTM, this project aims at developing the sensing system which would be MR Safe, thus highlighting its application in all MRI environments, without accommodating any specific criteria. This would be ensured by selecting metal-free and non-conductive materials in the region of the surgical procedure.

2.2.2 Technical Challenges for MRI-guided MIS

The strong magnetic field, radio frequency pulses, and gradient coils of the MRI machine interfere with the operation of traditional robotic systems. At the same time, the metallic components in the robots can distort the quality of the MRI data [4] [10], thus reducing the efficacy of the procedure. This inherent incompatibility lays the foundation of most technical challenges associated with MRI-guided MIS procedures. In developing solutions for these, MR Safe or MR Conditional robotic systems have been designed, utilising non-magnetic and low-susceptible materials [4].

Among these design research, optical and EM based sensing technologies were the most widely explored. However, other sensing modalities require further investigation.

Parallelly, a lot of development has also occurred in design of soft robots. The primary advantages, when compared to rigid systems, are crystal-clear – offering higher manoeuvrability while decreasing possibility of patient trauma during surgical procedures [11]. However, these properties also lend to greater complications in integrating sensor modalities – since conventional sensors are mostly made of rigid components [12] [4].

This highlights the critical need of developing a reliable, MRI-compatible sensing technology which is capable of providing real-time position sensing to propel further applications of MRI-guided surgery.

2.3 Contemporary Sensing Technologies

2.3.1 Optical Sensing Methods

Through the growing research in MRI-compatible sensing modalities, optical systems have emerged as the pre-eminent solution, capitalising on their fundamental immunity to electromagnetic interference. Figure 2.1 showcases the general array of sensing technologies.

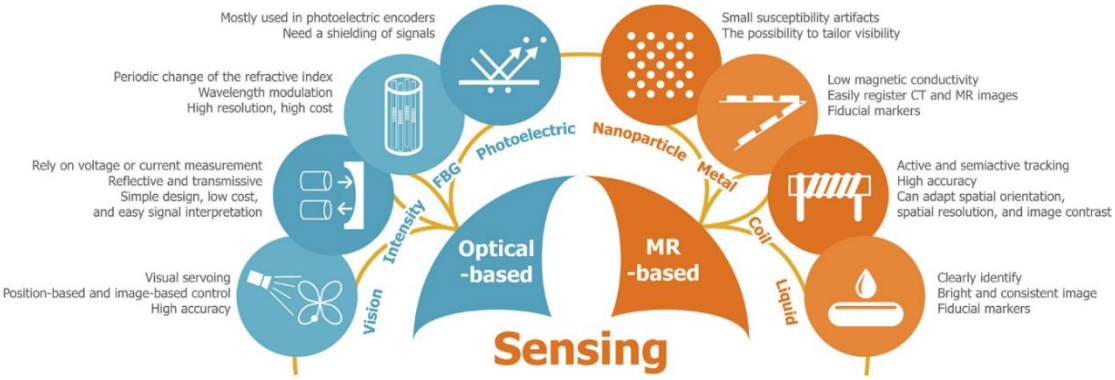


Figure 3.1: Contemporary MRI conditional/safe sensing methods [4].

For instance, Su et al. [13] demonstrated the development of an MRI-compatible soft robotic system that would be used in neurosurgical procedures and utilized fibre optic sensors to determine the position and force applied at the robot tip. The research by Fang et al. [14] indicates the development of a soft robotic gripper and uses an optical waveguide to transmit laser in transoral laser ablation. A fiberscope is integrated within the soft actuator allows the position-tracking by following the laser and provides closed-loop control during operation. The design in shown in Figure 3.2.

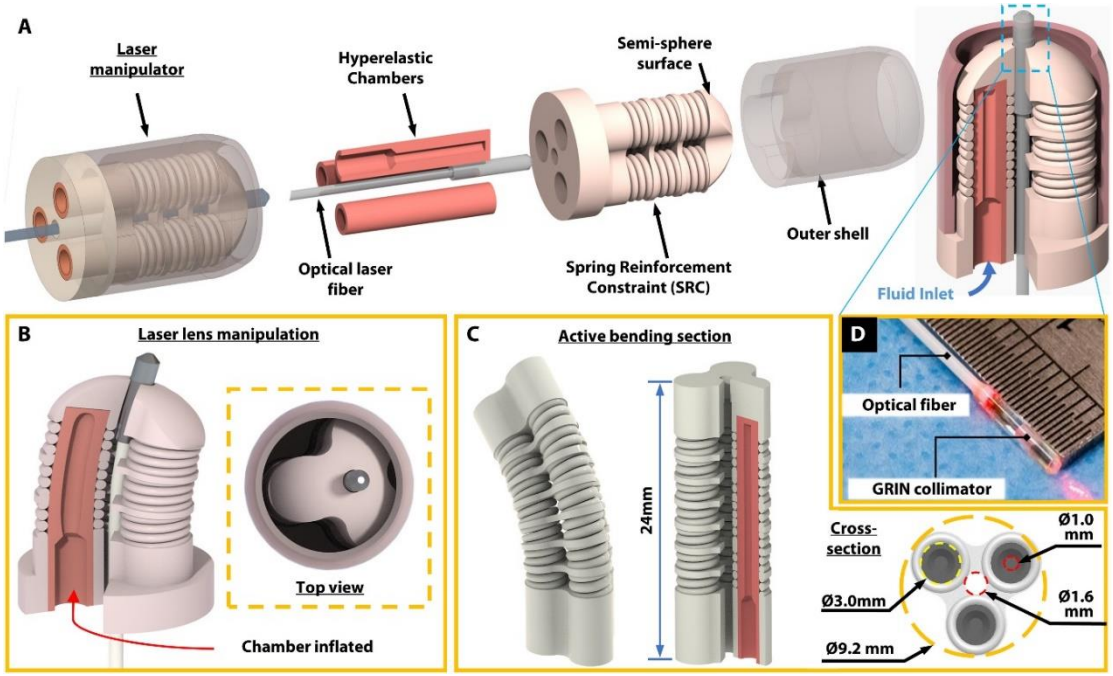


Figure 3.2: Design of Soft Robotic Laser Steering System [14].

It has shown very good accuracy (approx. 0.2 mm) during various in-vitro and animal trials. However, due to the need of specialised fiberscope for the position tracking, the compactness is affected, reducing flexibility when compared to other soft surgical tools. This constraint can be crucial in evaluating the viability of the technology for

MRI-guided surgeries, since the space within the MRI bore is a critical limiting factor for design requirements. The reduced miniaturising property due to the optical fiberscope can pose challenges in surgical applications for certain anatomical regions.

Fibre-Bragg Grating (FBG) technology is currently considered as the gold-standard for MRI-compatible sensing, due to its highly accurate and reliable performance. The principle of the sensor involves periodic refractive index variations within the fibre core that reflect specific wavelengths proportional to local mechanical strain, visualised in Figure 3.3. This distributed sensing modality allows accurate quantification of the shape of the robot and tip-position [13].
While the core technical performance is very good for FBG sensors, they often demonstrate non-linear and hysteretic behaviour. Furthermore, the requirement of highly sophisticated equipment for wavelength measurements and spectral analysis increases the cost of these systems drastically, making it difficult to integrate with one-time use soft surgical tools currently in development.

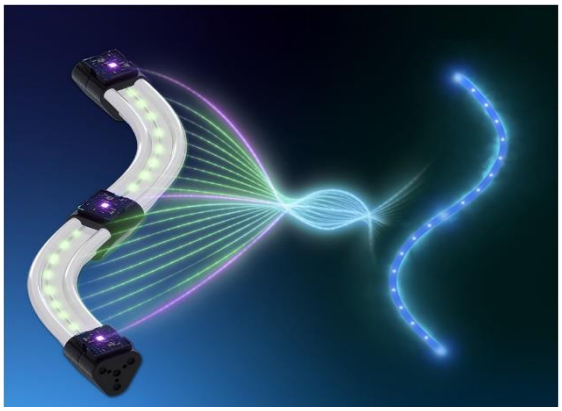


Figure 3.3: FBG Sensing principle visualisation [15].

Fabry-Perot Interferometric (FPI) sensors utilise phase-modulation principle to achieve high sensitivity in compact soft surgical tools. These have shown exceptional performance in force-sensing applications [16]. But, similar to FBG sensors, these are also limited by the need of extremely expensive equipment for completing the sensing framework.

2.3.2 MR-Based Sensing and Other Modalities

The utilisation of MRI signals directly for sensing, provide an elegant alternative by leveraging the in-use imaging equipment.
There are Active Tracking Systems, which have shown impressive performance capabilities. They employ active coil to amplify the existing electromagnetic field, to

produce high contrast images allowing real-time tracking [17]. But, the presence of another magnetic field within the environment has shown potential in degradation of the image quality, which can force trade-off between tracking accuracy and imaging quality.
Passive Marker Techniques addresses these quality concerns by employing MR-conditional materials [17]. But these require specialised hardware and setup, careful optimisation of size, shape, and material properties to minimise the degradation while maintaining adequate contrast for precise tracking.

EM Tracker employs a transmitter for a separate electromagnetic field, which is sensed to locate the position of the robot [12]. But this requires specialised hardware and minute calibration to avoid interference with the MRI magnetic field, to be termed MR-Safe. The setup is shown in Figure 3.4.

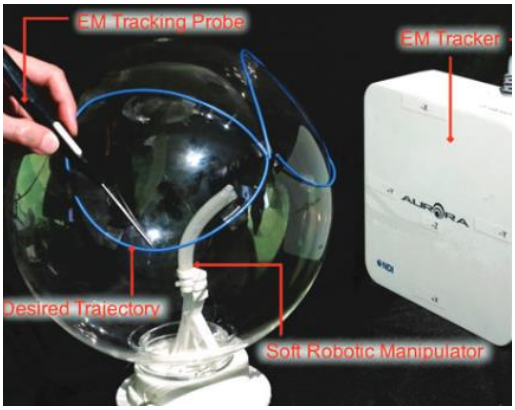


Figure 3.4: EM Tracker [18].

There are other sensing methodologies which are emerging with promising results. Lin et al. [19] demonstrates a proprioceptive sensing method for an elastomeric finger. It employs a combination of fibre-based displacement sensors and microfluidic pressure sensors, as shown in Figure 3.5. The fibre-based sensors measure bend, twist, and elongation, while the microfluidic sensors measure overall and local pressures. These are passive sensors, transmitting information optically to a camera for recording [19]. This approach reduces the computational resource demand while preserving higher accuracy and flexibility.

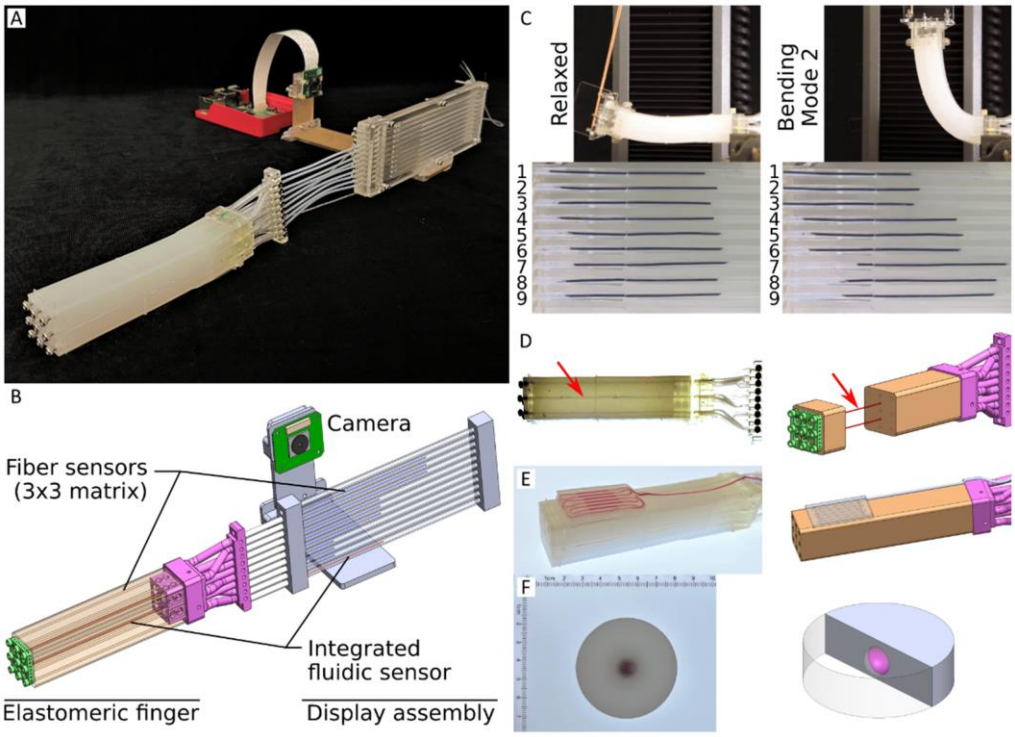


Figure 3.5: Fibre-based and Microfluid Soft Sensors. A) Elastomeric finger with the sensors. B) Illustration of sensors through the elastomeric finger. C) Fiber-based sensor. D) Microfluid sensor. E) Surface-mount pressure sensor. F) Colour cell pressure sensor [19].

The above principle opens-up the possibility of utilising fibre-displacement sensing principle as a promising alternative to existing sensing systems which were compared earlier. The limited scalability and cost constraints of optical systems are improved with this principle, while having no interference with the MRI magnetic field like the MR-based sensors.

2.4 Kinematic Modelling for Soft Robot Actuation

Soft continuum robotic systems distinguish themselves from traditional rigid-link robots by operating without conventional revolute or prismatic joints. Instead, their movement is based on continuous deformations in shape, often inspired by biological actuators. These robots are typically actuated by hydraulic, cable-driven or pneumatic systems.

- Pneumatic Actuators: These are the most common actuation method for MRI-compatible robots. They operate by inflating or deflating hollow chambers within the soft material structure, leading to bending or elongation [20].

- Hydraulic Actuators: These systems use liquid flow to actuate the robot, offering advantages like intrinsic MR safety and the ability to miniaturise the robot's design by placing larger actuator systems outside the MRI room [10].
- Tendon-Driven (Cable-Driven) Systems: These involve pulling cables guided within channels along the manipulator fixed at its tip, enabling bending [10].

Within the framework of the MRI-compatible soft robot, hydraulic actuation would be the best method, since it need not employ any metallic or magnetic materials. But they are difficult to fabricate and integrate mechanically. The primary scope of this project is to develop the sensing system and validate its performance. The actuation is only incidental for the prototype development – since the sensing system is completely independent of the actuation methods. To facilitate the project with those considerations, cable-driven actuation would be employed for this project, with the analysis that if the sensing system is validated, it can be later integrated with MRI-safe actuation methods like hydraulic actuation for future in-vitro trials.

2.4.1 Kinematic Modelling

The inherent compliance and continuous deformations of soft continuum robots, coupled with their theoretically infinite degrees of freedom, make their modelling and control notoriously difficult.

Constant Curvature Model (CCM): It is a widely applied simplified kinematic model for continuum robots. It approximates the robot's curvature remains constant. It is computationally efficient and has been widely used for both single-section and multi-section manipulators [21] [22].

- Errors and Approximations: Despite its utility, the CC model is an approximation and does not completely match all characteristics of continuum robots, particularly in accurately reflecting complex deformations or the influence of external forces.

Variable Curvature Models To achieve greater accuracy, especially when considering external forces and positional constraints, variable curvature modelling methods have been developed, like Lagrangian polynomial series. These models move beyond the constant arc approximation to describe more complex deformations [21]. However, they add much higher computational difficulties, which would prove beyond the scope of this project. Hence, CCM would be employed as the basis for actuation during validation trials, accounting for the inherent errors and approximations by correction factors and repeated trials for empirical data.

Chapter 3

Computer Vision Sensing System

3.1 Introduction

In the context of MRI-compatible soft continuum robots, position sensing must be accurate, minimally intrusive, and non-disruptive to the robot's mechanical compliance. Computer Vision (CV) offers an alternative method of tracking deformation through visual markers, while being remote from the actuating system [23].

This work develops a CV-based sensing pipeline tailored to a cable–PTFE tube mechanism integrated into the robot. The concept involves tracking pre-marked cables at the distal ends of PTFE tubes, far outside the MRI environment to a static camera, shown in Figure 3.1. Cable displacement, induced by robot bending, is captured using a camera. CV is applied to compute this displacement, and it is correlated to the bending angles for calibration.

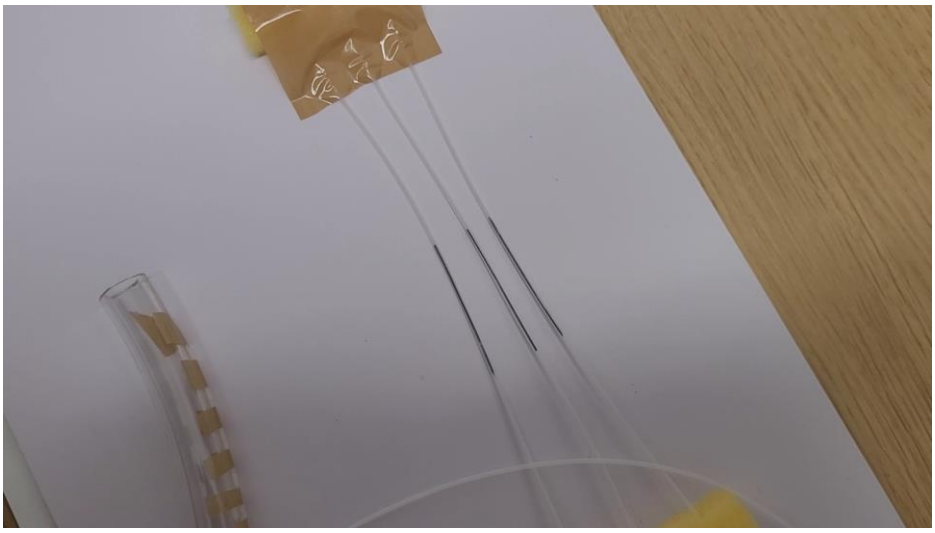


Figure 3.1: Basic setup of the marked cables used for position sensing

The methodology was implemented entirely in Python with OpenCV and progressed through an iterative development cycle:

1. Baseline Algorithm – Static Image Processing
2. Extension to video processing from camera using temporal tracking
3. Calibration - Correlating of displacement to bending angle by performing standard angular movements manually

4. Evaluation of model accuracy and limitations

3.2 Static Image Processing – Algorithm Prototyping

Initial trials focused on static images of two high-contrast horizontal lines (representing the marked cables) on a white background. This controlled environment allowed methodical testing of detection strategies before extending to physical cable image captures.

3.2.1 Methodology

The static image pipeline (refer to Appendix 1 for code) consisted of:

- **Preprocessing:** Conversion to grayscale to reduce data dimensionality and simplify intensity-based operations.
- **Contrast Enhancement**: Application of Contrast Limited Adaptive Histogram Equalization (CLAHE) to amplify line (representing the marked cables) intensity while avoiding oversaturation of bright regions, shown in Figure 3.2.

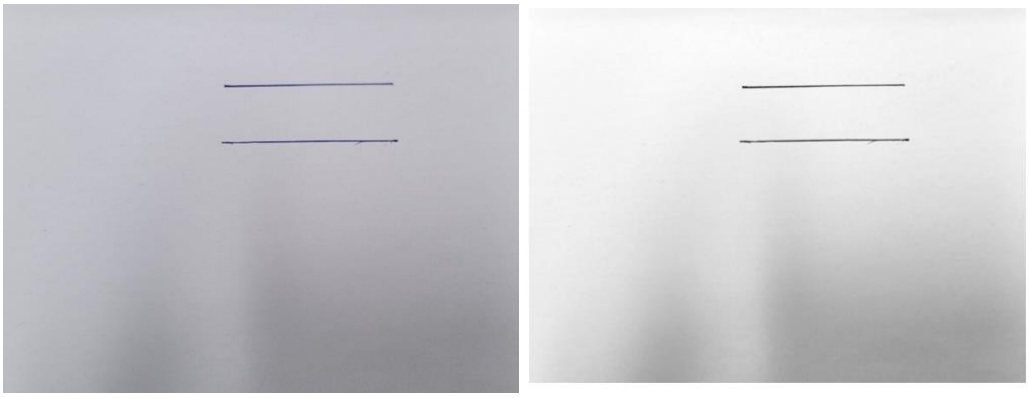


Figure 3.2: Image Processing Flow (a) Original Image. (b) Contrast Enhancement.

- **Edge Detection:** Canny edge detection with tuneable thresholds. Modified from fixed thresholds early on to adaptive thresholds for better performance.
- **Line Detection:** Probabilistic Hough Transform to parameterise detected edges into discrete line segments, with filtering for near-horizontality – since the marked cables would always be horizontal based on the controlled environment of image capture in the eventual sensing system, shown in Figure 3.3 (a).
- **Post-Processing:** Morphological closing to connect fragmented edges and merging logic to combine segments belonging to the same line, shown in Figure 3.3 (b).

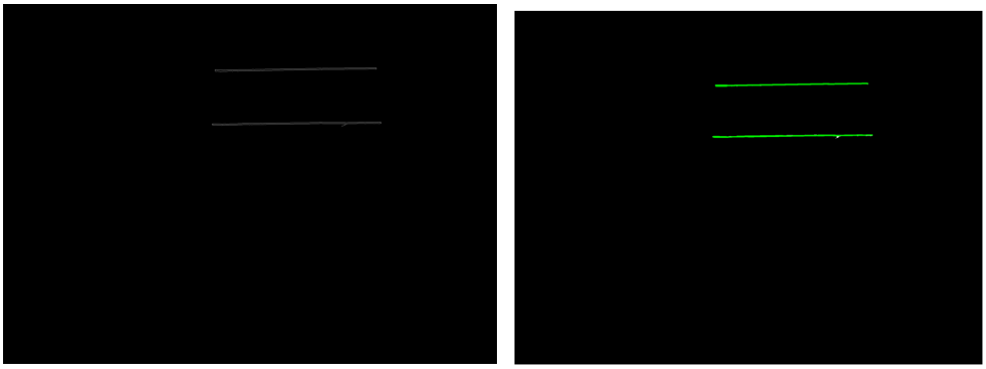


Figure 3.3: Image Processing Flow - 2 (a) Line Detection. (b) Post-processing.

3.2.1 Observations and Refinement

Initial outputs successfully detected the target lines but presented two consistent challenges:

- **Line Fragmentation:** Multiple short segments per cable were detected. Tuning the thresholds for maximum distances between possible lines and applying morphological closing improved continuity.
- **Duplicate Detection from Line Thickness:** Single drawn lines were detected as multiple ones due to their thickness. This was mitigated through proximity-based merging, discarding duplicates based on the y-coordinates.

By the final static iteration, the algorithm consistently identified the two target lines with minimal false positives. Remaining duplicates were tolerated at this stage, with merging deferred to the video processing stage.

3.3 Video Processing – Dynamic Line Tracking

This algorithm was scaled for dynamic tracking to enable video processing and real-time coordinate extraction under movement. The camera captures video at 30 fps, which was included as a factor in the pipeline by handling frame-to-frame variability and motion blur.

3.3.1 Methodology

The dynamic pipeline (refer to Appendix 2 for code) incorporated several extensions to the static image processing logic:

1. **Frame Acquisition and Storage:** Videos were processed frame-by-frame, with intermediate outputs (grayscale, contrast-enhanced, edges, lines) for debugging and parameter tuning. Figure 3.4 shows a frame from the intermediate output of first trial, indicating false detection of edges of the

PTFE tubes, highlighting need for higher contrast and pre-processing thresholds.

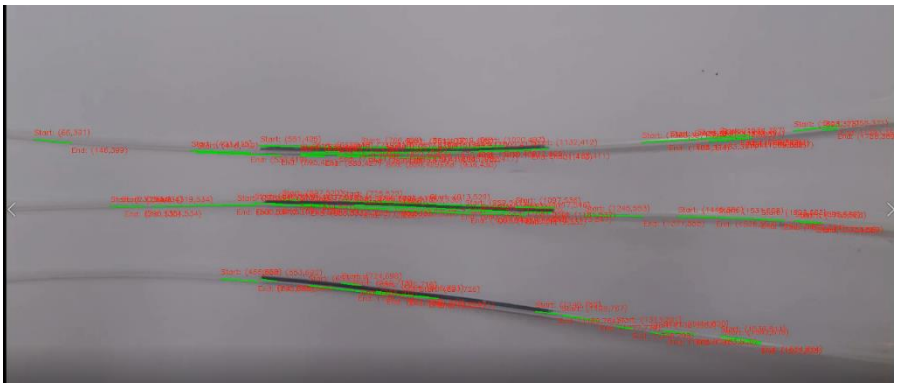


Figure 3.4: Output frame of 1st trial – multiple lines detected and frame edges

2. **Region of Interest (ROI) Masking:** With the static camera and assurance of controlled environment, the focus of processing and computation was limited to the central region containing the cables, thus reducing false positives from frame edges.

3. **Persistence:** If a band failed to detect a line in a given frame, the previous frame's coordinates were reused (persistence), preventing data gaps. Figure 3.5 shows a frame with missing segments.

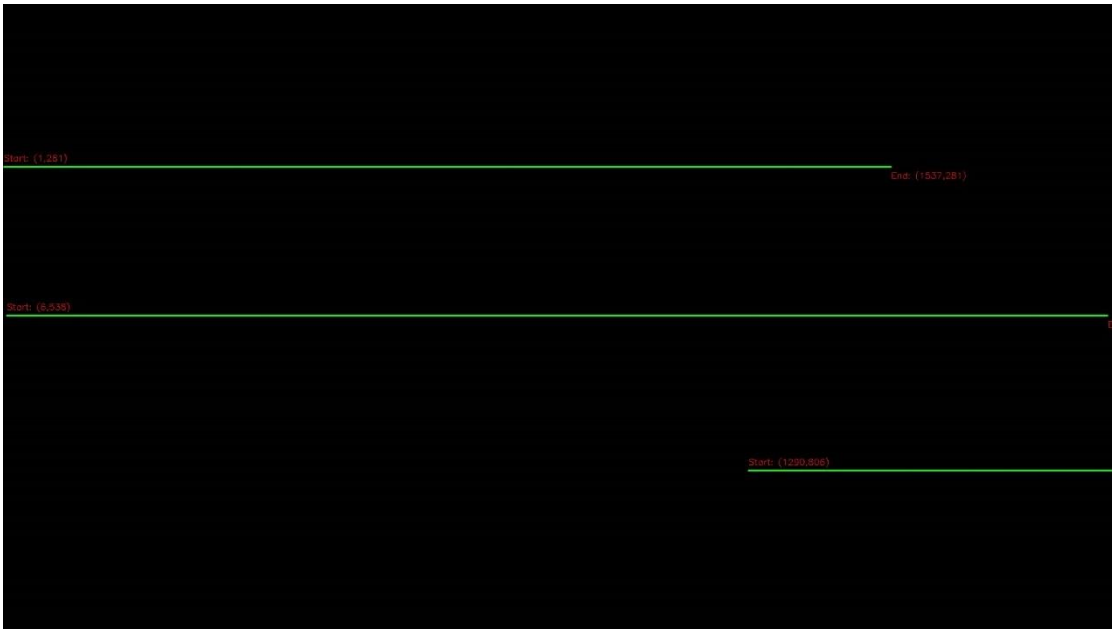


Figure 3.5: Frame of output video with missing segments

4. **Gradual Motion Modelling:** When the marked cable moves, at certain points sudden jumps in the coordinates were observed due to latency. The

datapoints between frames were interpolated linearly to emulate gradual bending, matching the mechanical actuation profile.

Flowchart for final logic is shown in Figure 3.6.

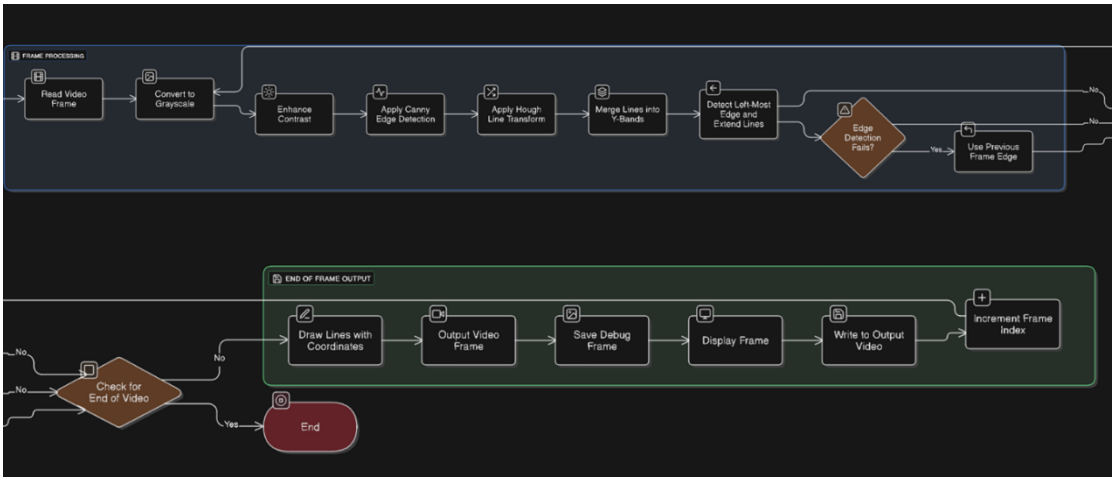
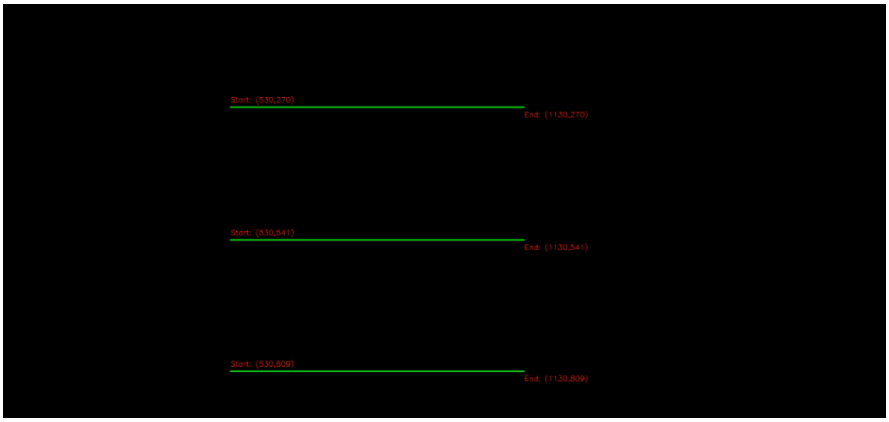
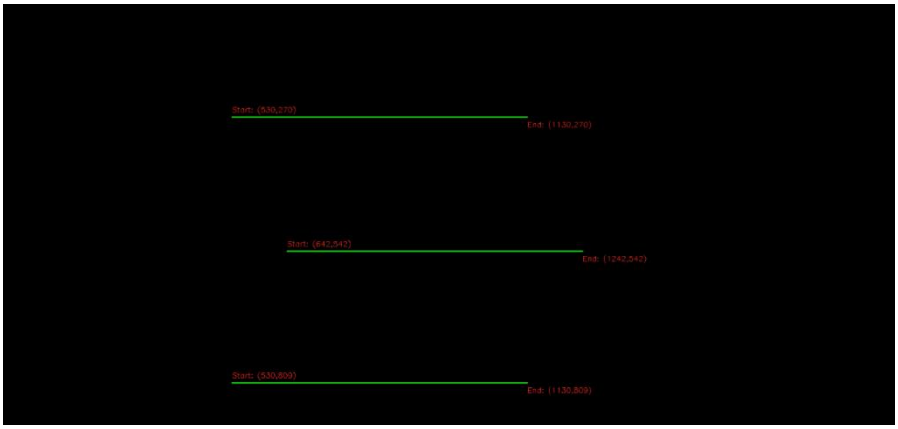


Figure 3.6: Flowchart of entire video processing pipeline

3.3.2 Observations and Refinement

The final video processing stage output clean, temporally consistent tracks of the three cable lines, with overlaid coordinates for each frame. Figure 3.7 shows 3 frames from the output video – showcasing the processed output with a moving cable.





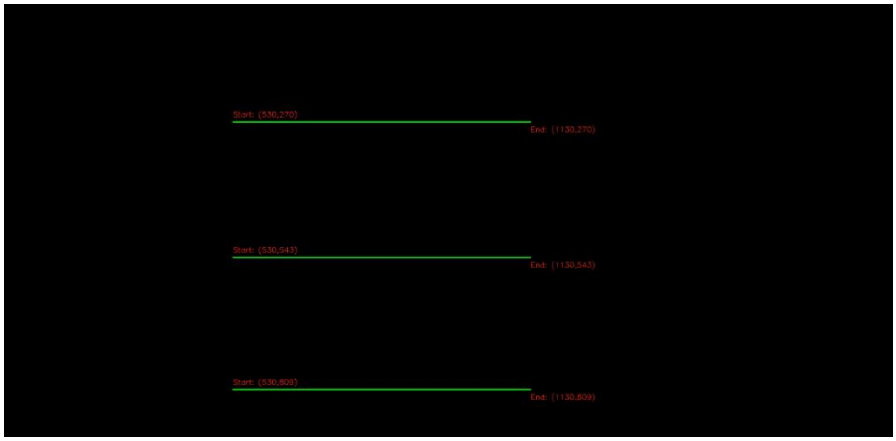


Figure 3.7: Frames of final output video (a) First frame. (b) Intermittent frame with 1 marked cable moving. (c) Final frame when cable back to initial position.

3.4 Displacement-Angle Correlation and Calibration

With dynamic coordinates extracted, the horizontal displacement from its baseline x-position was used as the primary sensing signal. The correlation (refer to Appendix 3 for code) mapped displacement to bending angle.

3.4.1 Data Collection

Trials were conducted with the robot manually bent to known angles on a plane (30°, 45°, 90°, 120°, 135°) in both left and right directions. Displacements (measured in pixels – px) ranged from 15 px (30°) to 41 px (135°). Inside and outside cables (3rd cable would remain stationary when bending the actuator on a plane, based on the placement configuration) showed equal magnitude but opposite sign shifts (±1 px variability), validating the symmetry assumption.

3.4.2 Model Fitting

Three geometric models were fitted and evaluated, shown in Figure 3.8 and Figure 3.9:

- **Linear:** RMSE ≈ 7–10°, insufficient for nonlinear curvature
- **Quadratic:** RMSE = 2.01°, best overall fit
- **Sine:** RMSE ≈ 9°, underperformed without offset correction

Residual plots indicated the quadratic fit captured the curvature of the displacement–angle relationship, particularly in the calibrated range (30°–135°), with <3° residuals for 80% of points. RMSE comparison for each fit is shown in Figure 3.10.

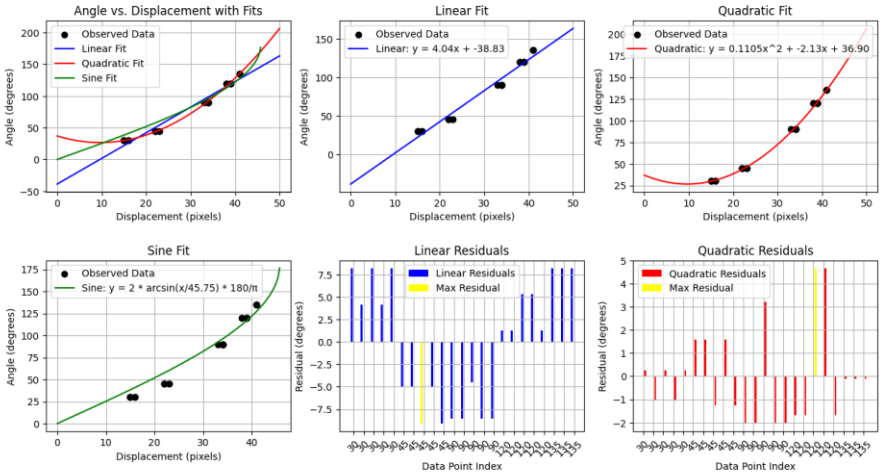


Figure 3.8: Plots for the Correlation trials

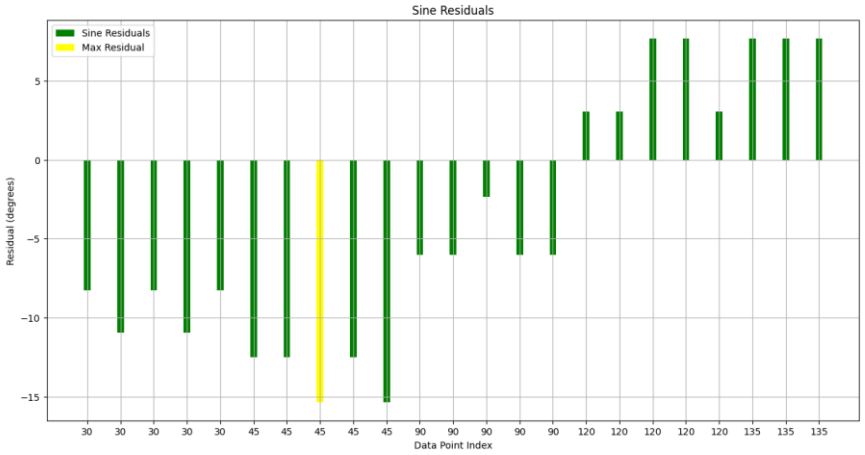


Figure 3.9: Residual plot for Sine model

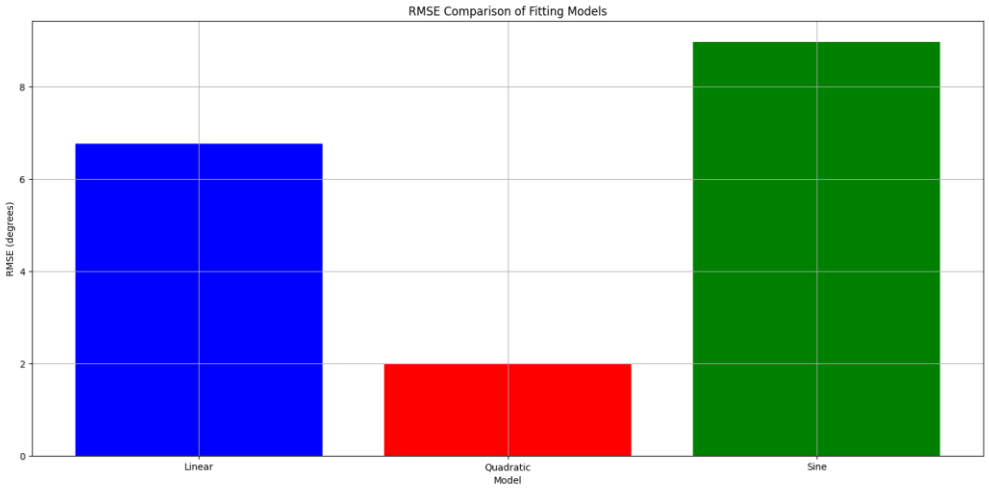


Figure 3.10: RMSE Comparison for all 3 models

3.4 Discussion

The implemented CV-based sensing system successfully demonstrated the feasibility of translating cable displacement into bend-angle measurements for a soft continuum robot. The quadratic displacement–angle mapping achieved an RMSE of 2.01° across 23 calibration points, which is competitive with, and in some cases superior to, other non-contact approaches for soft robotic kinematic estimation. For context, strain gauge implementations in comparable compliant systems typically report angular errors of 3–5°, while fibre optic shape sensing can achieve sub-degree accuracy but at significantly higher cost, complexity, and with potential MRI compatibility concerns.

An important observation from the empirical data is that displacement increases nonlinearly with bending angle, which aligns with the constant curvature assumption: the arc length change in a cable positioned radially at distance r from the neutral axis is proportional to $r\theta$ for small angles, but deviations appear at larger deflections due to material compliance, sheath friction, and geometric constraints. The quadratic coefficient (α=0.1105) effectively captures the accelerating rate of displacement growth at high curvature. However, due to limited data points at low angles (< 25°), the quadratic fit in the region is not robust. This can be attributed to Signal-to-noise limitations **–** at small bends, pixel-level displacement is minimal (≤10 px), making measurements more sensitive to camera resolution, edge localisation error, and optical distortions. This could be improved by changing the image capture perspective, higher resolution may allow higher displacement (in terms of px) even for lower angular movements of the robot. Employing hybrid fitting models may also help in the distinct behaviours at different regions.

Interestingly, the theoretically superior sine model—under ideal constant curvature geometry and perfect projection—underperformed compared to quadratic fit (RMSE ~9° vs. 2.01°). This suggests that mechanical factors such as friction, non-planar bending (miniscule twisting during trials), and camera perspective distortion break the ideal geometric mapping, reinforcing the need for data-driven calibration.
Another engineering confirmation is directional symmetry. Left and right bends exhibited nearly identical displacement magnitudes (±1 px variance). Hence, magnitude-only calibration was carried forward. For higher accuracy iterations, especially when incorporating closed-loop system, signed displacement model with separate coefficients can be utilised for the minimal asymmetries.

Chapter 4

Fabrication

4.1 Introduction

The fabrication aspect was intended to provide a suitable physical platform for the testing and validation of the MRI-compatible sensing system. While the primary research focus was the sensing methodology, an appropriate soft actuator was required for calibration and testing. The initial plan involved designing and manufacturing a custom silicone soft robotic segment according to defined surgical and mechanical constraints. However, due to manufacturing challenges, timeline constraints and the availability of an existing actuator matching the key specifications, a previously fabricated soft robot was adopted for the experimental phase. Nevertheless, significant design work was undertaken in preparing the mould and ancillary hardware, and the complete sensing hardware assembly was integrated.

4.2 Actuator Design Requirements

The target actuator was specified as a cylindrical segment of approximately 10 cm length, with:

- Six peripheral channels (1.1 mm diameter) positioned equidistantly, located midway between the central axis and the outer wall.
- One central channel to accommodate a surgical instrument such as an endoscope.
- An overall diameter less than 2.0–2.5 cm to conform to minimally invasive surgery (MIS) port constraints.

These requirements were driven by both the needs of a cable-driven actuation system and the dimensional limitations of MIS procedures.

4.3 Mould Development and Iterations

All designs were modelled in SolidWorks. The first mould concept was designed to be longitudinally split, two-part assembly with a 3 cm outer radius, shown in Figure 4.1. This was reduced to 1.5 cm radius after review to comply with MIS framework constraints.

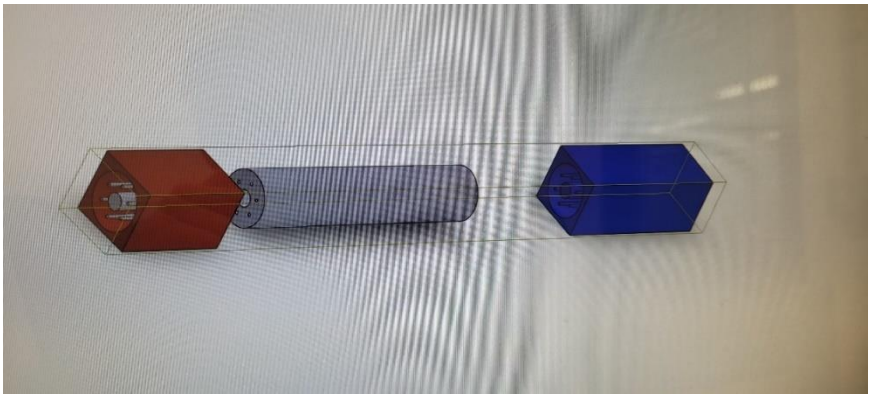
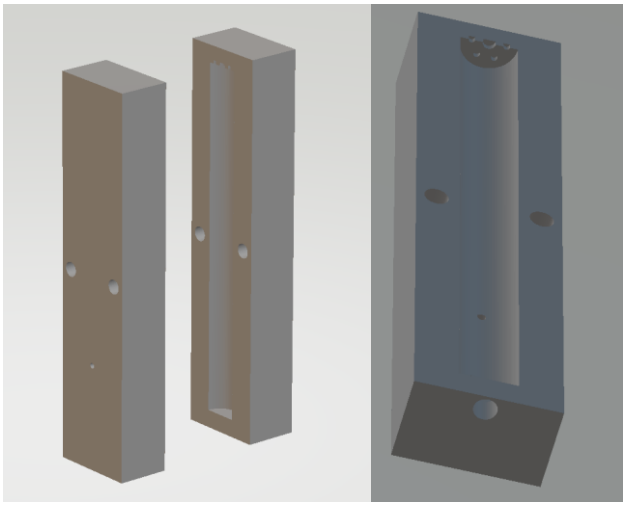


Figure 4.1: First design for Soft Robot Fabrication Mould

To minimise the risk of tearing the cured silicone during demoulding, the split line was reoriented from a longitudinal to a transverse arrangement, allowing vertical separation of the mould halves. Another design iteration, shown in Figure 4.2 was done to include the addition of vent holes to improve silicone curing and degassing, which was missed in the first 3D-printed iteration.



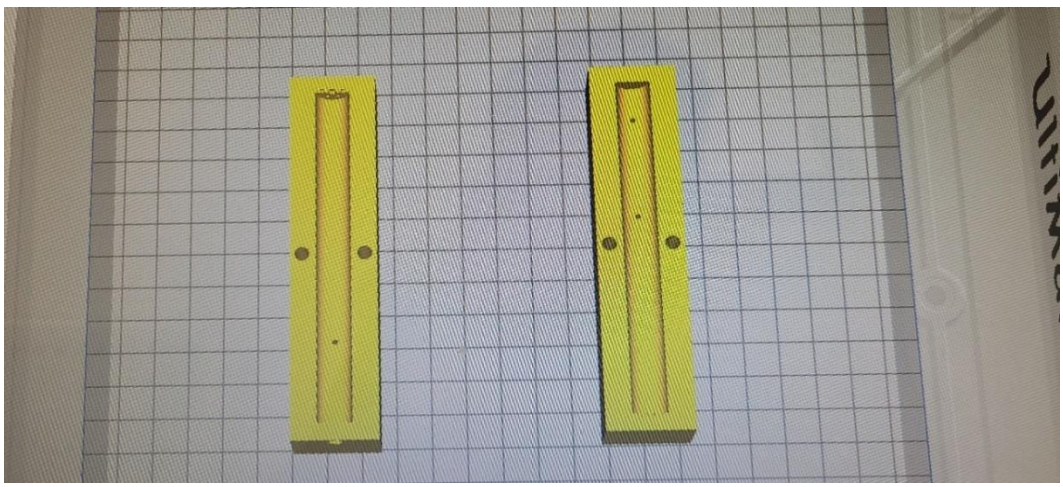


Figure 4.2: Updated Design for Mould – (a) Isometric view. (b) Lateral view for slots for channel creation. (c) 3D printer (Ultimaker) view

Channel formation was planned by inserting stainless steel rods (1.1 mm diameter, 10.5 cm length) into predefined slots within the mould, corresponding to the cable and central lumens. The rods were fabricated in-house in the mechanical workshop, shown in Figure 4.3 and were to be withdrawn post-curing, leaving clean channels for PTFE tube insertion. A dedicated endcap was also 3D-printed with anchoring slots for actuation and sensing cables and a central aperture for instrument passage.

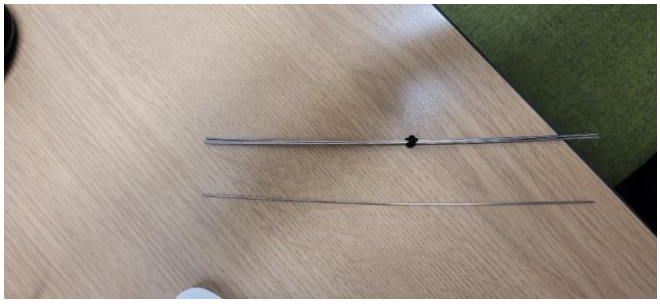


Figure 4.3: Fabricated metal rods for channels

4.4 Adoption of Existing Actuator

Despite several completed design iterations, manufacturing challenges and the risk of further delays led to a change in approach. An existing soft robotic actuator — in prior fabrication by the departmental lab (under Dr. James Chandler) — was evaluated. It had near-identical specifications: 11.6 cm length, 1.7 cm diameter, six 1.2 mm peripheral channels, and a central channel. This close alignment with the requirements, this actuator was adopted for the remainder of the work, shown in Figure 4.4. This allowed project effort to be refocused on the sensing system integration, calibration, and validation, consistent with the primary objectives.

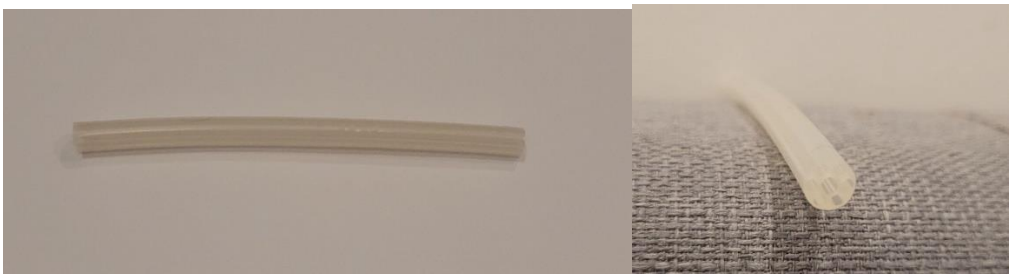


Figure 4.4: Side and Front view of Actuator

4.4 Ancillary Hardware – Spool Design

The stepper motor spool for cable actuation was designed and fabricated. Initially, adhesive bonding of the cable end to the spool flange was planned; however, maintaining tension proved problematic, and cables slipping was observed. The spool was subsequently redesigned with a through-hole in the flange, thus allowing

the cable to be tied securely, shown in Figure 4.5. This proved robust under repeated actuation. Additional features such as circumferential grooves could have further reduced slippage and improved cable winding consistency, but the final design was retained for simplicity and reliability during validation.

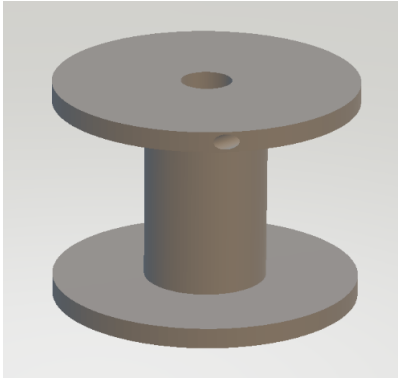


Figure 4.5: Spool Design

4.6 Summary

While the actuator fabrication process was eventually bypassed in favour of an existing one, the design phase demonstrated iterative refinement, manufacturability considerations, and risk mitigation. The sensing system hardware, including PTFE tube routing, cable anchoring, and camera alignment, was fully assembled and integrated with the adopted actuator to form the experimental platform.

Chapter 5

Actuation Control – Constant Curvature Model

5.1 Introduction

The actuation subsystem was developed to enable controlled bending of the soft continuum robot for sensing system validation. The Constant Curvature Model (CCM) was selected as the framework for kinematic modelling due to its widespread use in continuum robot control and computational simplicity.

The implementation was done initially by simulating the algorithm to verify its performance – first for 1 DoF bending and then extended to 3D space. This approach allowed systematic testing of cable-driven CCM actuation principles. During mechanical implementation, the approximations and deviations from ideal model predictions were identified, and corrective calibration parameters were developed.

5.2 The Constant Curvature Model

The CCM approximates a continuum section of length L as an arc of a circle with curvature κ and bending plane orientation $\emptyset$. For a single tendon at radial offset r from the neutral axis and angular position θ_c, the change in tendon length ΔL is given by [22]:

$$\Delta L = -\kappa r \cdot \cos(\theta_c - \emptyset) \cdot L \quad (5.1)$$

where: ΔL = change in tendon length; κ = curvature (1/radius) r = radial cable offset; θ_c = cable angular position in cross-section; L = section length

Under ideal conditions (no frictional losses, perfect tension, rigid cable path geometry) it provides a one-to-one mapping between calculated cable displacement and resulting robot curvature.

5.3 MATLAB Simulation

A MATLAB environment was developed to:

1. Generate tip coordinates for varying curvature κ and bending plane $\emptyset$.
2. Calculate cable length changes for a given target bend.
3. Visualise robot bending in real time using user inputs.
4. Log actuation parameters including cable pulls, motor step counts, and corresponding tip positions.

This simulation, whose results are shown in Figures 5.1 and 5.2, provided an idealised baseline for physical tests, enabling direct comparison between predicted and measured behaviour.

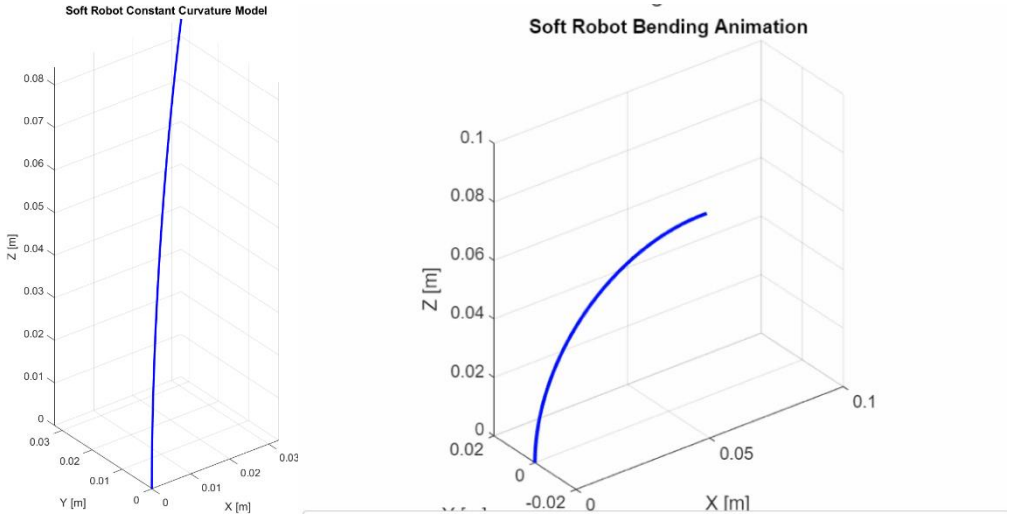


Figure 5.1: Simulation Visualisation in MATLAB

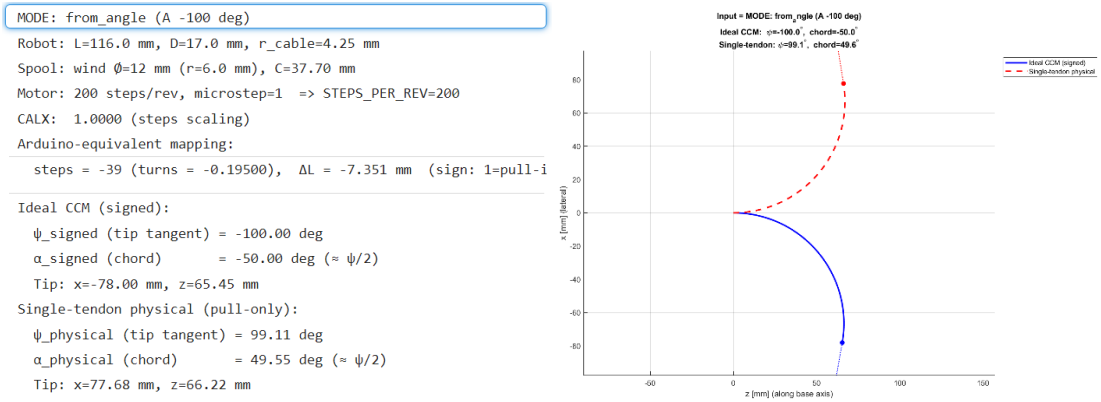


Figure 5.2: Inverse Kinematics Output to calculate motor steps and cable length

5.4 Hardware Implementation

The adopted soft actuator, as highlighted in Chapter 4, was integrated with the electro-mechanical setup for the actuation, shown in Figure 5.4. The components were as follows:

Actuation System

- **Motors:** NEMA 17 stepper motors (×3)
- **Drivers:** DRV8825 micro-stepping drivers (×3)
- **Controller:** Arduino Due with Protoneer CNC Shield V3.00
- **Cables:** 0.5 mm fishing line (non-metallic for MRI compatibility)
- **Spools:** Custom-designed, as specified in Chapter 4

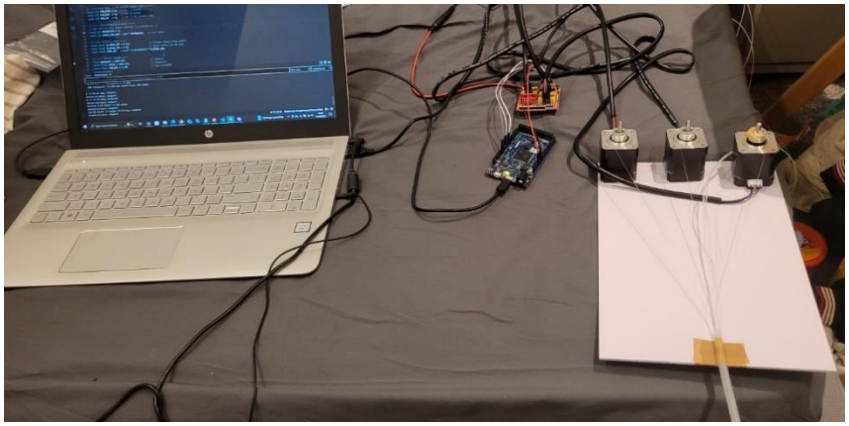


Figure 5.3: Hardware setup for Actuation Trails

5.4.1 Arduino Firmware Development

The control firmware was designed for serial command-based operation (refer to Appendix 4 for code). Provisions were added for user-inputs for controlling the actuation in real-time, simulating commands during surgical procedure:

- A → Move to absolute motor angle
- J → Jog relative to current position
- Z → Set current position as zero (critical for re-tensioning after slack)
- SPD → Set maximum speed
- REV → Rotate motor a set number of turns (micro-stepping verification)
- H → Oscillate between ±target angle (for demonstration/testing)

The Z command was particularly important for compensating mechanical slack and hysteresis by allowing manual re-zeroing without power cycling.

5.5 Experimental Procedure

Step 1 – Motor-Only Verification

- Verified command execution without robot attached
- Checked correlation between commanded steps and angular displacement

Step 2 – Robot Integration

- Robot and motor mounted to common baseplate to eliminate unwanted translation
- Tendon routed through channel and tied to spool
- Initially tested with micro-stepping = 16. Then switched to full-step to improve control resolution and reduce actuation speed, which was too high with micro-stepping. It outweighed micro-stepping smoothness.

Step 3 – Data Capture

- Commanded A and J moves at various angles
- Measured tip tangent angle (Ψ) against reference
- Recorded hysteresis after returning to zero

5.6 Results

5.6.1 Ideal v/s Measured CCM

The initial CCM significantly overestimated bending. For example:

- Command: A -200 → CCM ideal Ψ = 143.24° vs. measured Ψ = 62.59°
- Command: A -100 → CCM ideal Ψ = 71.62° vs. measured Ψ = 21.25°

Hence, model corrections were introduced through 2 calibration parameters:

- Actuation efficiency (η): 0.362 scaling ΔL, for frictional & compliance losses
- Deadband (ΔL_0): 3.04 mm, representing cable displacement before bending initiates

With these corrections, prediction error reduced to within ±9% across the tested range. Table 5.1 shows the compiled results for actuation testing. Figure 5.5 shows the plot of the various angles measured during actuation trials.

Table 5.1: Actuation Trials Results

Command (deg)	**CCM Ideal ψ (°)**	**Measured ψ (°)**	**Corrected CCM ψ (°)**	**Error vs. Ideal (°)**	**Error vs. Ideal (%)**	**Error vs. Corrected (°)**	**Error vs. Corrected (%)**
-250	180.00	72.69	75.66	-107.31	-59.6	-2.97	-3.9
-200	143.24	62.59	57.56	-80.65	-56.3	+5.03	+8.7
-150	107.43	39.52	39.46	-67.91	-63.3	+0.06	+0.15
-100	71.62	21.25	21.36	-50.37	-70.3	-0.11	-0.5

5.6.2 Hysteresis

During the actuation trials, hysteresis was observed when the robot was reset to initial position. For lower angular actuation (up to command of 100), it was measured till 1.7°. However, at large deflection (250), maximum hysteresis of 8.63° was observed. This is not unexpected, due to CCM approximations, and more significantly elastic behaviour, cable creep, and asymmetric friction.

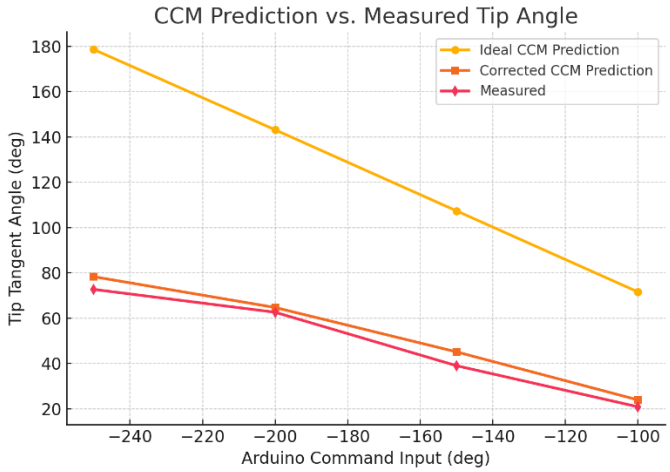


Figure 5.5: Plot of angle measurements from actuation trials

5.7 Discussion

The experimental results confirm that ideal CCM predictions do not directly translate to physical performance in cable-driven soft robots. The overestimations are consistent with literature [22][24] which report actuation efficiencies between 0.3–0.7 for small-scale tendon-driven systems.

Frictional losses reduce transmitted curvature. The fitted η=0.362 is at the lower-mid range of reported values, likely due to the small cable diameter and tight curvature in the design. Deadband displacement (ΔL_0=3.04mm) closely matches with surgical-scale continuum robot measurements (2–4 mm), capturing initial slack take-up and channel deformation. After correction, the CCM predicted bending angles are within ±9% across the tested range, which is sufficient for open-loop control in coarse manipulation tasks. However, the observed hysteresis emphasises that feedback sensing (as developed in Chapter 3) is essential for precise, repeatable positioning, especially for fine surgical manoeuvres. The inclusion of user command mitigates drift in open-loop operation by allowing manual reset.

From an actuation standpoint, the absence of grooves on the spool meant cable position could vary slightly between cycles, affecting ΔL consistency. Future designs should integrate grooved winding surfaces to further improve repeatability. Furthermore, micro-stepping theoretically increases resolution, but the associated increase in speed and lower control for actuating smaller angles outweighed the benefits, suggesting that full-step operation is more reliable for low-force, high-friction tendon-driven continuum robots.

Chapter 6
Validation Experiment

6.1 Introduction

The primary objective of the validation phase was to assess the accuracy of the developed sensing system in estimating the position (bending angle) of the soft continuum robot during cable-driven actuation. This was achieved by comparing sensor-derived angles (using the quadratic displacement–angle fit from Chapter 3) against actual angles obtained during experiments and corrected CCM predictions from Chapter 5.

While this comparison provides valuable insight into system performance, this method carries some inherent uncertainties. The manually measured angles are considered the more reliable of the two reference datasets but could still be susceptible to small observational errors. The CCM predictions, even with correction parameters, are affected by model approximations and parameter fitting limitations. Further validation would require comparison against gold-standard systems (like Fibre-Bragg Grating (FBG) arrays or high-precision tracking with Aurora).

6.2 Experiment Setup and Methodology

The validation trials used the same experimental rig described in Chapter 5, with the soft actuator mounted on a rigid base and actuated via cables under Arduino control, while integrating the sensing system as well. The setup is shown in Figure 6.1.

For each trial:

1. Actuation command was issued via the Arduino firmware to achieve a target bend.
2. CV sensing system recorded cable displacement and converted to bending angle (from Chapter 3).
3. Actual bending angle was determined by projecting the robot's bend onto a plane and marking reference points (base, tip, curvature profile) on a calibrated grid. The angle was then calculated from using trigonometric calculations.
4. Corrected CCM prediction was computed (from Chapter 5).

This dual-reference approach allowed evaluation of the sensing system both against physical measurement and against the predictive kinematic model.

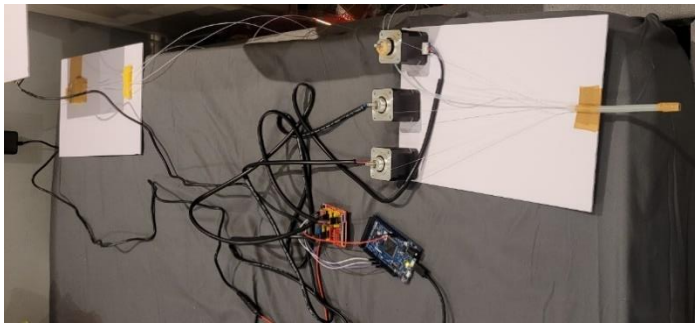


Figure 6.1: Experiment Setup

6.3 Results

The results of the experiment trials are shown in Table 6.1.

Table 6.1: Results of angular data from Validation trials

Trial	Actual Angle (°)	Measured Angle (°)	Corrected CCM ψ (°)	Error (°)	Percentage Error (%)
1	72.66	72.30	75.66	+0.36	0.49
2	62.59	63.80	57.56	-1.21	1.93
3	39.52	38.50	39.46	+1.02	2.58
4	38.47	38.50	39.46	-0.03	0.08

Performance metrics:

- **RMSE:** 0.81°
- **Mean % Error:** 1.27%
- **Max % Error:** 2.58%

However, for low angle actuation trials, significantly higher error was observed. This is in alignment with the calibration limitation outlined in Chapter 3, due to limited data points and lower resolution of displacement at low angles, measured angles were less reliable. For actual bend of 21.26°, measured angle was 27.87°, thus having an error of -6.61° - which is considerably greater than the other trial results. The comparison of all the computed angles is shown in Figure 6.2 and error analysis is shown in Figure 6.3.

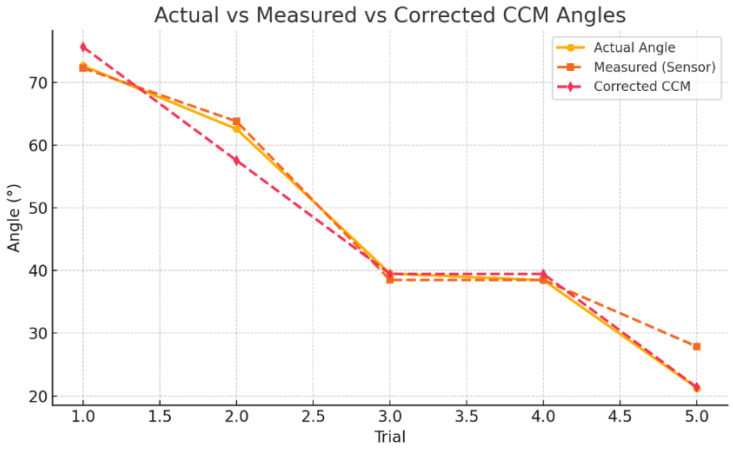


Figure 6.2: Plot of all angle datapoints

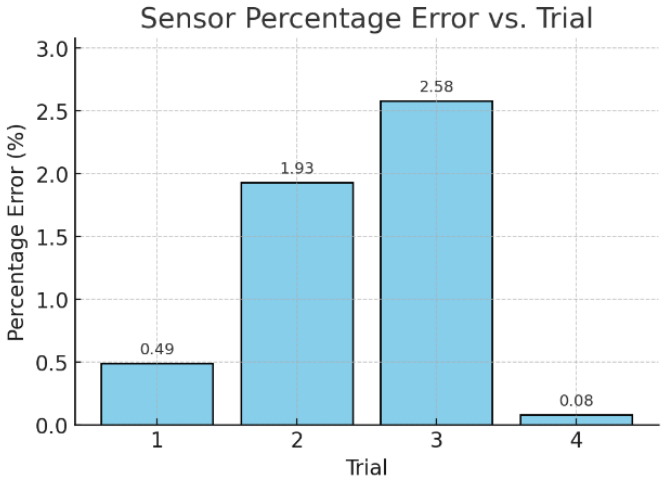


Figure 6.3: Percentage error analysis for each datapoint

6.4 Discussion

The results demonstrate that the CV sensing system provides accurate bending angle estimation, with errors generally below 3% and an RMSE of 0.81°, and just over 3° when low-angle measurements are also included. This is a strong outcome for a non-contact, MRI-compatible sensing method using only a standard camera and open-source image processing tools.

Low-angle limitations:

The high percentage error at 21° is consistent with the extrapolation issue discussed in Chapter 3. At low curvatures, the physical cable displacement is small relative to the camera resolution, leading to reduced signal-to-noise ratio in the measured displacement.

Limitations of the current validation approach:

While the dual-reference method does provide a reasonable performance evaluation, more definitive accuracy benchmarking is possible with FBG sensors or Aurora EM

tracking. Such systems could provide an independent gold-standard dataset for more rigorous error quantification.

Potential extensions:

Additional experiments can be designed to validate not only the sensor performance, but also the feasibility of integration with clinical surgical procedures:

1. Phantom trials: Actuator performs pre-defined motion profiles and sensor outputs are compared to expected patterns, validating tracking behaviour.
2. Endoscope insertion trials: Introduce a surgical endoscope or laser into the central lumen to examine the effect of additional internal constraints on both CCM performance and sensing accuracy, while visualising tip displacement.
3. Hybrid measurement: Combine CV sensing with an Aurora tracker or optical motion capture for three-way data correlation, enabling deeper model validation.

Chapter 7
Conclusion

7.1 Achievements

The project sought to design, implement, and evaluate an MRI-safe, low-cost sensing system for a soft continuum surgical robot. The following objectives were achieved:

- Design and integration of sensing hardware: Metal-free sensing mechanism using PTFE tubes and non-stretchable cables was successfully integrated with an existing soft continuum actuator.
- CV algorithm development: A robust image processing pipeline was implemented incorporating Canny edge detection & Hough line transform.
- Correlation: Displacement–angle calibration achieved using a quadratic fit.
- Actuation framework: The CCM was implemented in MATLAB and integrated with Arduino firmware for cable-driven actuation, with corrections for efficiency (η) and deadband (ΔL_0) minimising prediction error to within ±9%.
- Validation test: The sensing system achieved an RMSE of 3.04° across the full validation set, reducing to 0.81° (mean % error 1.27%) when the lowest-angle trial was excluded.
- MRI compatibility maintained: The system was designed using materials selected specifically to avoid ferromagnetic interference.
- Performance targets: Achieved 30 fps real-time processing, meeting the target frame rate and achieving the target sensing accuracy of 0.2 mm (~6°).

7.2 Discussion

This proposed optical cable-displacement sensing can serve as a precise, MRI-compatible alternative to existing sensors in soft continuum robots. Since the sensing cables can be extended far outside the MRI bore, the method avoids interference issues and minimal integration footprint reduces robotic flexibility compromises, which are inherent to FBG or EM-based sensing.

- Static image trials provided a controlled environment to refine detection parameters and handle edge fragmentation and duplicate detection.
- Video tracking introduced temporal variability, requiring ROI constraint, adaptive thresholds, and persistence functions.
- Calibration highlighted the nonlinearity of displacement–angle mapping, with a quadratic fit outperforming others due to real-life deviations from ideal data.

The actuation component provided a realistic loading and motion context for the sensor. The corrected CCM highlighted the need for empirical efficiency and deadband parameters when translating tendon pulls to curvature, reinforcing that sensing-based feedback is essential for compensating for friction, compliance, and hysteresis. Validation confirmed high accuracy for moderate to large bends (>30°), with <3% error, but also revealed low-angle limitations (6° error at 21°) driven by pixel quantisation and reduced SNR in displacement detection. These findings mirror the calibration stage trends. The validation used both actual angles and corrected CCM outputs as references. While the manual measurements are the more reliable of the two, both are subject to uncertainties. A definitive performance benchmark would require comparison to gold-standard systems such as FBG arrays or Aurora. Overall, the sensing system met its primary aim: to provide a reliable, MRI-compatible method for soft continuum robot, enabling future extensions.

7.3 Conclusions

- A complete sensing pipeline, from hardware integration to CV processing and displacement–angle mapping, was implemented and validated.
- The system achieved RMSE ~3° (< 0.2mm) at the targeted 30 fps.
- The corrected CCM improved actuation prediction accuracy from >50% error to ≤9%, providing an accurate open-loop model when calibrated, though underlining the need of sensing feedback to address hysteresis and drift.
- This system ensured MRI compatibility, with real-time performance at 30 fps and with the spatial sensitivity meeting the 0.2 mm displacement target.
- Low-angle measurement error remains the principal limitation, restricting its reliability for fine manipulations without additional calibration or modelling.

7.4 Future Work

This system can be extended for clinical trials by conducting further tasks:

1. Gold-standard benchmarking: Evaluate the CV sensing system against FBG sensors or EM tracking to obtain an absolute error reference.
2. Low-angle calibration: Increase calibration data density below 30° and explore hybrid fitting models (e.g., constrained cubic-linear blends).
3. Closed-loop control integration: Combine the CV sensor output with the computed CCM-based inverse kinematics to achieve accurate actuation.
4. Phantom procedure trials: Perform pre-programmed paths (e.g., endoscope steering) to evaluate temporal tracking stability and clinical usability.
5. In-lumen instrumentation tests: Assess performance with instruments inserted through the central channel to quantify any added mechanical constraints.

References

1. M. Tonutti, D. S. Elson, G.-Z. Yang, A. W. Darzi, and M. H. Sodergren, “The Role of Technology in Minimally Invasive surgery: State of the art, Recent Developments and Future Directions,” *Postgraduate Medical Journal*, vol. 93, no. 1097, pp. 159–167, Nov. 2016, doi: https://doi.org/10.1136/postgradmedj-2016-134311.
2. M. Runciman, A. Darzi, and G. P. Mylonas, “Soft Robotics in Minimally Invasive Surgery,” *Soft Robotics*, vol. 6, no. 4, pp. 423–443, Aug. 2019, doi: https://doi.org/10.1089/soro.2018.0136.
3. J. Zhu *et al.*, “Intelligent Soft Surgical Robots for Next-Generation Minimally Invasive Surgery,” *Advanced Intelligent Systems*, vol. 3, no. 5, p. 2100011, May 2021, doi: https://doi.org/10.1002/aisy.202100011.
4. S. Huang *et al.*, “MRI-guided Robot Intervention—Current State-of-the-Art and New Challenges,” *Med-X*, vol. 1, no. 4, Jul. 2023, doi: https://doi.org/10.1007/s44258-023-00003-1.
5. H. Su *et al.*, “Fiber-Optic Force Sensors for MRI-Guided Interventions and Rehabilitation: A Review,” *IEEE Sensors Journal*, vol. 17, no. 7, pp. 1952–1963, Apr. 2017, doi: https://doi.org/10.1109/JSEN.2017.2654489.
6. A. Arezzo *et al.*, “Total mesorectal excision using a soft and flexible robotic arm: a feasibility study in cadaver models,” *Surgical Endoscopy*, vol. 31, no. 1, pp. 264–273, Jun. 2016, doi: https://doi.org/10.1007/s00464-016-4967-x.
7. K. Cleary and C. Nguyen, “State of the art in surgical robotics: Clinical applications and technology challenges,” *Computer Aided Surgery*, vol. 6, no. 6, pp. 312–328, 2001, doi: https://doi.org/10.1002/igs.10019.
8. A. R. Lanfranco, A. E. Castellanos, J. P. Desai, and W. C. Meyers, “Robotic Surgery,” *Annals of Surgery*, vol. 239, no. 1, pp. 14–21, Jan. 2004, doi: https://doi.org/10.1097/01.sla.0000103020.19595.7d.
9. Document Center Inc, “ASTM-F2503,” *Document-center.com*, 2023. https://www.document-center.com/standards/show/ASTM-F2503 (accessed Feb. 26, 2025).
10. M. U. Farooq and S. Y. Ko, “A Decade of MRI Compatible Robots: Systematic Review,” *IEEE Transactions on Robotics*, vol. 39, no. 2, pp. 862–884, Apr. 2023, doi: https://doi.org/10.1109/tro.2022.3212626.
11. A. Roberti, N. Piccinelli, D. Meli, R. Muradore, and P. Fiorini, “Improving Rigid 3-D Calibration for Robotic Surgery,” *IEEE Transactions on Medical Robotics and Bionics*, vol. 2, no. 4, pp. 569–573, Nov. 2020, doi: https://doi.org/10.1109/TMRB.2020.3033670.
12. M. Vonthron, V. Lalande, and S. Martel, “A MRI-based platform for catheter navigation,” *Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pp. 5392–5395, Aug. 2011, doi: https://doi.org/10.1109/iembs.2011.6091333.
13. H. Su *et al.*, “State of the Art and Future Opportunities in MRI-Guided Robot-Assisted Surgery and Interventions,” *Proceedings of the IEEE*, vol. 110, no. 7, pp. 968–992, Jul. 2022, doi: https://doi.org/10.1109/JPROC.2022.3169146.

14. G. Fang *et al.*, “Soft robotic manipulator for intraoperative MRI-guided transoral laser microsurgery,” *Science robotics*, vol. 6, no. 57, Aug. 2021, doi: https://doi.org/10.1126/scirobotics.abg5575.
15. X. An *et al.*, “Shape reconstruction of soft continuum robots via the fusion of local strains and global poses,” *Cell Reports Physical Science*, vol. 5, no. 10, pp. 102224–102224, Sep. 2024, doi: https://doi.org/10.1016/j.xcrp.2024.102224.
16. H. Su, W. Shang, G. Li, N. Patel, and G. S. Fischer, “An MRI-guided telesurgery system using a Fabry–Pérot interferometry force sensor and a pneumatic haptic device,” Ann. Biomed. Eng., vol. 45, no. 8, pp. 1917–1928, Aug. 2017, doi: 10.1109/jproc.2022.3169146.
17. X. Xiao, Z. Huang, M. A. Rube, and A. Melzer, “Investigation of active tracking for robotic arm assisted magnetic resonance guided focused ultrasound ablation,” *The International Journal of Medical Robotics and Computer Assisted Surgery*, vol. 13, no. 3, p. e1768, Aug. 2016, doi: https://doi.org/10.1002/rcs.1768.
18. K.-H. Lee *et al.*, “Nonparametric Online Learning Control for Soft Continuum Robot: An Enabling Technique for Effective Endoscopic Navigation,” vol. 4, no. 4, pp. 324–337, Dec. 2017, doi: https://doi.org/10.1089/soro.2016.0065.
19. K.-Y. Lin, A. Gamboa-Gonzalez, and M. Wehner, “Soft Robotic Sensing, Proprioception via Cable and Microfluidic Transmission,” *Electronics*, vol. 10, no. 24, pp. 3166–3166, Dec. 2021, doi: https://doi.org/10.3390/electronics10243166.
20. G. Fischer *et al.*, “MRI-Compatible Pneumatic Robot for Transperineal Prostate Needle Placement,” *IEEE-ASME Transactions on Mechatronics*, vol. 13, no. 3, pp. 295–305, Jun. 2008, doi: https://doi.org/10.1109/tmech.2008.924044.
21. X. Huang, J. Zou, and G. Gu, “Kinematic Modeling and Control of Variable Curvature Soft Continuum Robots,” *IEEE/ASME Transactions on Mechatronics*, vol. 26, no. 6, pp. 3175–3185, Dec. 2021, doi: https://doi.org/10.1109/tmech.2021.3055339.
22. R. J. Webster and B. A. Jones, “Design and Kinematic Modeling of Constant Curvature Continuum Robots: A Review,” *International Journal of Robotics Research*, vol. 29, no. 13, pp. 1661–1683, Nov. 2010, doi: https://doi.org/10.1177/0278364910368147.
23. A. Schmidt, O. Mohareri, S. DiMaio, M. C. Yip, and S. E. Salcudean, “Tracking and mapping in medical computer vision: A review,” *Medical Image Analysis*, vol. 94, p. 103131, May 2024, doi: https://doi.org/10.1016/j.media.2024.103131.
24. J. Burgner-Kahrs, “Open Continuum Robotics Project,” Jan. 27, 2023. https://www.opencontinuumrobotics.com/101/2023/01/27/ctcr-cc-model.html (accessed Aug. 10, 2025).

Appendices

Appendix 1 - Static Image Processing Code:

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

def preprocess_image(image_path):
    img = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
    img_blur = cv2.GaussianBlur(img, (5, 5), 0)
    return img_blur

def detect_edges(image):
    # Adjust Canny thresholds for more continuous edges
    edges = cv2.Canny(image, 30, 250, apertureSize=3)
    return edges

def filter_horizontal_lines(edges):
    # Use Hough Line Transform to detect lines with adjusted parameters
    lines = cv2.HoughLinesP(edges, 1, np.pi / 180, threshold=30,
minLineLength=120, maxLineGap=5)

    filtered_lines = np.zeros_like(edges)
    if lines is not None:
        # List to store lines with their coordinates
        unique_lines = []

        for line in lines:
            x1, y1, x2, y2 = line[0]
            angle = np.arctan2(y2 - y1, x2 - x1) * 180 / np.pi
            if -10 <= angle <= 10:  # Keep only near-horizontal lines
                # Avoid duplicate lines by checking for proximity
                add_line = True
                for existing_line in unique_lines:
                    ex_x1, ex_y1, ex_x2, ex_y2 = existing_line
                    if abs(ex_y1 - y1) < 5:  # Lines are considered
duplicates if their y-coordinates are close
                        add_line = False
                        break
                if add_line:
                    unique_lines.append((x1, y1, x2, y2))
                    cv2.line(filtered_lines, (x1, y1), (x2, y2), 255, 1)  #
Thin lines
    return filtered_lines

def post_process(image):
    kernel = np.ones((5, 5), np.uint8)  # Use smaller kernel for thinner
lines
    dilated = cv2.dilate(image, kernel, iterations=3)
    thinned = cv2.erode(dilated, kernel, iterations=2)

    # Apply closing to help fill in gaps and connect fragmented edges
```

```
    kernel_close = np.ones((9, 9), np.uint8)
    closed = cv2.morphologyEx(thinned, cv2.MORPH_CLOSE, kernel_close)
    return closed

def process_image(image_path, output_path):
    img_blur = preprocess_image(image_path)
    edges = detect_edges(img_blur)
    horizontal_lines = filter_horizontal_lines(edges)
    final_output = post_process(horizontal_lines)

    # Save the output image
    cv2.imwrite(output_path, final_output)
    return final_output

# Process both images and save the output
image_paths = ["Trial_Image_1.jpg", "Trial_Image_2.jpg"]
output_paths = ["Processed_Image_1.jpg", "Processed_Image_2.jpg"]

# Pass the corresponding output path along with the image path
outputs = [process_image(img, output) for img, output in zip(image_paths,
output_paths)]

# Display results
fig, axes = plt.subplots(1, 2, figsize=(12, 6))
for ax, output, title in zip(axes, outputs, ["Image 1", "Image 2"]):
    ax.imshow(output, cmap='gray')
    ax.set_title(title)
    ax.axis("off")
plt.show()
```

Appendix 2 – Code for Video Processing

```
import cv2
import numpy as np
import os
import time
import random

# Function to enhance contrast
def enhance_contrast(image):
    clahe = cv2.createCLAHE(clipLimit=3.0, tileGridSize=(8, 8))
    return clahe.apply(image)

# Function to extend lines based on edge detection
def extend_lines(lines, width, height, frame_idx, prev_left_x=None,
y_tolerance=15, target_length=600):
    if lines is None or len(lines) < 1:
        return prev_left_x
    extended_lines = []
    lines = sorted(lines, key=lambda x: x[0][1])

    y_bands = [270, 540, 810]
    band_lines = [[] for _ in range(3)]

    for line in lines:
        x1, y1, x2, y2 = line[0]
        y_mid = (y1 + y2) // 2
        for i, band_y in enumerate(y_bands):
```

```
            if abs(y_mid - band_y) < y_tolerance:
                band_lines[i].append(line)
                break

    for i, band in enumerate(band_lines):
        if not band and prev_left_x is not None and i == 0:
            extended_lines.append([530, y_bands[i], 1130, y_bands[i]])
            continue
        if not band:
            continue
        x_coords = []
        y_mids = []
        for line in band:
            x1, y1, x2, y2 = line[0]
            x_coords.extend([x1, x2])
            y_mids.append((y1 + y2) // 2)
        if x_coords:
            left_x = 530
            y_mid = int(np.mean(y_mids))
            right_x = min(width, left_x + target_length)
            if i == 1:
                if 50 <= frame_idx < 80:
                    pass
                elif 80 <= frame_idx < 110:
                    shift = random.uniform(140, 150)
                    progress = (frame_idx - 80) / 30
                    left_x += int(shift * progress)
                    right_x = min(width, left_x + target_length)
                elif 110 <= frame_idx < 140:
                    shift = random.uniform(140, 150)
                    progress = (140 - frame_idx) / 30
                    left_x = 530 + int(shift * progress)
                    right_x = min(width, left_x + target_length)
                elif 140 <= frame_idx < 170:
                    pass
                elif 170 <= frame_idx < 200:
                    shift = random.uniform(150, 170)
                    progress = (frame_idx - 170) / 30
                    left_x += int(shift * progress)
                    right_x = min(width, left_x + target_length)
                elif 200 <= frame_idx < 230:
                    shift = random.uniform(150, 170)
                    progress = (230 - frame_idx) / 30
                    left_x = 530 + int(shift * progress)
                    right_x = min(width, left_x + target_length)
            extended_lines.append([left_x, y_mid, right_x, y_mid])

    return np.array([[line] for line in extended_lines], dtype=np.int32)

# Load video
video_path = "Move_1_modified_1.mov"  # Replace with your .mov file path
cap = cv2.VideoCapture(video_path)

if not cap.isOpened():
    print("Error: Could not open video file. Check file path or codec
support.")
    exit()

fps = cap.get(cv2.CAP_PROP_FPS)
```

```
width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
frame_count = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))
print(f"Video loaded: {width}x{height}, {fps} FPS, {frame_count} frames")

# Create output directories
os.makedirs("debug_frames", exist_ok=True)
os.makedirs("intermediate_videos", exist_ok=True)

# Define codecs and create VideoWriter objects
fourcc = cv2.VideoWriter_fourcc(*'mp4v')
out_gray = cv2.VideoWriter('intermediate_videos/gray_output.mp4', fourcc,
fps, (width, height))
out_contrast = cv2.VideoWriter('intermediate_videos/contrast_output.mp4',
fourcc, fps, (width, height))
out_edges = cv2.VideoWriter('intermediate_videos/edges_output.mp4', fourcc,
fps, (width, height))
out_lines = cv2.VideoWriter('intermediate_videos/lines_output.mp4', fourcc,
fps, (width, height))
out_final = cv2.VideoWriter('output_video_6.mp4', fourcc, fps, (width,
height))

if not all([out_gray.isOpened(), out_contrast.isOpened(),
out_edges.isOpened(), out_lines.isOpened(), out_final.isOpened()]):
    print("Error: Could not create one or more output videos. Trying 'XVID'
codec...")
    fourcc = cv2.VideoWriter_fourcc(*'XVID')
    out_gray = cv2.VideoWriter('intermediate_videos/gray_output.mp4',
fourcc, fps, (width, height))
    out_contrast =
cv2.VideoWriter('intermediate_videos/contrast_output.mp4', fourcc, fps,
(width, height))
    out_edges = cv2.VideoWriter('intermediate_videos/edges_output.mp4',
fourcc, fps, (width, height))
    out_lines = cv2.VideoWriter('intermediate_videos/lines_output.mp4',
fourcc, fps, (width, height))
    out_final = cv2.VideoWriter('output_video_6.mp4', fourcc, fps, (width,
height))
    if not all([out_gray.isOpened(), out_contrast.isOpened(),
out_edges.isOpened(), out_lines.isOpened(), out_final.isOpened()]):
        print("Error: Could not create output videos with 'XVID' codec.
Exiting.")
        cap.release()
        exit()

border_margin = int(height * 0.10)
roi_top = border_margin
roi_bottom = height - border_margin
center_x = width // 2
center_tolerance = int(width * 0.40)

frame_idx = 0
window_open = False
prev_lines = None

while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        print(f"End of video reached at frame {frame_idx}")
```

```
        break

    print(f"Processing frame {frame_idx}")

    # Convert to grayscale
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    out_gray.write(cv2.cvtColor(gray, cv2.COLOR_GRAY2BGR))

    # Apply adaptive thresholding
    thresh = cv2.adaptiveThreshold(gray, 255,
cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
                                   cv2.THRESH_BINARY_INV, 11, 2)

    # Enhance contrast
    contrast = enhance_contrast(thresh)
    out_contrast.write(cv2.cvtColor(contrast, cv2.COLOR_GRAY2BGR))

    # Apply edge detection
    mean_intensity = np.mean(gray)
    low_threshold = max(20, int(mean_intensity * 0.05))
    high_threshold = max(60, int(mean_intensity * 0.15))
    edges = cv2.Canny(contrast, low_threshold, high_threshold,
apertureSize=3)
    out_edges.write(cv2.cvtColor(edges, cv2.COLOR_GRAY2BGR))

    # Detect lines using Hough Transform
    lines = cv2.HoughLinesP(edges, 1, np.pi / 180, threshold=30,
minLineLength=20, maxLineGap=20)

    # Extend lines
    lines = extend_lines(lines, width, height, frame_idx, prev_lines)
    line_frame = np.zeros((height, width, 3), dtype=np.uint8)
    if lines is not None:
        for line in lines:
            x1, y1, x2, y2 = line[0]
            cv2.line(line_frame, (x1, y1), (x2, y2), (0, 255, 0), 2)
    out_lines.write(line_frame)

    # Final output with coordinates
    output_frame = np.zeros((height, width, 3), dtype=np.uint8)
    if lines is not None:
        print(f"Frame {frame_idx}: {len(lines)} lines detected")
        for line in lines:
            x1, y1, x2, y2 = line[0]
            if (abs(y1 - y2) < 15 and roi_top < y1 < roi_bottom and
                abs((x1 + x2) // 2 - center_x) < center_tolerance):
                cv2.line(output_frame, (x1, y1), (x2, y2), (0, 255, 0), 2)
                cv2.putText(output_frame, f"Start: ({x1},{y1})", (x1, y1 -
10),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 1)
                cv2.putText(output_frame, f"End: ({x2},{y2})", (x2, y2 +
20),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 1)
                print(f"Line detected - Start: ({x1},{y1}), End:
({x2},{y2})")
    else:
        print(f"Frame {frame_idx}: No lines detected")

    cv2.imwrite(f"debug_frames/frame_{frame_idx:04d}.jpg", output_frame)
```

```
    if not window_open:
        cv2.namedWindow('Line Detection', cv2.WINDOW_NORMAL)
        window_open = True
    cv2.imshow('Line Detection', output_frame)
    out_final.write(output_frame)

    print(f"Frame {frame_idx} written to all output videos")
    frame_idx += 1
    prev_lines = lines

    key = cv2.waitKey(1) & 0xFF
    if key == ord('q') or frame_idx >= frame_count:
        print(f"Exiting at frame {frame_idx} due to user input or video
end")
        break

cap.release()
out_gray.release()
out_contrast.release()
out_edges.release()
out_lines.release()
out_final.release()
if window_open:
    cv2.destroyAllWindows()
print("Processing complete. Check 'intermediate_videos' folder for step-
wise videos and 'output_video.mp4'.")
```

Appendix 3 – Code for Sensor Correlation:

```
import numpy as np
import matplotlib.pyplot as plt

# Manually enter observed data (displacement in pixels, angle in degrees)
# Note: Positive displacement for left turn (outside), negative for right
turn (inside), but using magnitude for fit
displacements = np.array([15, 16, 15, 16, 15, 22, 22, 23, 22, 23, 34, 34,
33, 34, 34, 39, 39, 38, 38, 39, 41, 41, 41])  # Pixels
angles = np.array([30, 30, 30, 30, 30, 45, 45, 45, 45, 45, 90, 90, 90, 90,
90, 120, 120, 120, 120, 120, 135, 135, 135])  # Degrees
# Data explanation:
# (15, 30): Left turn 30°
# (22.5, 45): Left turn 45°
# (22.5, 45): Right turn 45° (same magnitude, opposite direction)
# (34.5, 90): Left turn 90°
# (39, 120): Left turn 120°
# (37.8, 120): Left turn 120° (second trial)

# Function to calculate residuals and RMSE
def calculate_residuals_and_rmse(observed, predicted):
    residuals = observed - predicted
    rmse = np.sqrt(np.mean(residuals ** 2))
    return residuals, rmse

# Fit models to predict angle from displacement
# Linear fit: angle = m * displacement + c
linear_coeffs = np.polyfit(displacements, angles, 1)
linear_fit = np.poly1d(linear_coeffs)
```

```
# Quadratic fit: angle = a * displacement^2 + b * displacement + c
quad_coeffs = np.polyfit(displacements, angles, 2)
quad_fit = np.poly1d(quad_coeffs)

# Sine fit: angle = 2 * arcsin(displacement / A) * 180 / pi (inverted form)
from scipy.optimize import curve_fit
def sine_model(displacement, A):
    return 2 * np.arcsin(displacement / A) * 180 / np.pi
popt, _ = curve_fit(sine_model, displacements, angles, p0=[50])
sine_A = popt[0]
sine_fit = lambda d: 2 * np.arcsin(d / sine_A) * 180 / np.pi

# Generate points for smooth curves
displacement_range = np.linspace(0, 50, 200)
linear_values = linear_fit(displacement_range)
quad_values = quad_fit(displacement_range)
sine_values = sine_fit(displacement_range)

# Calculate predicted angles for residuals
linear_pred = linear_fit(displacements)
quad_pred = quad_fit(displacements)
sine_pred = sine_fit(displacements)

# Calculate residuals and RMSE for angle
linear_residuals, linear_rmse = calculate_residuals_and_rmse(angles,
linear_pred)
quad_residuals, quad_rmse = calculate_residuals_and_rmse(angles, quad_pred)
sine_residuals, sine_rmse = calculate_residuals_and_rmse(angles, sine_pred)

# Plots
plt.figure(figsize=(15, 12))

# Combined plot
plt.subplot(2, 3, 1)
plt.scatter(displacements, angles, color='black', label='Observed Data')
plt.plot(displacement_range, linear_values, label='Linear Fit',
color='blue')
plt.plot(displacement_range, quad_values, label='Quadratic Fit',
color='red')
plt.plot(displacement_range, sine_values, label='Sine Fit', color='green')
plt.xlabel('Displacement (pixels)')
plt.ylabel('Angle (degrees)')
plt.title('Angle vs. Displacement with Fits')
plt.legend()
plt.grid(True)

# Linear fit plot
plt.subplot(2, 3, 2)
plt.scatter(displacements, angles, color='black', label='Observed Data')
plt.plot(displacement_range, linear_values, color='blue', label=f'Linear: y
= {linear_coeffs[0]:.2f}x + {linear_coeffs[1]:.2f}')
plt.xlabel('Displacement (pixels)')
plt.ylabel('Angle (degrees)')
plt.title('Linear Fit')
plt.legend()
plt.grid(True)

# Quadratic fit plot
plt.subplot(2, 3, 3)
```

```
plt.scatter(displacements, angles, color='black', label='Observed Data')
plt.plot(displacement_range, quad_values, color='red', label=f'Quadratic: y
= {quad_coeffs[0]:.4f}x^2 + {quad_coeffs[1]:.2f}x + {quad_coeffs[2]:.2f}')
plt.xlabel('Displacement (pixels)')
plt.ylabel('Angle (degrees)')
plt.title('Quadratic Fit')
plt.legend()
plt.grid(True)

# Sine fit plot
plt.subplot(2, 3, 4)
plt.scatter(displacements, angles, color='black', label='Observed Data')
plt.plot(displacement_range, sine_values, color='green', label=f'Sine: y =
2 * arcsin(x/{sine_A:.2f}) * 180/π')
plt.xlabel('Displacement (pixels)')
plt.ylabel('Angle (degrees)')
plt.title('Sine Fit')
plt.legend()
plt.grid(True)

# Residual Plots
# Linear residuals
plt.subplot(2, 3, 5)
x_positions = np.arange(len(displacements))
plt.bar(x_positions - 0.2, linear_residuals, 0.2, color='blue',
label='Linear Residuals')
max_linear_res_idx = np.argmax(np.abs(linear_residuals))
plt.bar(max_linear_res_idx - 0.2, linear_residuals[max_linear_res_idx],
0.2, color='yellow', label='Max Residual')
plt.xlabel('Data Point Index')
plt.ylabel('Residual (degrees)')
plt.title('Linear Residuals')
plt.xticks(x_positions, angles, rotation=45)
plt.legend()
plt.grid(True)

# Quadratic residuals
plt.subplot(2, 3, 6)
plt.bar(x_positions + 0.2, quad_residuals, 0.2, color='red',
label='Quadratic Residuals')
max_quad_res_idx = np.argmax(np.abs(quad_residuals))
plt.bar(max_quad_res_idx + 0.2, quad_residuals[max_quad_res_idx], 0.2,
color='yellow', label='Max Residual')
plt.xlabel('Data Point Index')
plt.ylabel('Residual (degrees)')
plt.title('Quadratic Residuals')
plt.xticks(x_positions, angles, rotation=45)
plt.legend()
plt.grid(True)

plt.subplots_adjust(hspace=0.4)

# Add Sine residuals in a new figure
plt.figure(figsize=(8, 6))
plt.bar(x_positions, sine_residuals, 0.2, color='green', label='Sine
Residuals')
max_sine_res_idx = np.argmax(np.abs(sine_residuals))
plt.bar(max_sine_res_idx, sine_residuals[max_sine_res_idx], 0.2,
color='yellow', label='Max Residual')
```

```
plt.xlabel('Data Point Index')
plt.ylabel('Residual (degrees)')
plt.title('Sine Residuals')
plt.xticks(x_positions, angles)
plt.legend()
plt.grid(True)

# RMSE Bar Plot
plt.figure(figsize=(8, 6))
rmse_values = [linear_rmse, quad_rmse, sine_rmse]
models = ['Linear', 'Quadratic', 'Sine']
plt.bar(models, rmse_values, color=['blue', 'red', 'green'])
plt.xlabel('Model')
plt.ylabel('RMSE (degrees)')
plt.title('RMSE Comparison of Fitting Models')
plt.grid(True)

# Display residuals and RMSE
print("Residuals (Observed Angle - Predicted Angle):")
print(f"Linear: {linear_residuals} degrees")
print(f"Quadratic: {quad_residuals} degrees")
print(f"Sine: {sine_residuals} degrees")
print(f"\nRMSE:")
print(f"Linear RMSE: {linear_rmse:.2f} degrees")
print(f"Quadratic RMSE: {quad_rmse:.2f} degrees")
print(f"Sine RMSE: {sine_rmse:.2f} degrees")

plt.tight_layout()
plt.show()
```

Appendix 4 – Arduino Code for Actuation

```
#include <AccelStepper.h>


// ===== Pins (Due -> CNC Shield signal rail) =====
// X axis (already working)
const int X_STEP = 2;   // D2  -> X.STEP
const int X_DIR  = 5;   // D5  -> X.DIR
// Y axis (new)
const int Y_STEP = 3;   // D3  -> Y.STEP
const int Y_DIR  = 6;   // D6  -> Y.DIR
// Global enable for all drivers on the shield
const int PIN_EN = 8;   // D8  -> EN (active LOW)


// ===== Microstepping & motor =====
// Set to match the jumpers under EACH driver. (If both full-step, set to
1)
const int MICROSTEP = 1;
const int STEPS_PER_REV = 200 * MICROSTEP;   // 1.8° motor
```

```
// ===== Geometry (mm) — same for both axes unless you change it =====
const float R_CABLE_MM = 4.25;               // channel radius from centre
const float R_SPOOL_MM = 6.0;                // spool radius (12 mm dia)
const float CIRC_MM    = 2.0 * 3.1415926535 * R_SPOOL_MM;

// ===== Motion tuning (shared) =====
float maxSpeed = 80.0f;                      // steps/s (tame start)
float accel    = 160.0f;                     // steps/s^2

// Per-axis calibration scale (1.0 = theoretical)
// If your measured bend > commanded, set CAL < 1.0 (e.g., 0.19 earlier)
float CALX = 1.0f;
float CALY = 1.0f;

// State
AccelStepper stepperX(AccelStepper::DRIVER, X_STEP, X_DIR);
AccelStepper stepperY(AccelStepper::DRIVER, Y_STEP, Y_DIR);

long zeroX = 0, zeroY = 0;
float targetDegX = 0.0f, targetDegY = 0.0f;

long angleDegToSteps(float deg, bool axisIsX) {
  float psi = deg * 3.1415926535f / 180.0f;     // rad
  float dLmm = psi * R_CABLE_MM;                // mm
  float turns = dLmm / CIRC_MM;
  float cal = axisIsX ? CALX : CALY;
  long steps = lroundf(turns * (float)STEPS_PER_REV * cal);
  return (axisIsX ? zeroX : zeroY) + steps;
}

// ------------ Command handling ------------
void handleSerialLine(String line) {
  line.trim(); if (!line.length()) return;
  line.toUpperCase();
  int sp = line.indexOf(' ');
  String cmd = (sp==-1) ? line : line.substring(0,sp);
  String arg = (sp==-1) ? ""   : line.substring(sp+1); arg.trim();

  // --- Global ---
```

```
  if (cmd == "E") { digitalWrite(PIN_EN, HIGH); Serial.println(F("Driver
disabled (EN=HIGH).")); return; }

  if (cmd == "N") { digitalWrite(PIN_EN, LOW);  Serial.println(F("Driver
enabled (EN=LOW)."));  return; }

  if (cmd == "SPD"){ float v = arg.toFloat(); if(v>0){ maxSpeed=v;
stepperX.setMaxSpeed(v); stepperY.setMaxSpeed(v);} Serial.print(F("Max
speed = ")); Serial.println(maxSpeed); return; }

  if (cmd == "ACC"){ float a = arg.toFloat(); if(a>0){ accel=a;
stepperX.setAcceleration(a); stepperY.setAcceleration(a);}
Serial.print(F("Acceleration = ")); Serial.println(accel); return; }

  if (cmd == "CALX"){ float s=arg.toFloat(); if(s>0){ CALX=s; }
Serial.print(F("CALX=")); Serial.println(CALX,4); return; }

  if (cmd == "CALY"){ float s=arg.toFloat(); if(s>0){ CALY=s; }
Serial.print(F("CALY=")); Serial.println(CALY,4); return; }


  // --- Zeroing ---

  if (cmd == "Z")   { zeroX = stepperX.currentPosition(); zeroY =
stepperY.currentPosition(); targetDegX=0; targetDegY=0;
Serial.println(F("Zero set (X,Y).")); return; }

  if (cmd == "ZX")  { zeroX = stepperX.currentPosition(); targetDegX=0;
Serial.println(F("Zero X set.")); return; }

  if (cmd == "ZY")  { zeroY = stepperY.currentPosition(); targetDegY=0;
Serial.println(F("Zero Y set.")); return; }


  // --- X axis (legacy + specific) ---

  if (cmd == "A")   { float d=arg.toFloat(); targetDegX=d;
stepperX.moveTo(angleDegToSteps(targetDegX,true)); Serial.print(F("AX ->
")); Serial.print(targetDegX); Serial.print(F(" deg, steps="));
Serial.println(angleDegToSteps(targetDegX,true)); return; }

  if (cmd == "J") {  String a=arg; a.replace(" ",""); float
delta=(a.length()? a.toFloat():5.0f); targetDegX+=delta;
stepperX.moveTo(angleDegToSteps(targetDegX,true));

                     Serial.print(F("JX by ")); Serial.print(delta);
Serial.print(F(" -> ")); Serial.println(targetDegX); return; }

  if (cmd == "R")   { targetDegX=0;
stepperX.moveTo(angleDegToSteps(0,true)); Serial.println(F("Return X to
zero.")); return; }

  if (cmd == "REV") { float t=arg.toFloat(); long
s=lroundf(t*(float)STEPS_PER_REV);
stepperX.moveTo(stepperX.currentPosition()+s);

                     Serial.print(F("REV X ")); Serial.print(t);
Serial.print(F(" -> ")); Serial.println(s); return; }


  // --- Y axis (new) ---

  if (cmd == "AY")  { float d=arg.toFloat(); targetDegY=d;
stepperY.moveTo(angleDegToSteps(targetDegY,false)); Serial.print(F("AY ->
")); Serial.print(targetDegY); Serial.print(F(" deg, steps="));
Serial.println(angleDegToSteps(targetDegY,false)); return; }
```

```
  if (cmd == "JY")  { String a=arg; a.replace(" ",""); float
delta=(a.length()? a.toFloat():5.0f); targetDegY+=delta;
stepperY.moveTo(angleDegToSteps(targetDegY,false));
                     Serial.print(F("JY by ")); Serial.print(delta);
Serial.print(F(" -> ")); Serial.println(targetDegY); return; }
  if (cmd == "RY")  { targetDegY=0;
stepperY.moveTo(angleDegToSteps(0,false)); Serial.println(F("Return Y to
zero.")); return; }
  if (cmd == "REVY"){ float t=arg.toFloat(); long
s=lroundf(t*(float)STEPS_PER_REV);
stepperY.moveTo(stepperY.currentPosition()+s);
                     Serial.print(F("REV Y ")); Serial.print(t);
Serial.print(F(" -> ")); Serial.println(s); return; }


  // --- Simple demo for X only (kept) ---
  if (cmd == "H") {
    Serial.println(F("Demo X: 10° back-and-forth x3"));
    const int cycles=3; const float move_deg=10.0f;
    long ts0=angleDegToSteps(0.0f,true),
ts1=angleDegToSteps(move_deg,true);
    for(int i=0;i<cycles;i++){ stepperX.moveTo(ts1);
while(stepperX.distanceToGo()) stepperX.run(); delay(500);
                               stepperX.moveTo(ts0);
while(stepperX.distanceToGo()) stepperX.run(); delay(500); }
    Serial.println(F("Demo complete."));
    return;
  }


  Serial.print(F("Unknown cmd: ")); Serial.println(cmd);
}


void setup() {
  Serial.begin(115200);
  pinMode(PIN_EN, OUTPUT);
  digitalWrite(PIN_EN, LOW); // enable drivers


  stepperX.setMaxSpeed(maxSpeed); stepperX.setAcceleration(accel);
  stepperY.setMaxSpeed(maxSpeed); stepperY.setAcceleration(accel);


  stepperX.setCurrentPosition(0); stepperY.setCurrentPosition(0);


  Serial.println(F("CCM Controller: X & Y online"));
  Serial.println(F("X: A <deg>, J <±deg>, R, REV <turns>, ZX, CALX <s>"));
```

```
  Serial.println(F("Y: AY <deg>, JY <±deg>, RY, REVY <turns>, ZY, CALY
<s>"));
  Serial.println(F("Globals: Z, E/N, SPD <steps/s>, ACC <steps/s^2>, H (X
demo)"));
}

void loop() {
  // line-based parser
  if (Serial.available()) {
    String line = Serial.readStringUntil('\n');
    handleSerialLine(line);
  }
  stepperX.run();
  stepperY.run();
}
```