Computer Vision Based Angle Sensing for Continuum Robots

James Nguyen

***Abstract*— Finding the orientation of continuum robots differs greatly from rigid link robotics. Using computer vision (CV) to understand the position and orientation of a continuum robot can be a simpler and easier process than computing them with other form of internal sensors. For this paper, a computer vision (CV) system was developed to monitor the bending angle of a pangolin's tail robot. The CV monitors the movement of the tail in real time and processes video data. The work done in this paper can hopefully be extended so that the CV system can also work as an external sensor and part of a motor control system for the pangolin's tail robot.**

Keywords—Visual Servoing, Computer Vision, Continuum Robots

I. Introduction (*Heading 1*)

Continuum robots are often and, in our case, inspired by biology. They have many more degrees of freedom (DOF) compared to traditional robotics. Continuum robotics have a structure that is generally nonlinear and highly deformable [4] based on the environment which makes them much harder to accurately measure and control. The purpose of the project was to create a CV system that could monitor the bending angle of the pangolin's tail robot. This system could be used to calibrate the motion or measure the error of control systems. The system can also be improved by adding external sensing capabilities to serve as a basis to develop a system for eye-to-hand visual servoing. Visual servoing is an interdisciplinary field of study with robotics and CV by using vision sensor data as feedback information to control the movement of a robot [5]. Eye-to-hand visual servoing describes a CV system where the camera is external to the robot and fixed in the workspace with the entire robot and a part of its surrounding environment in view. There has been research done in this field already that proves how effective an eye-to-hand visual system can be [1],[2].

Vision sensors are versatile as a sensor as it can perform both internal and external sensing and can be less complex than other sensors, but there are weaknesses. First, an eye-to-hand vision system's view is fixed and cannot move with the robot [2]. To add more adaptability, more cameras and thus extra complexity is needed. Also, CV systems are limited to where they can be applied. Adding cameras can be not practical due to constraints of physical space or weight. Also, CV cannot be used to sense how the robot can interact with its environment but there has been research done to improve on this [3],[6].

Fig. 1: The experimental setup with the camera at the top and the continuum robot on the table over a white tablecloth.

II. Experimental Setup

The design of the experiment had three major components: the camera, the continuum robot, and the Arduino. The camera was fixed about 3 feet above the continuum robot looking downwards while the robot was placed onto the table below the camera as shown in Figure 1. This view allowed the camera to see the entire robot's workspace while restricting the robot's movement to only two-dimensional space. Restricting the movement to two dimensions made the experiment much easier to perform and hopefully can be expanded on in the future. Also, the robot was placed over a white background to reduce the amount of visual noise that is present in the normal conditions of the lab. To aid in finding the angle of the robot, two green colored stickers were placed on the top and bottom of the first and last bone of the continuum robot tail. For our purposes, most solid-colored backgrounds would also work if the color were not like the color of the stickers. The CV system was connected to an Arduino to communicate angle information. Unfortunately, the project did not get far enough so that the CV system could help in driving the motor controls, but the communication was successful and timely. The camera that was used can run at 60 frames per second, but the program was only set to run calculations once every 30 frames. This limitation was made

early on when the Arduino communication was not very well settled and the code was less optimized, so it could likely run faster especially if the CV system is run on better hardware.

III. Image Processing Methodology

After all the physical components of the project is in the right location, the camera is set to run and will perform image processing on the images captured by the live video feed of the robot and its current position. All the code for the CV system is done in Python with the OpenCV library. The methodology will be in the order below.

A. Color Detection

When a frame is captured, the image is transformed from blue, green, red (BGR) color space to hue, saturation, value (HSV) color space. BGR images are often transformed into HSV images as it is yields better results for color detection and a couple other CV algorithms. Unlike RGB, HSV separates the image intensity, from the color information. After the image is transformed into HSV color space, a mask is applied to the HSV image to find the green stickers. The green stickers can be seen on the first and last segments of the robot as seen in Figure 2. The mask image is an image that is only black and white, so it has very strong edges around the most important parts (the colored stickers) and only has edges around the stickers.

B. Contouring the Image

After the masked image is generated, the image is then contoured. Contours are the outlines representing the shape or forms of distinct features in an image. Contouring relies on using edge detection algorithms to find large changes in the image data. In this case, contours easily found edges in the masked images. In the OpenCV library, the findContours() function was very useful because it takes the two-dimensional image data and creates a three-dimensional array of the contours and their respective x and y locations. The three-dimensional array is helpful in setting up the centroid calculations. The contours are by default created in order from top to bottom.

C. Finding the Centroids

The centroids of each contour are calculated to find specific points along the bones. To calculate the centroids the following approach was implemented:

$$M_{ij} = \Sigma_X \Sigma_Y \, x^i y^j \, I(x,y) \tag{1}$$

$$C_X = M_{10}/M_{00} \,, \;\; C_y = M_{01}/M_{00} \tag{2}$$

where M_{ij} in (1) and (2) are representative of the moments, x_i the horizontal distance in the image, y_i the vertical distance, and I(x,y) the image data. M_{ij} is found using the cv.moments() function. The centroid locations C_X and C_y are then calculated for each contour. The centroid locations give points to use to estimate the orientation of the tail. For the design of this experiment, the tail was placed horizontally across the table as seen in Figure 2. To make sure that the points are in the right order left to right, an insertion sort algorithm was used with the x-values. Unfortunately, if the end effector of the tail were to wrap around the base, the code would not work, but the code works otherwise. With our current setup, having the tail bend to that degree would not be possible and was not a problem for this experiment.

D. Computing Bending Angle

After the centroid points are calculated and put in the correct order, vectors are made to find the direction the base and end effector are pointing towards. To find the vectors a simple equation in used:

$$\overrightarrow{AB} = (x_B - x_A, \, y_B - y_A) \tag{3}$$

After the vectors are created for both the first and last bone, the angle can be calculated with the following equation:

$$\Theta = \cos^{-1}((V_1 \cdot V_2)/(|V_1||V_2|) \tag{4}$$

where the angle Θ is positive or negative depending on if the end effector has a greater or lesser y-value compared to the base and V_1 and V_2 being the vectors created from the centroid points.

E. Communicating with the Arduino

After angle is calculated, the current angle is sent to the Arduino. Serial communication between the CV system and the Arduino was set up through the PySerial library. To send data to the Arduino the angle first had to be formatted into a string before being sent. The process of creating the vectors, calculating the angle, and communicating with the Arduino was done only once every 30 frames. Also, the process was only done if 4 centroid points were found to make sure that the data being sent to the Arduino did not contain errors. The Arduino also was programmed so it would send the data that was read to make sure the Arduino was reading in values properly.

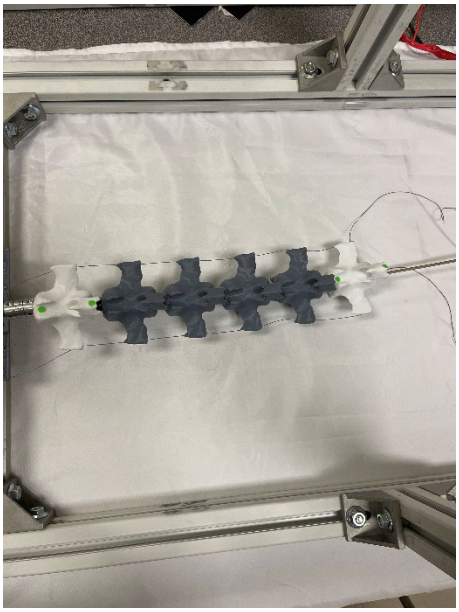

Figure 2: Example view of the robot from the camera's perspective with colored dots on the first and last segment of the robot.

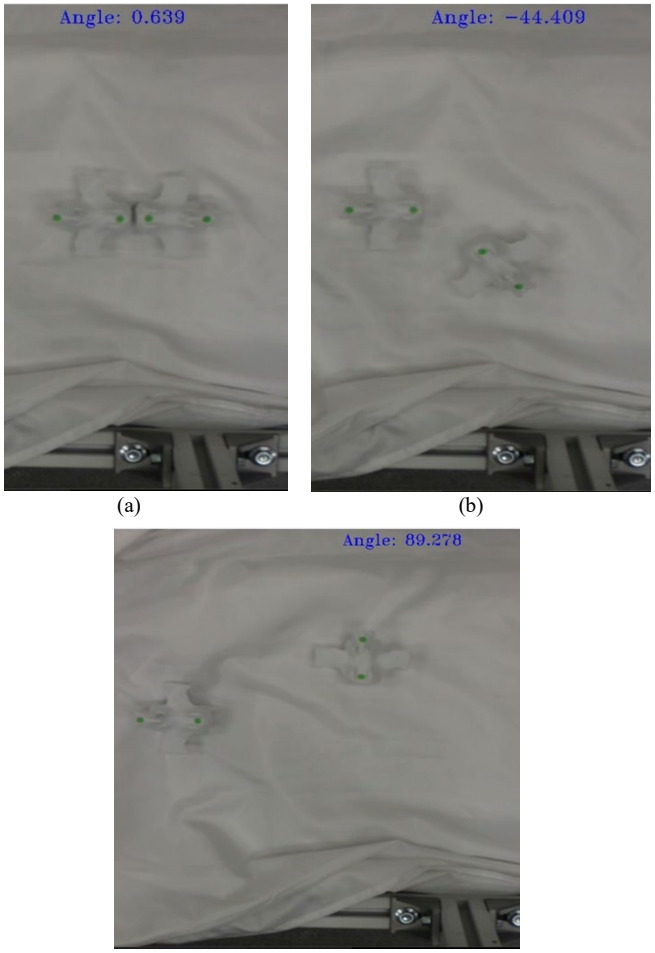

Figure 3: Angle reading with various orientations of the bones. In (a), the bones are placed close to each other to represent default 0 degrees. In (b), the bones are placed downwards with the intention of getting a –45 degree angle. In (c), the bones are placed upwards with the intention of getting a 90 degree angle.

IV. Results and Discussion

A. Analysis

The results of the experiment were good overall, and the angle measurements are seemingly accurate as seen in Figure 3. Due to time constraints, the entire tail was not ready to be tested on so real-time angle measurements were not performed on an actuated continuum robot. However, some examples of the code can be found in the figure below. Due to most of the code being run only once every 30 frames, the CV system did not update fast enough to be considered real-time but increasing the speed of the code should not be too difficult.

B. Future Work

This project was done to understand using CV systems along with continuum robotics and serves as a basis for learning and experimentation with visual servoing. In the future, the code can be integrated with a motor driver, in this case an Arduino, to provide feedback in a control system like in [2]. Also, in an input system can be developed so that a user can input a target angle for the robot to move. The CV system can serve as a calibration tool to make sure that the robot is moving precisely. Alternatively, object detection can be implemented so that the CV system can communicate with the robot tail to interact with it. To increase the accuracy of the measurements and to know the full orientation and position of the robot, the program and the robot can be modified so that the angles between each bone segment can be found. Adding this feature should help with simulating how the entire robot's workspace interacts with the environment [1]. Knowing the entire orientation also will allow the CV system to measure if there are multiple bends in the robot. Lastly, additional sensing capabilities can be added so that the bending angles can be measured in 3D space. Currently, the tail only curls about the Z-axis, so more work can be done so that a CV system can observe the robot in 3D space.

References

[1] M. Hannan and I. Walker, "Vision based shape estimation for continuum robots," *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, 2003, pp. 3449-3454 vol.3, doi: 10.1109/ROBOT.2003.1242123.J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[2] A. AlBeladi, E. Ripperger, S. Hutchinson and G. Krishnan, "Hybrid Eye-in-Hand/Eye-to-Hand Image Based Visual Servoing for Soft Continuum Arms," in *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 11298-11305, Oct. 2022, doi: 10.1109/LRA.2022.3194690.K. Elissa, "Title of paper if known," unpublished.

[3] Feng, F., Hong, W. & Xie, L. A learning-based tip contact force estimation method for tendon-driven continuum manipulator. *Sci Rep* **11**, 17482 (2021). https://doi.org/10.1038/s41598-021-97003-1Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[4] Wang X, Li Y, Kwok KW. A Survey for Machine Learning-Based Control of Continuum Robots. Front Robot AI. 2021 Sep 24;8:730330. doi: 10.3389/frobt.2021.730330. PMID: 34692777; PMCID: PMC8527450.

[5] "Visual servoing," *Wikipedia*, 21-Nov-2022. [Online]. Available: https://en.wikipedia.org/wiki/Visual_servoing. [Accessed: 08-Dec-2022].

[6] M. B. Wooten and I. D. Walker, "Environmental Interaction With Continuum Robots Exploiting Impact," in *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 10136-10143, Oct. 2022, doi: 10.1109/LRA.2022.3192771.