Vision Based 3-D Shape Reconstruction of Flexible Manipulators from Multiple Images

Guoyong Li, Zheng Li, Naveen Kumar Uppalapati

May 11, 2016

1 Introduction

Continuum flexible manipulators as shown in Fig 1have gained a lot of popularity and interest in the recent time for their inherent advantages of high maneuverability, low cost and their ability to interact safely with humans [4]. These soft manipulators have the ability to bend and also rotate simultaneously in 3-Dimensional space. They have been used in many delicate and sensitive applications like medical surgery[5], rescue and search operations [6]. These soft manipulators which are human safe can be used in applications like assistive devices and in human safe automation as shown in Fig 2. Unlike rigid robots, one major limitation for this

Figure 1: Soft flexible manipulator

continuum flexible manipulators is sensing, that is a robust method to estimate the shape of this flexible manipulators to implement real time control. There have been few sensors like strain gauges or fiber optic techniques which are developed recently, but they are highly dependent on the inaccurate models of the manipulators and therefore vision based sensing seemed promising [7].

There has been previous work on trying to use computer vision to predict the shape of these continuum flexible manipulators. Initially it started with sensing only planar continuum robots where only one camera is used to sense the shape of this planar continuum robots [9]. Then more recently it has been extended to sense the shape of a spatial continuum bending manipulator where three cameras are used to sense the shape of the spatial continuum robot [1]. Our work is mainly based on the later work where we have tried to use a similar approach to predict shape of the continuum robots which can simultaneously bend and also

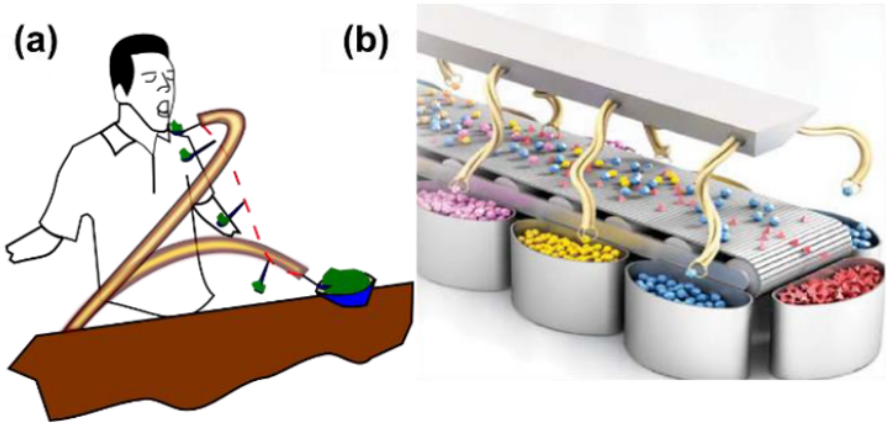

Figure 2: a) Assistive feeding device b) Human safe automation

rotate.

In our project we basically used a shape from silhouette technique to sense the shape. We have used three cameras to capture images of the manipulator from different positions. As shown in Fig 3, we start of with Camera calibration where we get the intrinsic and extrinsic parameters of the three cameras, then we extract the Silhouette from the three camera views. Then using the camera parameters we project the silhouettes to find their intersection in 3D volume, here we obtain a point cloud which is used for shape estimation and curve reconstruction in our final step to get the shape of the manipulator.

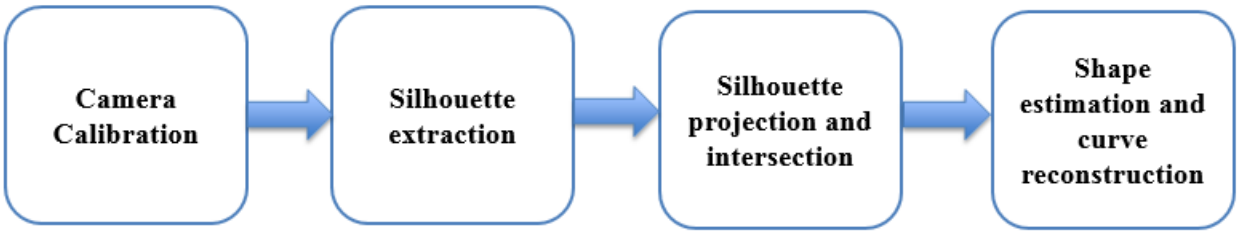

Figure 3: Shape sensing algorithm flowchart

2 Details of the approach

2.1 Camera Calibration

We have used MATLAB Single camera calibration toolbox to get the intrinsic and extrinsic parameters of the three cameras. With consideration to make it real time we have initially fixed the resolution of the three cameras to '320 $\times$ 240'. Then the steps detailed in [10] are followed to get the intrinsic parameters of the camera. Considering the three cameras are of same manufacturer we assumed the all three cameras will have same intrinsic parameters for same resolution with autofocus option disabled. Figure 5 shows that we have achieved mean error less than 0.2 pixels for our calibration parameters. Figure 4 shows the setup we

used to capture the three views of the manipulator. It can be observed from the figure that the cameras are not placed at 90 degrees to each other. This is to overcome the problem of fixing a common world center for three cameras. With MATLAB single camera calibration toolbox we cannot fix a common world center to all three cameras when they are placed at 90 degrees to each other (the checkerboard pattern is not visible to all cameras). When we have changed the positions of cameras from 90 degrees to each other, we are compromising on accuracy but we are able to fix a common world center as shown in Fig 6. Once the worldcenter is fixed we get the extrinsic parameters for the three cameras using MATLAB inbuilt function extrinsics. Once we have the camera parameters we calculate the camera matrix for the three cameras.

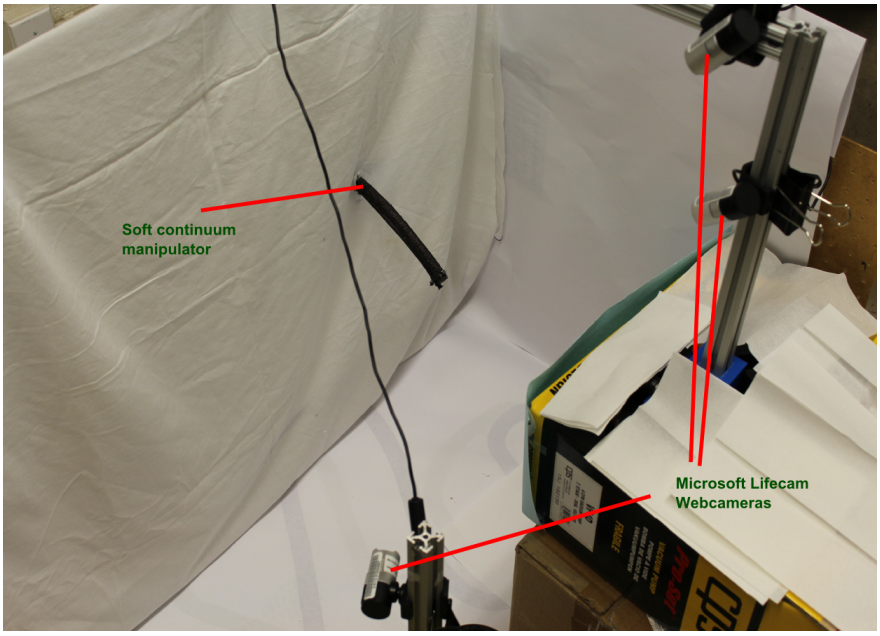

Figure 4: Setup to capture images of the manipulator from three views

This process is done only once for a fixed setup.

2.2 Silhouette extraction

For extracting the silhouettes from the images of each camera we have used a threshold to remove the background. Figure 7 shows the three images and their corresponding silhouettes after the pass through a threshold.

2.3 Point cloud from Silhouette extraction

We initially determine a cube that can contain the whole workspace of the manipulator. Then we set a corner point as original point of real world coordinate, and set X, Y and Z axis along the sides of cube. The box is 256mm $\times$ 256mm $\times$ 256mm. We set 2mm as one unit. Second, we divide the cube that contains the space that the pipe might occur into $128 \times 128 \times 128$ small cubes.
There are two ways to calculate the point cloud, the first method is by projecting each point from camera views into multi-cube and second method is to project each cube from the space that the pipe might occur into camera views. Both the solutions have same result, but different time cost. For the first solution, the time complexity is $O(L \times W \times 128)$($L \times W$ is the range of one picture). For the second solution, the time complexity is $O(128 \times 128 \times 128)$. Since the range of one picture is bigger than 128×128, so the solution two is implemented.

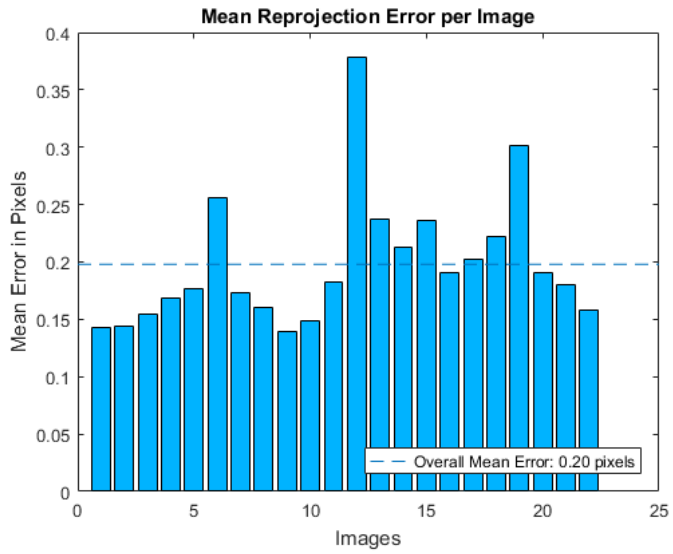

Figure 5: Mean Re-projection Error per Image

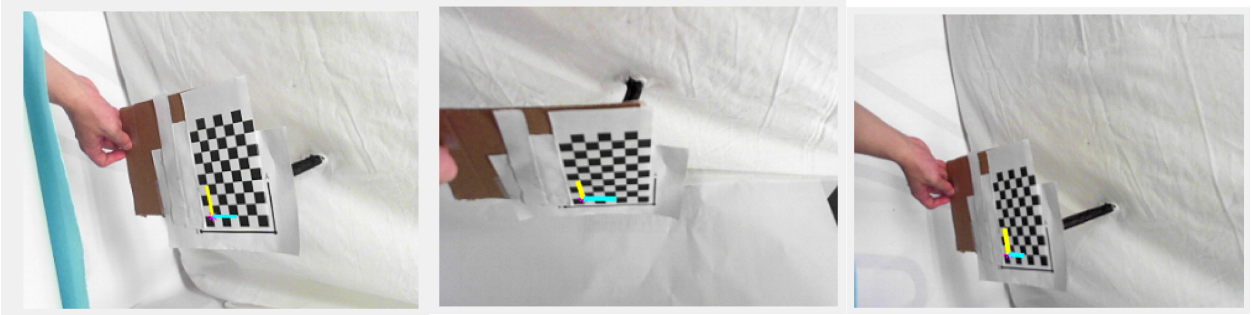

Figure 6: World center for three cameras with x axis in yellow and y axis in cyan

Since the pipe is thin and long, most of the space in $128 \times 128 \times 128$ cube should be empty, So we don't need to check every cube while getting the point cloud. Instead, we divide the entire cube space into $8 \times 8 \times 8$ big cubes. Before drill down each small cube, the bigger cube which contains the small cube need to be checked first. Because the length of pipe is around 256 mm, and we don't warp the pipe very much, it should cross more than one $32mm \times 32mm \times 32mm$ big cube. If a big cube is crossed by the pipe, some of the small cubes have 3D points. So that there are corresponding 2D points on all of the camera views. So, we can check the small cubes on surface of big cube. If all of the small cubes on surface of big cube don't have 3D point, it means the pipe doesn't cross the big cube. Since the pipe can't fit into one big cube, the big cube doesn't contain any pipe part at all. So, we don't need to check any small cube of the big cube. It should makes the algorithm reduce 1-6/16 of all time. The reason why we only check one level big cube but not check recursively is that if we check from a bigger cube (example $64mm \times 64mm \times 64mm$), the pipe might doesn't cross any bigger cube, so we might miss the whole pipe. If we check from a smaller cube (example $16mm \times 16mm \times 16mm$), the reduce time isn't obvious (1/4 of all time). For now, it takes 2 second to get one point cloud.

Figure 7: Three camera images and their corresponding Silhouettes of a frame of original data

2.4 Shape estimation and curve reconstruction

The shape estimation and reconstruction consists reordering the point cloud to represent the shape of the object and reconstruct 3D curve by fitting each axis using polynomial function.

The first thing to do once point cloud is obtained from points extraction, we estimate a center value of the point cloud by performing a 3D sphere fitting with points cloud so that sphere center point can be used as reference to define vectors in such way that each point in points cloud is unique represented by a vector. The point cloud is then re-ordered based on the angle between each vector and a reference vector (the first start vector).

The Fig8 shown below represents reconstruction process using synthetic data a) the point cloud, b) reordering point vector; c) polynomial fitting and d) 3D curve reconstruction results. Fig 8 b) shows the center point estimated by fitting a 3D sphere and how to reordering the data points according to the angle between each vector and the first start vector.

A polynomial estimation can be implemented with ordered points between point value and index on each axis (Fig 8 c) to construct an estimated point value through a link function of each axis. Finally, 3D curve is reconstructed in 3-D space using the estimated link function for x, y, z-axis as shown in Fig 8 d).

3 Results

Before we went ahead with collecting real data we have checked our method with synthetic data and the data available in the paper [1]. The following Fig 10 shows the synthetic data which have been generated using Autodesk Inventor software and Fig11 shows the extracted silhouettes.

We have captured video of the manipulator for 30 seconds at 30 frames/sec and reconstructed the 3D shape of the manipulator from the three videos. It took us 4 hours to reconstruct 900 frames which is 16 seconds for reconstructing one frame. The final reconstructed video along with the three videos of each camera can be viewed in the following link `https://uofi.box.com/v/CS543FinalProjectResults`. A snapshot of three frames from the camera and the reconstructed manipulator is shown in Fig12.

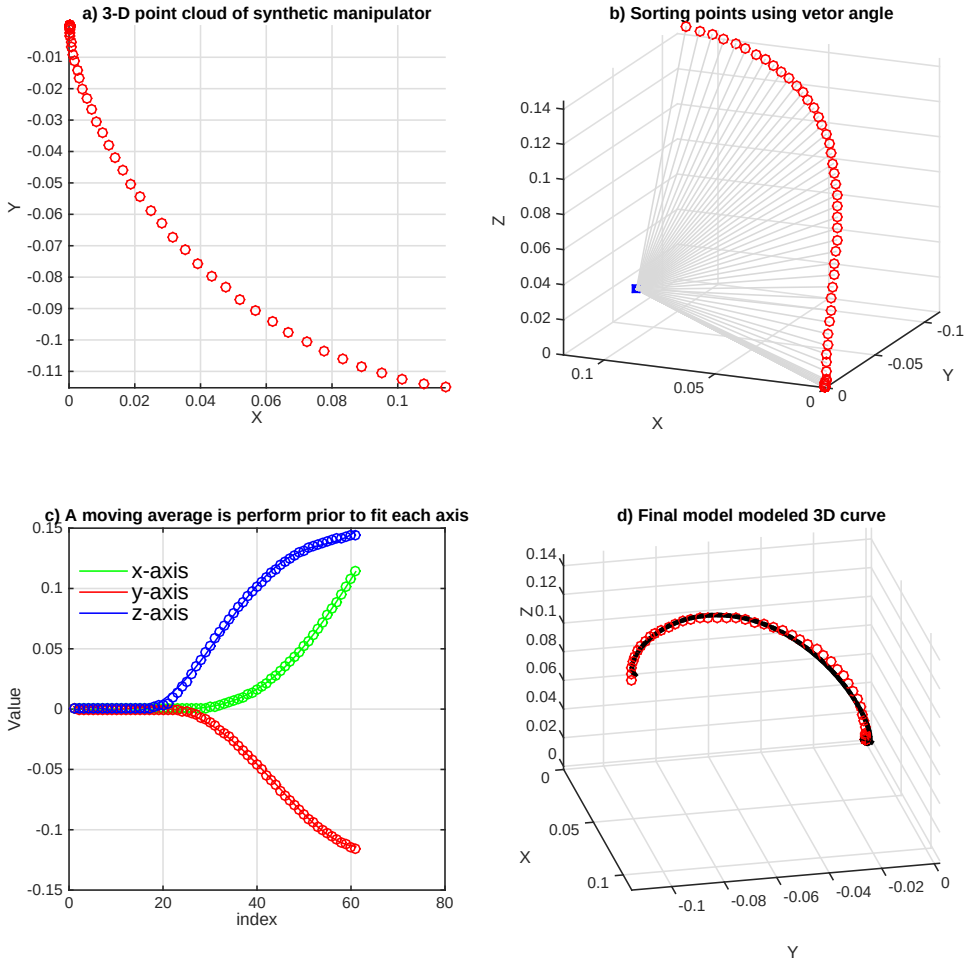

Figure 8: 3 steps to achieve the final model from obtained point cloud

4 Discussion and conclusions

Error analysis

The shape estimation using real data is presented in Fig 9. According to Fig 9, the reconstructed curve is able to represent the spatial dimension and orientation of manipulator. While implanting polynomial fitting, the reconstructed 3D curve is sensitive to the order selected. Given lower order (order = 1), the algorithm can only fit a certain shape (linear line for order equal to 1), which will not fully simulate the movement of manipulator. However, high order estimation allows more complexities but it is highly (order >3) possible for algorithm to over fit data and significantly increases the computational cost. Order 2 is therefore chosen to be reasonable value since the movement of 3D manipulator will have some curvature movement but not too complex which could be fully captured at the order of 2. It is worth noticing that this algorithm may fail to reconstruct 3D curve if a manipulator have a circle movement because it will have two points value with one given axis index and fails to construct link function.

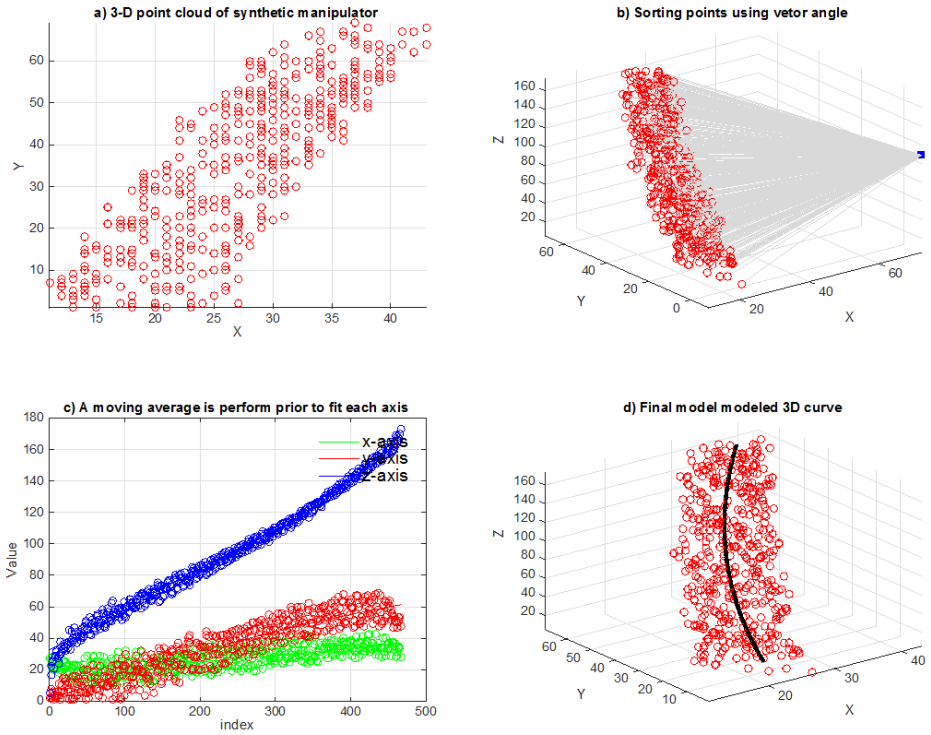

Figure 9: Shape estimation from one frame of real time data

A few improvements which will take care of the vibrations of the reconstructed curve seen in the video are to limit the voxel search to a bigger size which would give lesser points in the 3D cloud obtained after step 2, which can be used to a better robust shape fitting to the manipulator. From the video we can also observe that there is some small error in real manipulator shape and the reconstructed curve, one reason is that the manipulator diameter is larger than the reconstructed shape diameter and another issue is with the order of fitting. Considering the scope of this project, the results obtained are satisfactory and we expect we will be able to improve the speed by optimizing the algorithm and accuracy by more accurate calibration and better placement of cameras to capture the views.
In this project we have implemented an approach presented in [1] to a soft flexible manipulator which can bend and also inherently rotate. We are unable to achieve our maximum goal of implementing in real time due to limitations on time, but we have achieved satisfying results on other goals we have aimed for at the beginning of this work.

5 Statement of individual contributions

- Guoyong Li - Point cloud from Silhouette extraction, Algorithm improvement, Presentation and Final report

- Zheng Li - Shape estimation and curve reconstruction, Presentation and Final Report

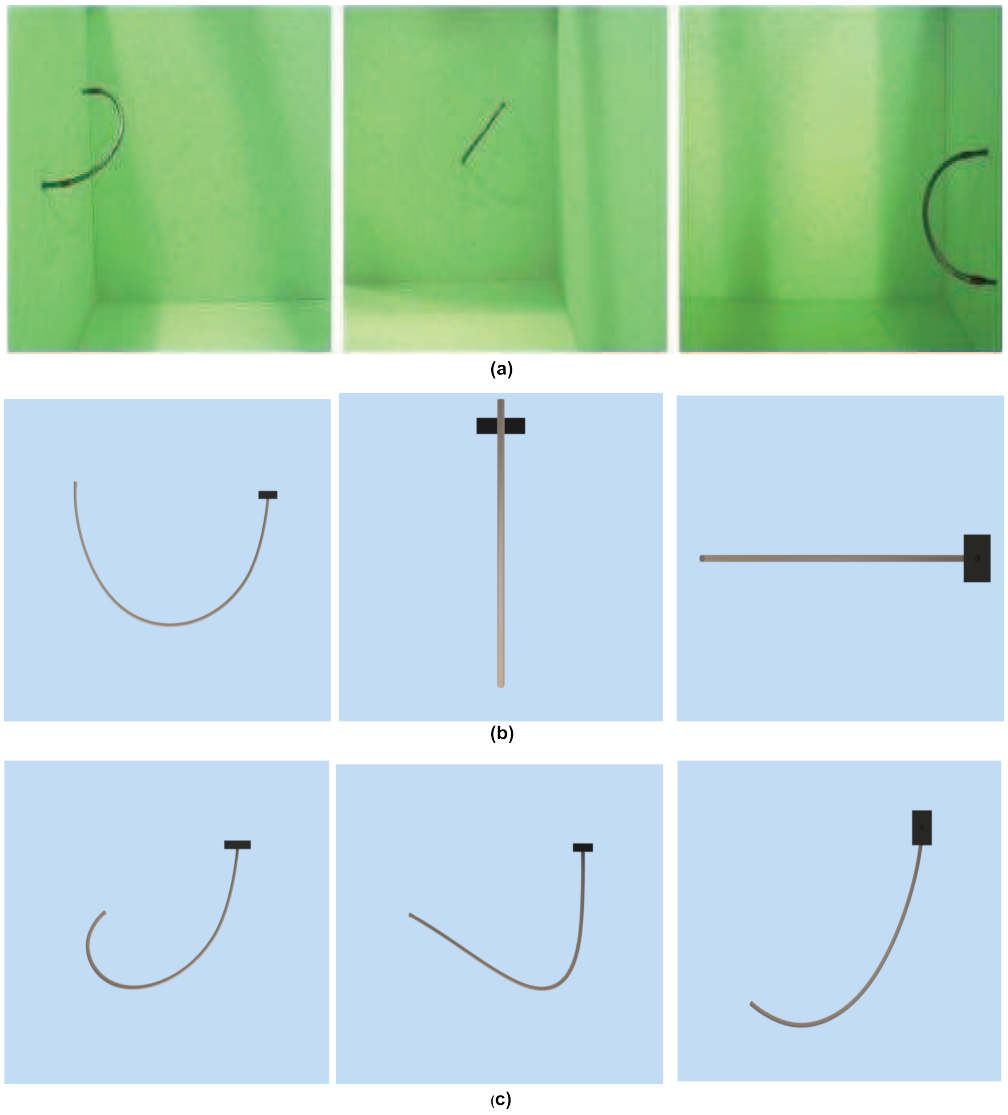

Figure 10: Data from (a)paper (b) and (c) are synthetic data

- Naveen Kumar Uppalapati - Synthetic data, Camera calibration, silhouette extraction, Presentation and Final report.

References

[1] Camarillo, David B., Kevin E. Loewke, Christopher R. Carlson, and J. Kenneth Salisbury. *Vision based 3-D shape sensing of flexible manipulators.* In Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on, pp. 2940-2947. IEEE, 2008.

[2] Mai, Fei, and Y. S. Hung. *3D Curves Reconstruction from Multiple Images.* In Digital Image Computing: Techniques and Applications (DICTA), 2010 International Conference on, pp. 462-467. IEEE, 2010.

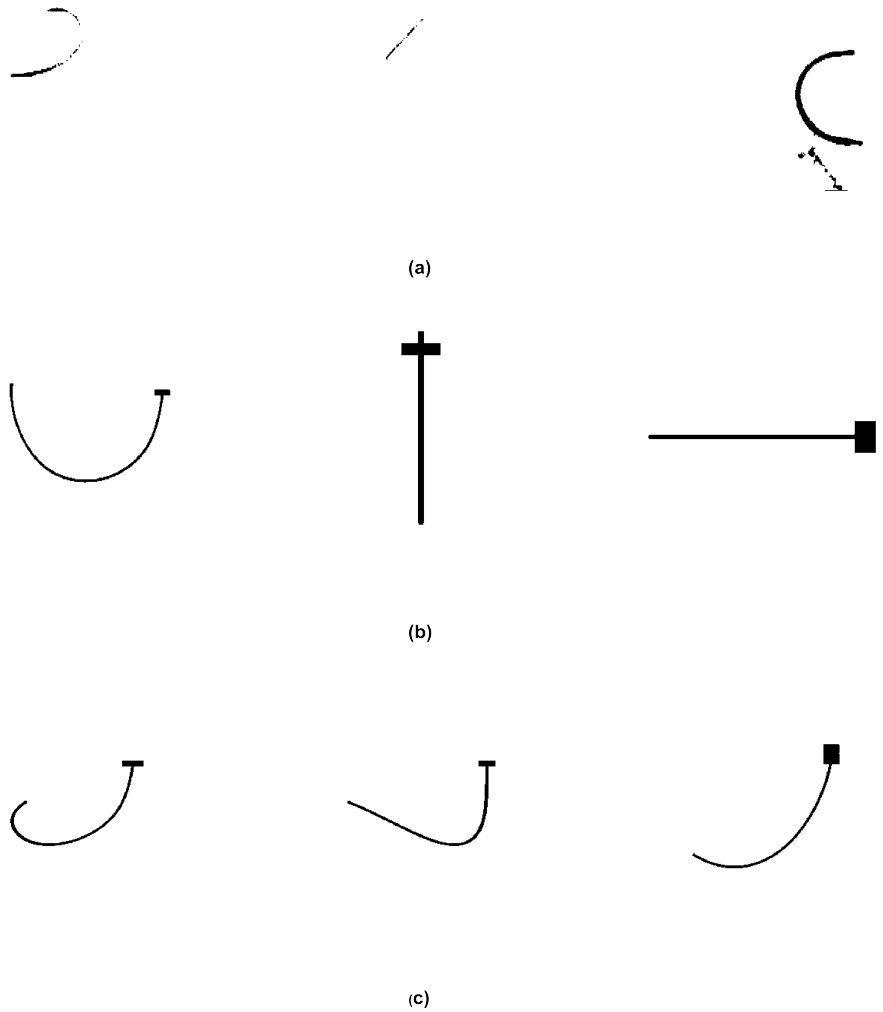

Figure 11: Extracted silhouettes

[3] Krishnan, Girish, Joshua Bishop-Moser, Charles Kim, and Sridhar Kota. "Kinematics of a Generalized Class of Pneumatic Artificial Muscles." Journal of Mechanisms and Robotics 7, no. 4 (2015): 041014.

[4] Trivedi, Deepak, Christopher D. Rahn, William M. Kier, and Ian D. Walker. "Soft robotics: Biological inspiration, state of the art, and future research." Applied Bionics and Biomechanics 5, no. 3 (2008): 99-117.

[5] Burgner-Kahrs, Jessica, D. Caleb Rucker, and Howie Choset. "Continuum Robots for Medical Applications: A Survey." Robotics, IEEE Transactions on 31, no. 6 (2015): 1261-1280

[6] McMahan, William, V. Chitrakaran, M. Csencsits, D. Dawson, Ian D. Walker, Bryan A. Jones, M. Pritts, D. Dienno, M. Grissom, and Christopher D. Rahn. "Field trials and testing of the OctArm continuum manipulator." In Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on, pp. 2336-2341. IEEE, 2006.

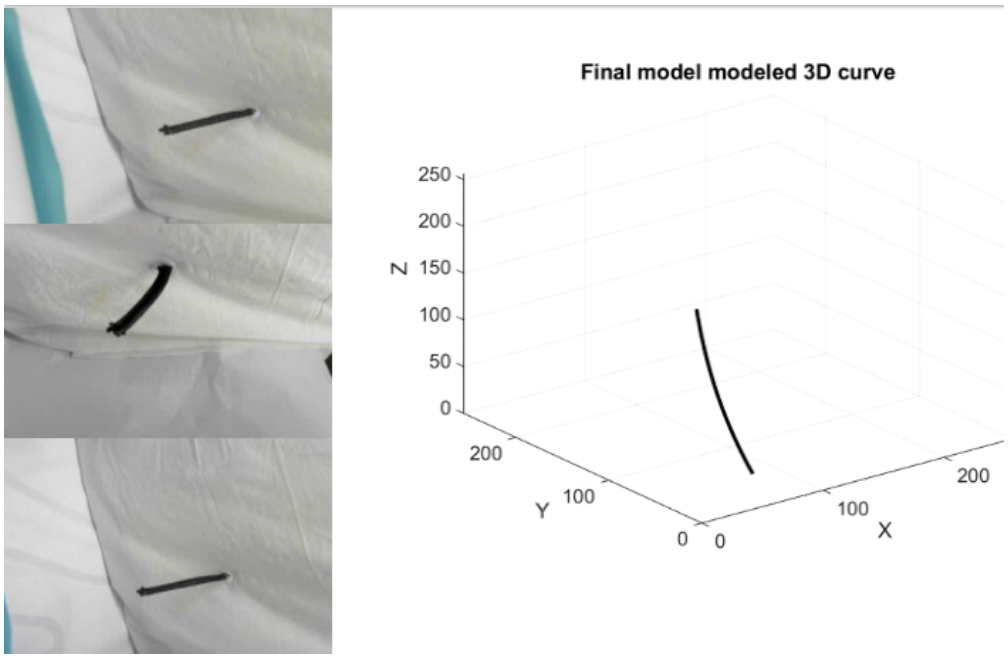

Figure 12: Three images of the camera on left and reconstructed shape on right

[7] Webster, Robert J., and Bryan A. Jones. "Design and kinematic modeling of constant curvature continuum robots: A review." The International Journal of Robotics Research (2010).

[8] Peng, Keju, Xin Chen, Dongxiang Zhou, and Yunhui Liu. "3D reconstruction based on SIFT and Harris feature points." In Robotics and Biomimetics (ROBIO), 2009 IEEE International Conference on, pp. 960-964. IEEE, 2009.

[9] Chitrakaran, Vilas K., Aman Behal, Darran M. Dawson, and Ian D. Walker. "Setpoint regulation of continuum robots using a fixed camera." In American Control Conference, 2004. Proceedings of the 2004, vol. 2, pp. 1504-1509. IEEE, 2004.

[10] http://www.mathworks.com/help/vision/ug/single-camera-calibrator-app.html