Robotica (2005) volume 23, pp. 645–651. © 2005 Cambridge University Press
doi:10.1017/S0263574704001018 Printed in the United Kingdom

Real-time shape estimation for continuum robots using vision

M. W. Hannan† and I. D. Walker‡

(Received in Final Form: August 31, 2004)

SUMMARY

This paper describes external camera-based shape estimation for continuum robots. Continuum robots have a continuous backbone made of sections which bend to produce changes of configuration. A major difficulty with continuum robots is the determination of the robot's shape, as there are no discrete joints. This paper presents a method for shape determination based on machine vision. Using an engineered environment and image processing from a high speed camera, shape determination of a continuum robot is achieved. Experimental results showing the effectiveness of the technique on our Elephant's Trunk Manipulator are presented.

KEYWORDS: Continuum robots; Vision; Shape estimation.

I. INTRODUCTION

Conventional robot manipulators feature a finite (and small) number of serially connected rigid links and actuated joints. This is an excellent design for structured precision manipulation, but has deficiencies. In particular, conventional manipulators lack maneuverability due to limited degrees of freedom. Hence recent interest in continuum style (i.e. continuous "backbone") robots, introduced in reference [1], which enable enhanced maneuverability, and can have a large number of degrees of freedom. A variety of different types of continuum manipulators have been developed.[2–9] Continuum manipulators are often inspired by biological manipulators such as trunks and tentacles.[10–12] The key to success of these continuum style robots is the bending of individual sections of the robots throughout their backbone (unlike traditional robots where the motions occur in discrete locations at the joints). Continuum style robots can achieve shapes that could only be obtained by hyper-redundant conventional robots, i.e. conventionally designed robots with many more degrees of freedom.[13]

However, a key practical problem with continuum style robots is in sensing the manipulator's configuration, i.e. shape. Presently, there are no straightforward solutions for gaining accurate feedback of these robot's shape. Note that there are not obvious 'joints' at which to place sensors. This practical limitation currently severely limits real-time implementation and development of continuum robots. A typical approximation method for shape estimation is that used previously by our group for the Elephant's Trunk at Clemson, deriving the shape based on a complex kinematic model and measurements of the change in length of its cable-actuated tendons.[14,15] This approach produces reasonable estimates, however during environmental contact shape changes not detectable by tendon length changes occur, and the estimation scheme fails. One appealing direction to look for a solution is through the use of image processing. Here we present a novel but effective solution using this strategy. The robot is located in an image from an externally located camera, and the shape of the robot is then computed. The key difficulty in this apparently straightforward approach is that the controller requires shape updates in real-time, and this imposes significant difficulties in implementation. The main contribution of this paper is to show, for the particular hardware example of an elephant's trunk manipulator at Clemson University [15, 16] how machine vision can be used for practical real-time shape estimation of continuum robot manipulators.

† Agricultural and Biological Engineering, University of Florida, Gainesville, FL 32611 (USA). E-mail: mwhannan@ufl.edu
‡ Electrical and Computer Engineering, Clemson University, Clemson, SC 29634 (USA). E-mail: ianw@ces.clemson.edu
Corresponding author: I. D. Walker.

II. HARDWARE TESTBED

The shape estimation work utilized the elephant's trunk manipulator hardware and its control system in the robotics laboratory at Clemson University.[16] The hardware additions used for the work reported in this paper comprised three main components: a Dalsa CCD high speed camera, a BitFlow Road Runner frame grabber, and an AMD 1300 MHz based PC. The Dalsa camera was selected due to its maximum frame rate of 955 frames per second. The camera was fixed at a distance of 3.05 m (10 ft) from the robot, see Figure 1, and thus giving a relatively wide view of the robot's workspace. To help facilitate real time performance the environment in the field of view of the camera was designed so that the desired regions of the robot could easily be located. This was done by making all non-important areas as bright as possible, and all the desired areas as dark as possible, see Figure 1. This significantly reduced the amount of image processing that needed to be done, and thus decreases the overall time needed for shape estimation (note that ideally the shape estimation would be completed in real-time).

III. IMAGE PROCESSING

The first main step in the approach is to capture useful information from the camera images. Here the key is to match the impage processing to the physical constraints of the robot. The Elephant's Trunk Manipulator is composed of four main sections. Each section can bend (pulled by tendons terminating at the section end) as demonstrated in Figure 2.

https://doi.org/10.1017/S0263574704001018 Published online by Cambridge University Press

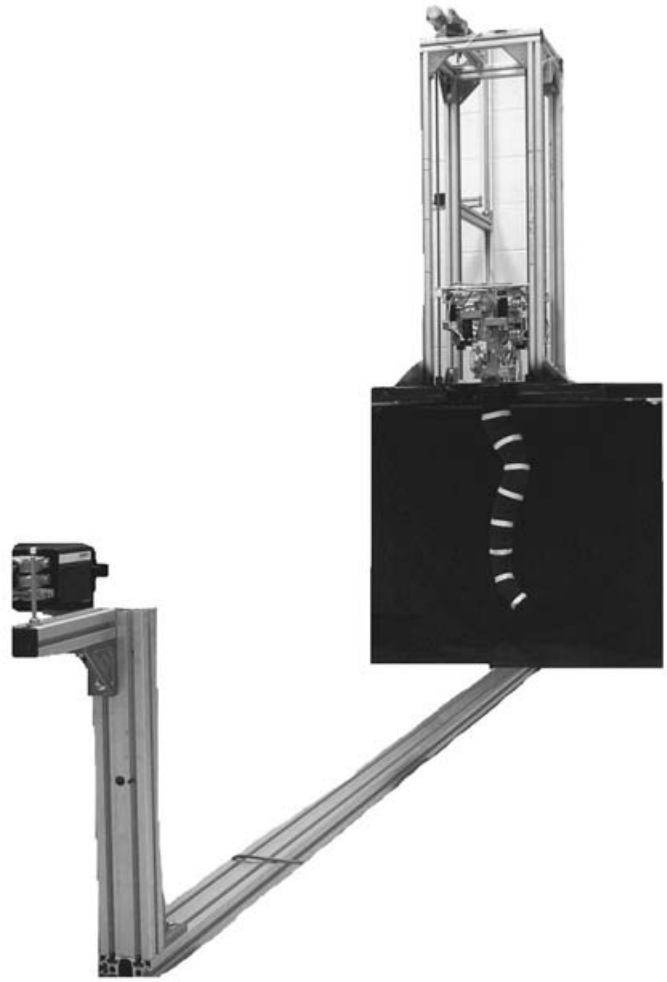


Fig. 1. Testbed configuration.

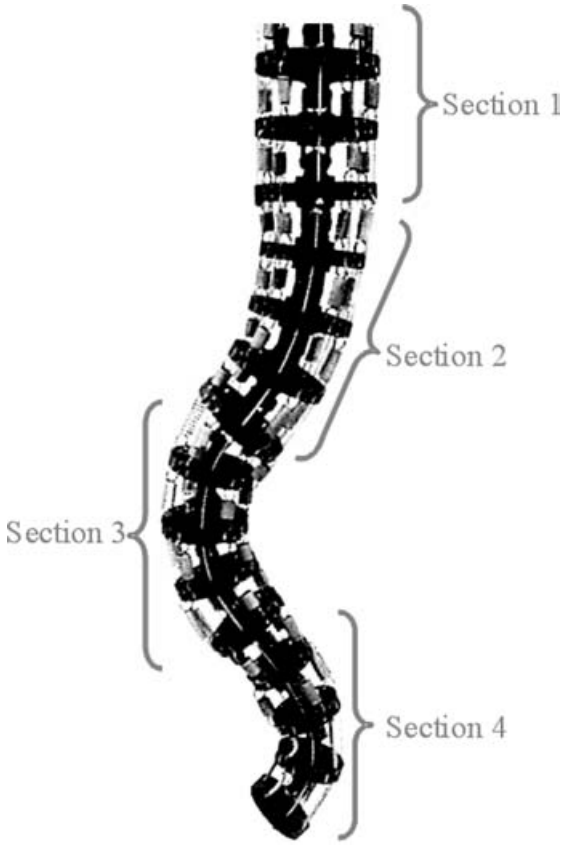


Fig. 2. Elephant's Trunk robotic manipulator.

In this work we only consider planar motion of the robot. The nature of the design constrains the robot to bend into four sections of approximately constant curvature. Thus, the kinematics model the planar robot as a serial connection of four curves with constant curvature.[15] The objective of this work is to determine a curvature for each section, and thus determine the shape of the robot.

The fundamental principle exploited to accomplish the above strategy is that determination of the curvature for a constant curvature arc requires a minimum of three points.[17] These three points completely define the entire curve. The most familiar example of a curve with constant curvature is a circle. Since the kinematic model of the Elephant's Trunk is based on constant curvature sections, each of the sections can be expressed geometrically as part of a circle. Therefore, the strategy chosen is to fit a circle to each of the sections. If three points can be determined from an image that represent the shape of one section, then a circle, and thus the curvature, can be found for the section. The points that yield the most accurate results for each section are the ones at each end of the section, and one point approximately in the center of the section, where all three points should be located on the center line of the section. This is the strategy underlying our approach.

The next step taken was to engineer the system to facilitate the identification of the required three points on each of the circles. The algorithm used is based on finding the center of each of three bands located on each section, see Figure 3. One band is located at the beginning of a section, the next is

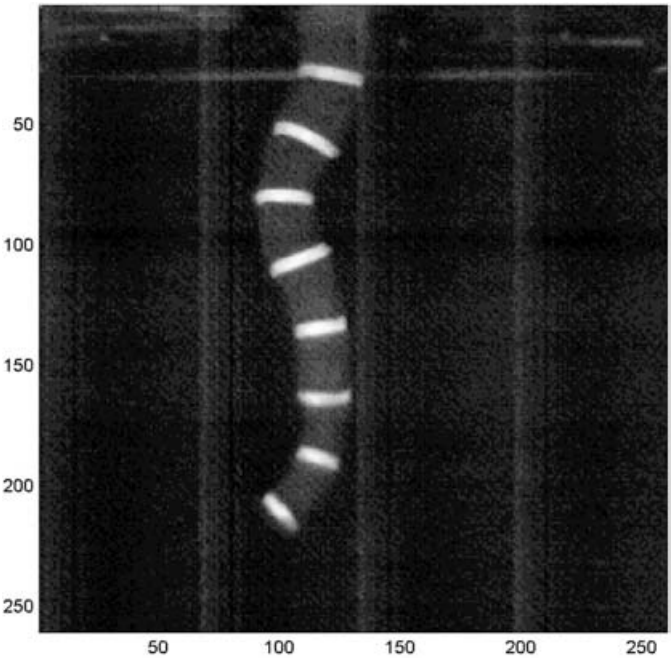


Fig. 3. Image captured by the camera.

https://doi.org/10.1017/S0263574704001018 Published online by Cambridge University Press

located in the middle, and the third is located at the end of the section. Standard image processing techniques are used to locate each band, and then find its center. This gives the three proper points need to define the circle, and thus the curvature for the section. This process is applied to all four sections of the robot, where the last band of one section is the first band of the next section. Since there are four sections, a total of nine bands are needed to find all the points. However, the first band on the base section of the robot is fixed due to the design of the robot. Therefore, no band is needed for the first point of the first section since it is a fixed point, and hence in Figure 1 only eight bands are visible. In the next section the complete details involved in processing a single frame of this strategy is presented. The process is repeated as the robot configuration evolves to generate the time-varying shape estimates.

III.1. Initial processing

Once an image is captured by the camera it needs to be processed such that the shape of the robot can be determined A typical image captured from the camera can be seen in Figure 3. Note, the image in Figure 3 was brightened by an intensity of 15 to make it viewable.

The first step is to use thresholding to determine which pixels in the image are of value. (Thresholding an image sets all the pixels greater than a given intensity to one intensity value of intensity, and any pixels with a value less than the given value to another value). To determine which value of intensity should be used in thresholding, the histogram of the image was used. This histogram is shown in Figure 4. Notice that from the histogram all of the pixel intensities are in the range from 5 to 25. Thus, the threshold value was initially picked in this range.

After some tuning, a threshold value of 12 was determined to yield the best results in separating useful pixels (bands) from the none useful ones (background). To threshold the image all the pixels were set to either 0 or 256. Those pixels with intensities above 12 had their intensity set to one level, and all the pixels with intensities less than 12 were set to the other. A resulting image after thresholding is shown in Figure 5.

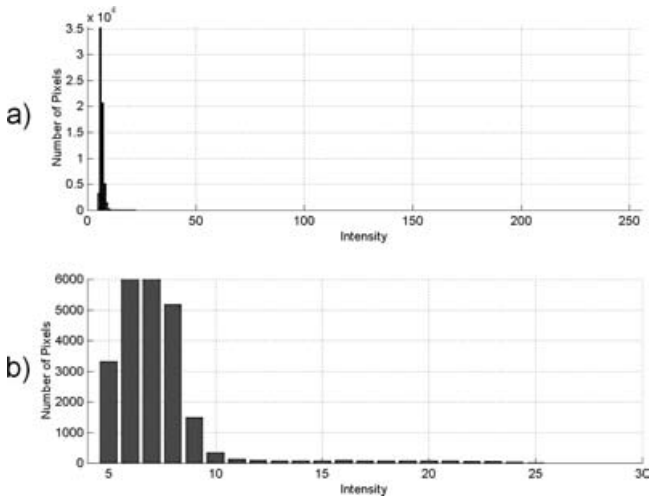


Fig. 4. Histogram of the captured image: a) results for all 256 levels of intensity, b) closeup examination of important pixel intensities.

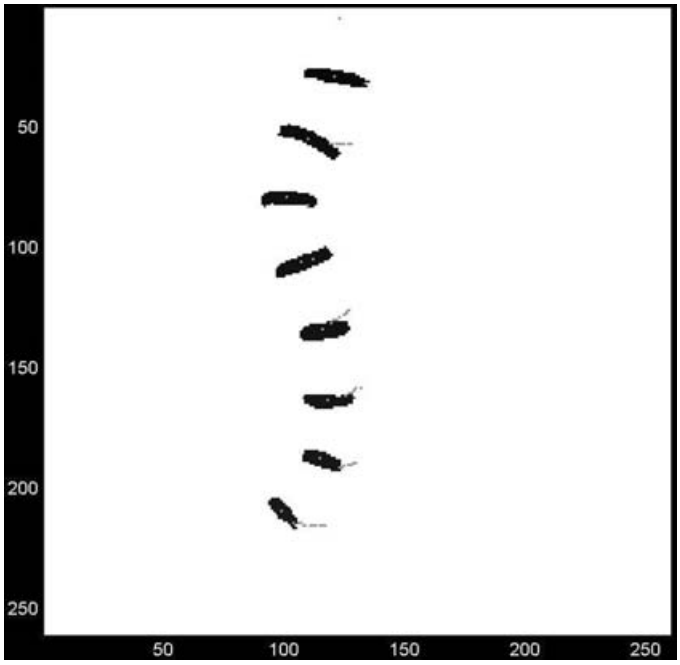


Fig. 5. Thresholded image.

Following the thresholding to identify the interesting pixels, the pixels must be sorted in such a way that they can be defined into logical regions. These regions can be defined using segmentation. (In segmenting the image an initial "seed" pixel is defined, and all the pixels that have connectivity with this seed pixel are defined as a region). The seed pixel simply starts out at the first pixel in the image, and after a region is "grown" around this seed pixel the location of the pixel is moved. Normally this seed pixel is simply shifted by one pixel from the previous location, but for efficiency in this work the seed point was shifted by 12 pixels every time instead of just one. This can be done since the bands occupy a large number of pixels, and thus only one out of these many pixels needs to be found as a seed for the region to be properly determined.

In segmenting the image in Figure 5, many different regions were found. Since only the regions that represent the eight bands are desired, a further step was required to filter out the unwanted regions. This was accomplished by simply filtering out any regions that did not have an area between 50 and 175 pixels.

The resulting segmented and filtered image is shown in Figure 6. Each region has its own pixel intensity, and this intensity is defined as the label for that region. It can be seen from Figure 6 that all eight desired regions were found. Each region was uniquely labeled by its color. The next step, following the identification of the eight desired regions, is to determine the four curvatures. This process is described in the following section.

III.2. Determination of curvature

The first stage in curvature determination is the determination of the curvature of the center point of each section. This was

https://doi.org/10.1017/S0263574704001018 Published online by Cambridge University Press

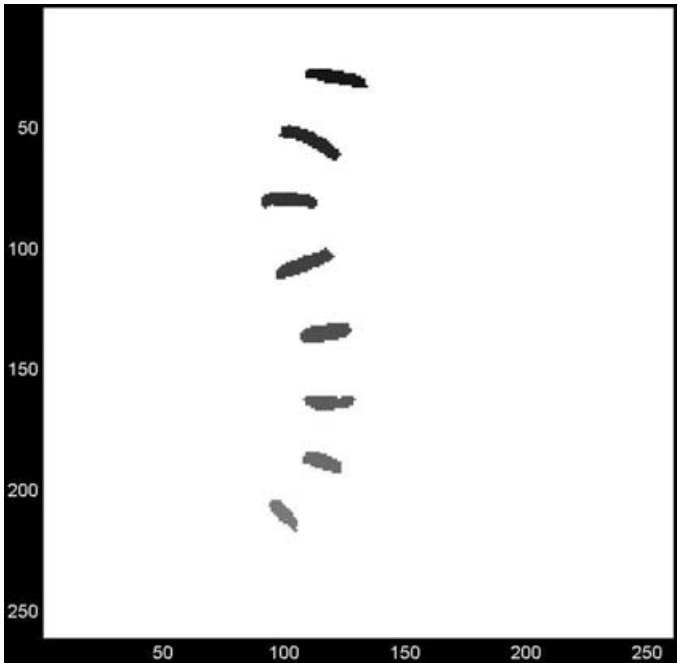


Fig. 6. Segmented image.

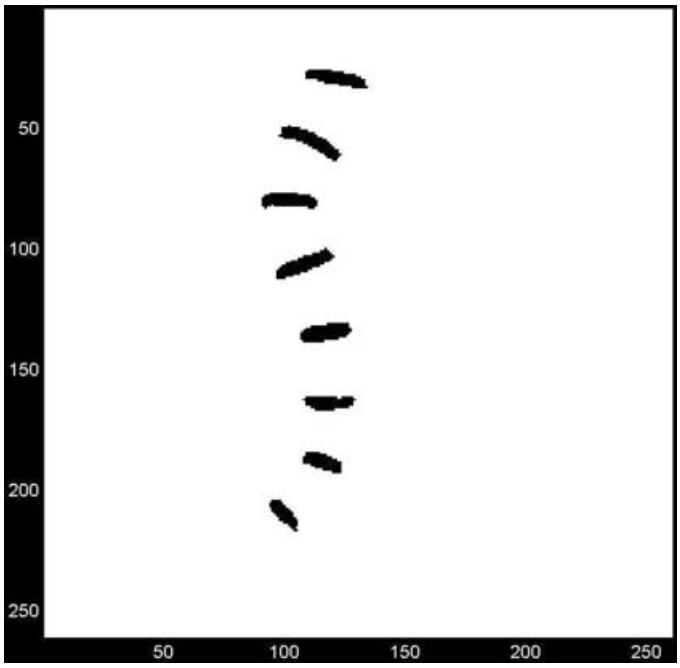


Fig. 7. Centroids of the regions.

accomplished by determining the centroid (center of mass) of each region.

As presented in reference [18] the centroid of each region given by its pixel coordinates of x_c and y_c can be calculated using the following:

$$m_{pq} = \sum_{-\infty}^{\infty} \sum_{-\infty}^{\infty} i^p j^q f(i, j) \tag{1}$$

$$x_c = \frac{m_{10}}{m_{00}} \qquad y_c = \frac{m_{01}}{m_{00}}, \tag{2}$$

where i and j are the pixel coordinates and $f(i, j)$ is the value of the pixel located at (i, j). The resulting centroids for each band can be seen as single gray pixels in the regions shown in Figure 7.

Simply finding the centroids is not sufficient however. Even though the centroids of each region have now been identified, the individual section curvatures cannot yet be directly calculated. The problem is that the regions that have been determined from the image thus far are in no specialized order. Thus, before the robot curvatures can be determined the regions must be ordered. The robot operates such that each section can be defined as set of three regions. The first section is defined by the base location and regions one and two. The second section is defined by regions two, three, and four. The third sections is defined by regions four, five, and six. The fourth and final section is defined by regions six, seven, and eight. Using the orientation of the principal axis for each region as a reference, the next section can be found by using a simple search algorithm.

The orientation θ of the principal axis of a region is calculated as

$$m_{pq} = \sum_{-\infty}^{\infty} \sum_{-\infty}^{\infty} (i - x_c)^p (j - y_c)^q f(i, j) \tag{3}$$

$$\theta = \frac{1}{2} \arctan\left(\frac{2m_{11}}{m_{20} - m_{02}}\right). \tag{4}$$

The physical constraints of the robot can now be used to aid the search. Notice that consecutive bands are approximately parallel to each other Given this and the above principal axis for a region, its principal axis will be perpendicular to the general direction of the next region to be found. Thus the search algorithm uses the normal to the principal axis of the present region to determine where to look for the next region. The proper direction (± 90 degrees) of the vector can be determined by comparing the normal vector's orientation to that of the orientation of the vector linking the present centroid's location to the previous centroid's location.

Once the proper direction to search has been determined, the algorithm then searches in an arc with a radius of 29 pixels over a range of 0.9 radians that is symmetric about the normal. The algorithm moves along the arc checking the pixel label at each iteration. If the label is that of the background, the search continues. However, if the pixel is not labeled as background, then the pixel label is checked against the list of labels for the regions. The region the label belongs to is the next region in the order. This process is repeated until all the regions are ordered.

Figure 8 shows the arcs used to find each region in the search algorithm. Note that the last ordered centroid corresponds to the end point of the manipulator, and thus directly provides the information needed for end point control (via $\mathbf{x}_d - \mathbf{x}$).

All that remains is to combine the information found in the last few steps. Given the order of the regions and their corresponding centroids, the curvature for the four sections of the robot can be easily found. The (constant) curvature of each section can be found by fitting a circle through the

https://doi.org/10.1017/S0263574704001018 Published online by Cambridge University Press

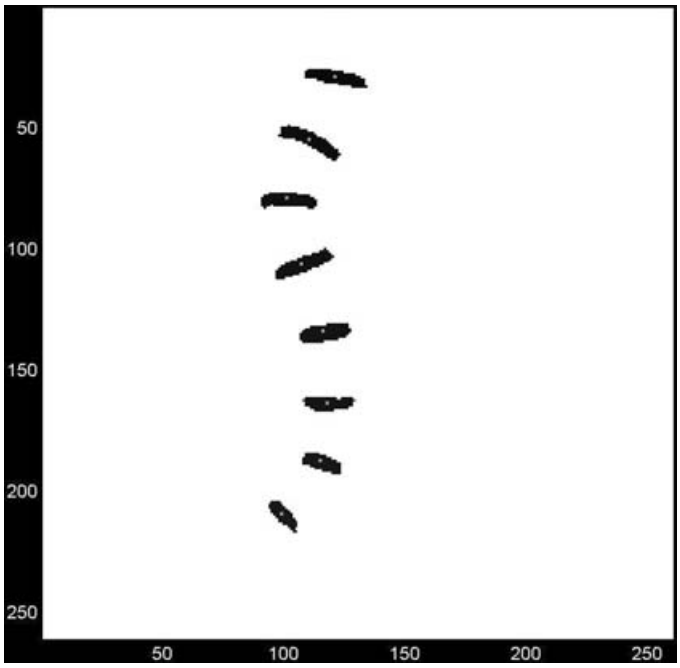


Fig. 8. Region ordering algorithm.

three centroids that correspond to each section. The general equation for a circle is

$$(x-a)^2+(y-b)^2=r^2, \tag{5}$$

where (a, b) is the location of the center of the circle in terms of (x, y) and r is the radius of the circle. Using the following substitution

$$\alpha = a^2+b^2-r^2 \tag{6}$$

(5) can be rewritten as

$$2xa+2yb-\alpha = x^2+y^2 \tag{7}$$

Since Equation (7) is a linear equation it can be rewritten in terms of the three centroid locations as

$$\begin{bmatrix} 2x_1 & 2y_1 & -1 \\ 2x_2 & 2y_2 & -1 \\ 2x_3 & 2y_3 & -1 \end{bmatrix} \begin{bmatrix} a \\ b \\ \alpha \end{bmatrix} = \begin{bmatrix} x_1^2+y_1^2 \\ x_2^2+y_2^2 \\ x_3^2+y_3^2 \end{bmatrix}, \tag{8}$$

where (x_1, y_1) is the location of the first centroid, (x_2, y_2) is the location of the second centroid, and (x_3, y_3) is the location of the third centroid. Since Equation (8) has three unknowns and three equations it can be solved for the parameters a, b, and α as

$$\begin{bmatrix} a \\ b \\ \alpha \end{bmatrix} = \begin{bmatrix} 2x_1 & 2y_1 & -1 \\ 2x_2 & 2y_2 & -1 \\ 2x_3 & 2y_3 & -1 \end{bmatrix}^{-1} \begin{bmatrix} x_1^2+y_1^2 \\ x_2^2+y_2^2 \\ x_3^2+y_3^2 \end{bmatrix}. \tag{9}$$

Once α is found, r can be found by solving Equation (6). From [17], the curvature κ for the section is simply calculated as

$$\kappa = \frac{1}{r}. \tag{10}$$

This technique is then applied to each of the sections in turn to obtain their respective curvatures, and hence the shape of the entire robot.

The computational complexity of the approach was found to be reasonable, at least for the particular problem studied here. The image processing routine was run in parallel with the image capture routine, which enabled real-time estimates. A synchronization strategy was used to avoid data corruption. The two routines were synchronized by the capture routine, which signaled the processing routine when an image had just finished being captured. If the processing routine was not ready at this moment, it then waited until a new image was ready.

Thus, with the camera running at 955 fps, the only available frame rates for the image processing were $\frac{955}{n}$, where n is a positive integer. Due to optimization of both the environment and the image processing strategy the processing routine was able to run at 477.5 fps where it took about 1.67 ms to process the image. The question of how 'real-time' this is is open to debate. However, it was sufficient for practical control of the robot, and very fast considering standard image processing routines typically run no faster than a few tens of frames per second. The frame rate is easily high enough to provide accurate and dependable information about the robot's shape to the controller.

III.3. Discussion/Results

The results of the above approach for one configuration are displayed in Figure 9. One circle was fit to each of the four sections of the elephant's trunk robot. The curvature value for sections 1 to 4 are 0.012071, 0.021203, 0.008261, and 0.019422 where the units are $\frac{1}{pixel}$.

One final issue that needs to be addressed is the sign of each curvature (which way the circle is oriented). The sign of the curvature is calculated above by comparing the direction of the principal normal of the Serret-Frenet frame at one of the centroids of one of the regions in the section to the direction of the vector from that centroid to the center of the calculated circle. The principal normal is simply the principal axis for the region, where the proper sign of the principal axis was determined during the centroid ordering algorithm. Referring to Figure 10, if the difference between the two vectors is 180 degrees than the curvature is positive, otherwise it is negative. Thus, the signed curvature values for sections 1 to 4 are $+0.012071$, -0.021203, $+0.008261$, and -0.019422. This completes the shape estimation procedure.

In order to evaluate the results of this vision-based approach to shape estimation, we next compare the results to those obtained to our 'baseline' of curvatures estimated via measuring the change in cable length directly at the actuators. These values (from cable length estimation) have units of $\frac{1}{inch}$. Since the units between the two sets of curvatures differ, the curvatures from image processing can be converted to the

https://doi.org/10.1017/S0263574704001018 Published online by Cambridge University Press

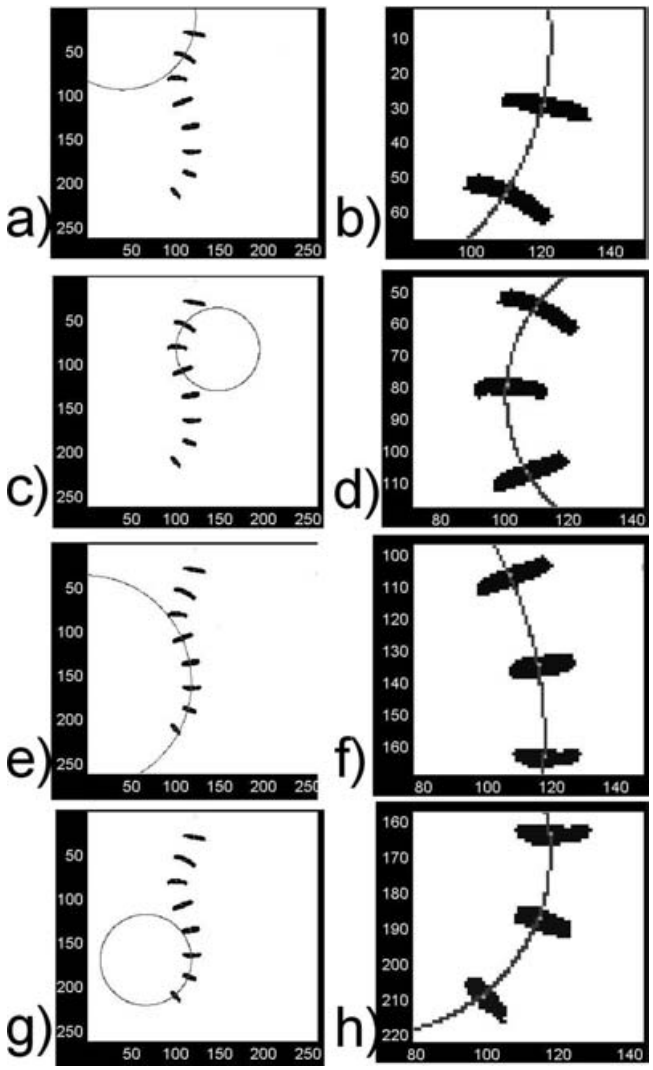


Fig. 9. Results of circle fitting routine: a) circle fit to section 1, b) closeup of Section 1, c) circle fit to Section 2, d) closeup of Section 2, e) circle fit to Section 3, f) closeup of Section 3, g) circle fit to Section 4, e) closeup of Section 4.

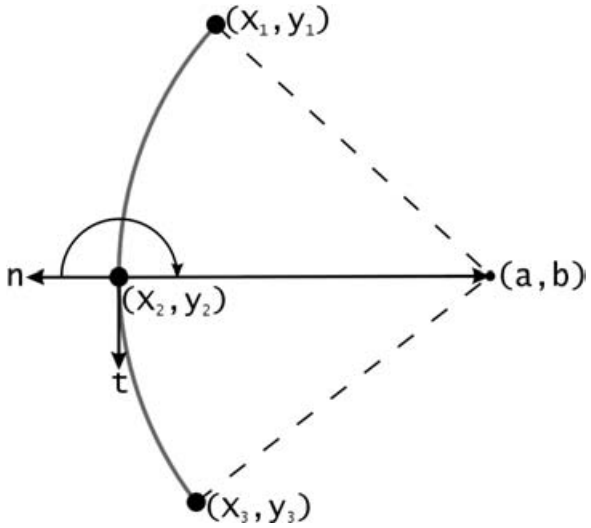


Fig. 10. Determination of the sign of the curvature.

Table I.

Section	κ_v	κ_c	Δ	%
1	+0.05541	+0.08956	−0.03415	−38.1
2	−0.1085	−0.1573	−0.0488	+31.0
3	+0.05133	+0.06130	−0.00997	−16.3
4	+0.1311	+0.1441	−0.013	−9.02

Table II.

Section	κ_v	κ_c	Δ	%
1	−0.0484	−0.0288	−0.0196	+68.1
2	−0.0448	−0.042	−0.0028	+6.67
3	+0.1068	+0.0963	+0.0105	+10.9
4	+0.1131	+0.0801	+0.033	+41.2

Table III.

Section	κ_v	κ_c	Δ	%
1	−0.1201	−0.0724	−0.0477	+65.9
2	+0.0901	+0.0685	+0.216	+31.5
3	+0.1410	+0.1087	+0.0323	+29.7
4	−0.1285	−0.131	+0.0025	−1.91

units of $\frac{1}{inch}$. The conversion was experimentally verified to be 7.42 pixels per inch. The resulting curvatures are given in Table I.

In Table I (and the subsequent Tables), κ_v is the curvature from image processing ($\frac{1}{inch}$), κ_c is the curvature derived from the change in cable length ($\frac{1}{inch}$), $\Delta = \kappa_v - \kappa_c$, and % is the percent error between κ_v and κ_c, which is calculated as $\% = 100(\frac{\Delta}{\kappa_c})$.

Measurements were made at two other configurations, with results for those cases shown in Tables II and III, respectively.

In the cases investigated in Tables I–III, the average error (magnitude) was 57.4%, 23.1%, 19.0%, and 17.4% respectively for each section. This demonstrates that there is a significant difference between the curvature measurements using the cable strategy versus the vision strategy. More analysis is necessary to better appreciate factors defining the accuracy of both the cable length based curvatures and the vision based curvatures. However, we are confident that most of the error arises from the cable measurement based approach. There are two major factors in this, as discussed in the following paragraphs.

A significant part of the errors in the cable based curvature estimates arise from coupling between sections. As a section moves, in some cases previous sections can change shape without the encoder detecting any movement. This issue arises fundamentally from the actuation strategy only using one motor and a cable tensioning system to actuate each pair of cables. When each section bends only the tensioned side of the section is being actuated by the motor, and thus only this is measured by the encoder. If a section further down the robot is moved in the same direction as the earlier section, the tensioning system can allow cable to be released on the tensioned side (and retract cable on the untensioned

https://doi.org/10.1017/S0263574704001018 Published online by Cambridge University Press

Table IV.

Section	κ_v	κ_c	Δ
1	−0.0115	0.0	−0.0115
2	0.0	0.0	0.0
3	0.0	0.0	0.0
4	0.0288	0.0	0.0288

side). Thus, the section's shape will evolve in a way that cannot be directly determined by the encoders. This is a problem inherent to tendon-based systems with feedback of only tendon measurements. Our ongoing work focuses on adapting strain guage technology to directly measure the robot shape.

Another problem that affects the results is that since the encoders were used to to determine the curvature there is no direct sensing of the initial robot's shape. A practical solution is to always start the robot with all four sections straight. This is a configuration where the curvature for each section can be easily determined without any external measuring device. The difficulty is that even when the manipulator appears straight, this may not be precisely the case. Thus, there are always inherent errors between the cable based and the vision based curvatures. The difference between the two quantities when the robot appears to be in the straight position is shown in Table IV. Note that there is a noticeable amount of error between the two different curvature measurements, even for this straight configuration.

IV. CONCLUSIONS

This paper has presented a practical vision-based method to sense the shape of continuum robots. By breaking down the manipulator section into distinct bands, and taking advantage of the geometric shapes achievable by the sections, the method uses image processing from an external camera to compute the (constant) curvature of each section. Serial combination of these curves provides the shape of the manipulator. In particular, by use of a high speed camera, and judicious processing, updates of the shape are obtained in real time. In addition, the shapes computed are shown to be significantly more accurate than those obtained by estimation from internal measurements for an Elephant's Trunk continuum robot arm.

Acknowledgements

This work was supported in part by NASA grant NAG5-9785, in part by NSF/EPSCoR grant EPS-9630167, and in part by the Defense Advanced Research Projects Agency (DARPA) through the Space and Naval Warfare Systems Center, San Diego, under Contract Number N66001-03-C-8043.

References

1. G. Robinson and J. B. C. Davies, "Continuum Robots – A State of the Art," *IEEE Conf. on Robotics and Automation* (1999) pp. 2849–2854.
2. V. C. Anderson and R. C. Horn, "Tensor Arm Manipulator Design," *ASME paper 67-DE-57*.
3. R. Buckingham and A. Graham, "Reaching the Unreachable – Snake-Arm Robots," *Proceedings of International Symposium of Robotics* (2003) pp. 1–6. Also available at: OCRobotics Ltd., http://www.ocrobotics.com.
4. G. S. Chirikjian, "Theory and Applications of Hyper-Redundant Robotic Mechanisms," *Ph.D. Thesis* (Dept. of Applied Mechanics, California Institute of Technology, 1992).
5. R. Cieslak and A. Morecki, "Elephant Trunk Type Elastic Manipulator – A Tool for Bulk and Liquid Materials Transportation," *Robotica* **17**, 11–16 (1999).
6. I. Gravagne and I. D. Walker, "On the Kinematics of Remotely-Actuated Continuum Robots," *IEEE Conf. on Robotics and Automation* (2000) pp. 2544–2550.
7. G. Immega and K. Antonelli, "The KSI Tentacle Manipulator," *IEEE Conf. on Robotics and Automation* (1995) pp. 3149–3154.
8. H. Ohno and S. Hirose, "Study on Slime Robot (Proposal of Slime Robot and Design of Slim Slime Robot)," *IEEE Conf. on Intelligent Robots and Systems* (2000) pp. 2218–2223.
9. K. Suzumori, S. Iikura and H. Tanaka, "Development of Flexible Microactuator and its Applications to Robotic Mechanisms," *IEEE Int. Conf. on Robotics and Automation* (1991) pp. 1622–1627.
10. W. M. Kier and K. K. Smith, "Tongues, Tentacles, and Trunks: The Biomechanics of Movement in Muscular Hydrostats," *Zoological Journal of the Linnean Society* **83**, 307–324 (1985).
11. K. K. Smith and W. M. Kier, "Trunks, Tongues, and Tentacles: Moving with Skeletons of Muscle," *American Scientist* **77**, 28–35 (1989).
12. S. Hirose, *Biologically Inspired Robots* (Oxford University Press, 1993).
13. H. Mochiyama, E. Shimemura and H. Kobayashi, "Shape Correspondence Between a Spatial Curve and a Manipulator with Hyper Degrees of Freedom," *IEEE Conf. on Intelligent Robots and Systems* (1998) pp. 161–166.
14. M. W. Hannan and I. D. Walker, "Novel Kinematics for Continuum Robots," *7th International Symposium on Advances in Robot Kinematics* (2000) pp. 227–238.
15. M. W. Hannan and I. D. Walker, "Analysis and Experiments with an Elephant's Trunk Robot," *International Journal of the Robotics Society of Japan* **15**, No. 8, 847–858 (2001).
16. M. W. Hannan and I. D. Walker, "Vision Based Shape Estimation for Continuum Robots," *IEEE International Conference on Robotics and Automation* (2003) pp. 3449–3454.
17. D. J. Struik, *Lectures on Classical Differential Geometry* (Addison-Wesley Publishing Company, 1961).
18. M. Sonka, V. Hlavac and R. Boyle, *Image Processing, Analysis, and Machine Vision* (Brooks/Cole Publishing Company, 1999).

https://doi.org/10.1017/S0263574704001018 Published online by Cambridge University Press