Washington State University

Pose Estimation of a 3D Curved Element

Using Machine Learning to Estimate Configuration from a Single Camera View

Emily Allen & Ben Saunders

ME 579 Machine Vision

Dr. John Swensen

December 12, 2019

Introduction

The field of Soft robotics is new and has the potential to revolutionize how we implement robotics into modern society. However, since the field is so new, how we collect data and implement it to control soft robots is a problem that challenges research in the field. If real time control is to be applied a system needs to be fast and accurate. Many systems developed to tackle this problem have been proven to achieve the desired accuracy needed but only on specific robotic designs and they require slow intensive calculations. A different approach needs to be found, one that will be both accurate and fast.

Machine learning has the potential to satisfy both of the needs described above. The idea behind machine learning is that time is spent training the computer using deep learning techniques, so the computer will be able to identify and provide data in a fast and accurate manner. In a situation where the orientation of an object is desired, machine learning can be applied with a camera system that will allow the computer to identify the object's orientation from the images it is provided. After the training, the computer should be able to identify the orientation of a soft robot fast and precise enough to provide a means for real time control of the robot. Our approach to achieving real time control of a soft robot is to implement a neural network that detects the orientation of a continuum rod soft robot with only a single camera.

Problem

Our goal is to characterize the 3-dimensional bending of a long slender element using a single camera view. The bending of the element is characterized by 4 curve parameters θ_1, θ_2, ϕ_1, and ϕ_2 as shown below.

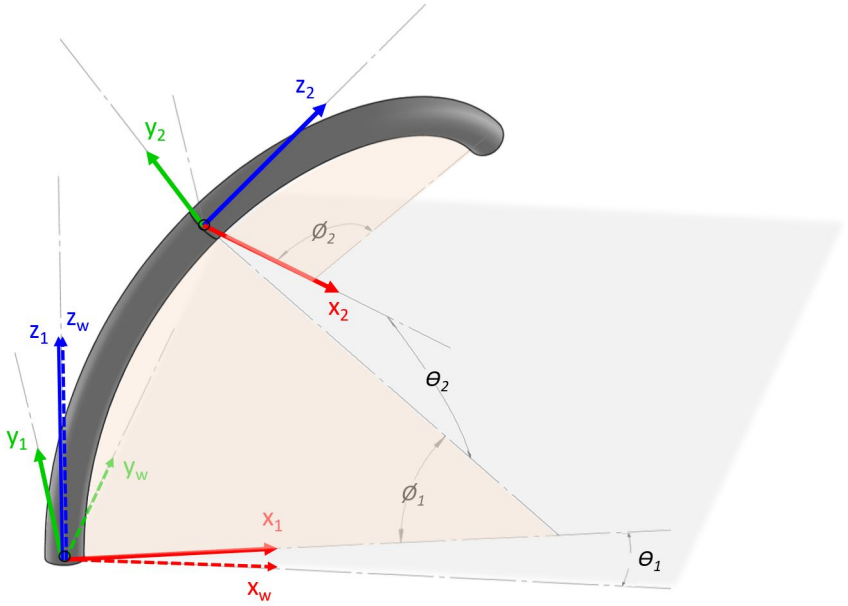

Figure 1. Curve characterization parameters θ_1, θ_2, ϕ_1, and ϕ_2.

Most machine learning environments are centered around the problem of image classification, where recognition of common features enables classification of an input image into one or more of several trained categories. Images may be categorized based on the main object in the image (cat, house, car, etc.), or based on more detailed information such as flower species or the number of bedrooms in a floorplan.

Here we face an entirely different type of machine learning problem. While it may be possible to categorize images based on the range of bend parameters that they indwell, it is more useful and accurate to estimate the four curve parameters using a regression model to predict continuous values. The classification method could tell us which of the trained images is most similar to our input image and report the curve parameters of the trained image, but the accuracy of these results would be extremely limited as it depends entirely on the closeness of the spacing of the training images. This method is also computationally demanding as it requires a huge number of categories for any reasonable degree of accuracy.

Estimation of the curve parameters based on regression, however, allows us to estimate parameters that lie between the values of trained images, thus improving accuracy while minimizing the number of training images needed. The regression method is enabled by the use of fully-connected and regression layers at the end of the Neural Network.

Method

Ground truth data images are gathered by modifying the curve parameters of an extruded cylinder in Blender and capturing images of different configurations from the same camera view. A total of 12500 images were gathered with curve parameters randomly selected within the ranges $\theta_1, \theta_2 \, \varepsilon \, [-\pi/16, \ \pi/16]$ and $\phi_1, \phi_2 \, \varepsilon \, [-\pi/2, \ \pi/2]$. With these parameter ranges, the element's bending is close to planar, never bending directly toward or away from the camera which would severely complicate the parameter prediction. The gathered images were separated into different groupings for training, validation, and testing of the Neural Network:

Total gathered:	12,500 images
Training set:	10,000 images
Validation set:	2,000 images
Testing set:	500 images

Several different software packages including Keras, Caffe, and Matlab can be used for implementing machine learning. For this project, using Matlab allowed customization of Neural Network layers and training options as well as simple interface with the Blender program.

The Convolutional Neural Network layers were constructed as follows:

Table I. Convolutional Neural Network Layering Architecture.

Layer Order	# Filters	Size	Stride	Padding	Rate	Dimension
Conv2D	32	5x5	1	Same		
BatchNormalization						
Relu						
MaxPooling2D			2			
Conv2D	32	3x3	1	Same		
BatchNormalization						
Relu						
MaxPooling2D			2			
Dropout					0.2	
Fully Connected						80
Fully Connected						40
Fully Connected						4
Regression						

The network is then trained using stochastic gradient descent with momentum on the training image set and its corresponding ground truth curve parameters. Training options were set with a mini batch size of 32 observations per iteration with validation between each of the 10 epochs.

The four-dimensional fully-connected layer and regression layer at the end of the network produces an output of four continuous values corresponding to the four curve parameters we used in the training data. Thus, the trained network can be used to predict the parameters for any input image with a curve configuration that lies within parameter ranges of the training images ($\theta_1, \theta_2 \, \varepsilon \, [-\pi/16, \ \pi/16 \,]$ and $\phi_1, \phi_2 \, \varepsilon \, [-\pi/2, \ \pi/2 \,]$).

Results & Discussion

The results of the Neural Network training are shown in the figure below.

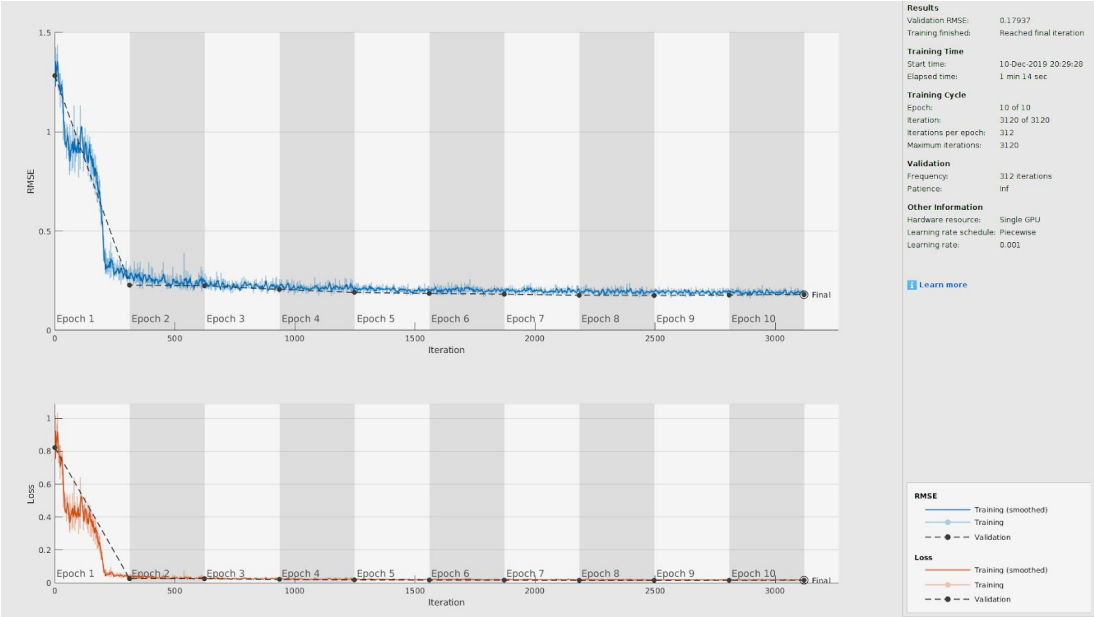

Figure 2: Training results with decreasing Loss and Root Mean Square Error every iteration

After training our network we then used it to predict the orientation of the 500 testing images of the rod in different positions. Four randomly selected results are shown in the figure below.

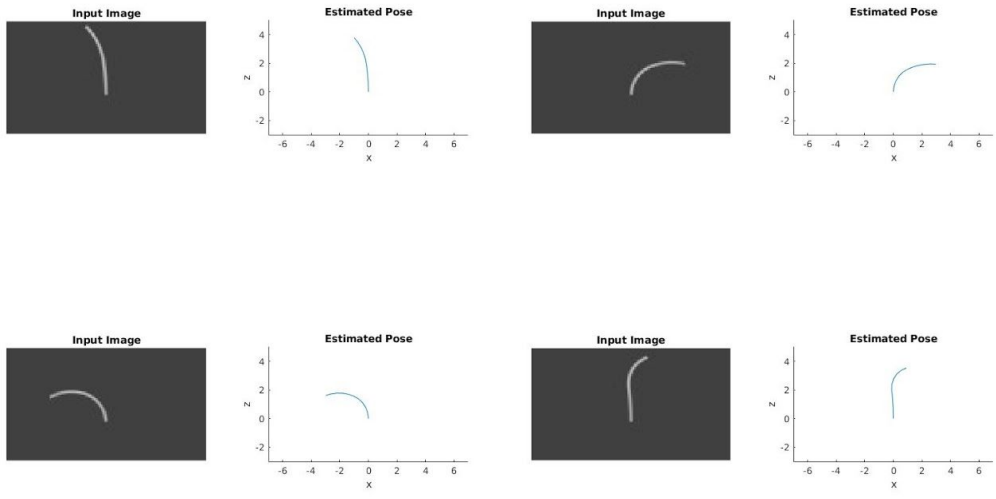

Figure 3: Images used to test Neural Network pose prediction. Images on the left are the actual images and images on the right are 3D curves reconstructed from the parameters our network predicted.

As seen in the figure above, the network Matlab produced was able to predict parameters that produced visually similar orientations when compared to the actual ones generated by Blender. However, the results proved to have limited accuracy in estimating the curve directions (θ_1 and θ_2) as seen from the figure below.

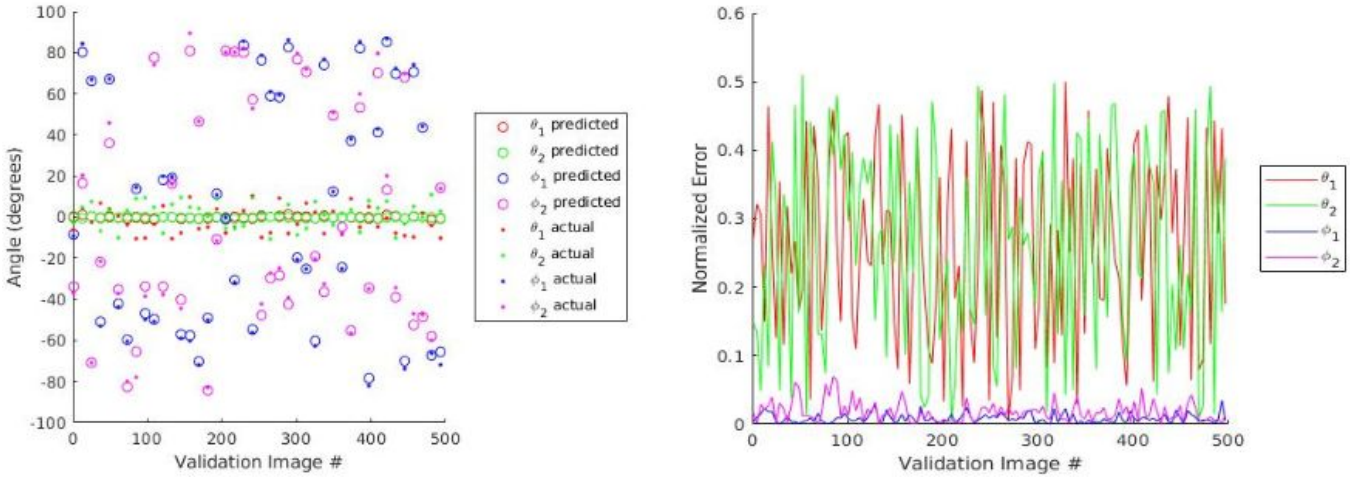

Figure 4: (Left) Plot comparing the actual parameters of the testing data with the predicted parameters from our network. (Right) Plot of normalized error (error as a fraction of the range) of the predicted parameters.

From Figure 4 it can be seen that the network was able to accurately predict ϕ_1 and ϕ_2 but, even with the limited range of motion for θ_1 and θ_2 it failed to generate accurate predictions for them. This is likely due to the depth ambiguity that arises from using only a single camera; even the different shadows and the deep learning employed could not distinguish between a bend toward the camera vs away from it. This ambiguity could also be affecting the slight error that is found in ϕ_1 and ϕ_2 .

Conclusion

Our findings from using machine learning with a single camera proved fruitful. Using a regression network we were able to predict 2 of the 4 parameters accurately. Using Matlab allowed us to construct the estimated 3D shape which looked similar to the actual images when viewed from the camera's perspective. Alterations to our system will need to be made to address the depth ambiguity problem and improve accuracy in the bend direction predictions.

Our current hypothesis is that the network struggled to predict two of the parameters due to depth ambiguity. With only one camera and the current data we are providing the network, the system has no way of being able to discern between a variety of combinations for θ_1 and θ_2 that could produce the image it is analyzing at any time. There are, however, multiple ways to approach finding a solution for this challenge.

One approach would be to add another camera at a different location for data retrieval. However, this would increase the amount of data required for training. It would also make setting up outside of simulation circumstances more complicated. Instead of two cameras the single camera being used could be moving to collect data from multiple perspectives. This however would dramatically increase the complexity of our situation. We would have to collect data on the camera's velocity at all times and the include the positional data of the camera into the training. The prediction time would also be greatly slowed down due to the network having to wait on multiple images from a moving camera to be able to guess the orientation of the rod. With this method, the Neural Network would not be necessary as the 3-dimensional shape could be reconstructed by triangulating points between the two camera views.

A better approach would be to improve the software. To do this we could modify the prediction system of our network so that it would take into account the last known position of the rod. It could then use this data to deduce the most likely values for θ_1 and θ_2. While the Neural Network may not be able to distinguish between a configuration with $\theta_1 = \pi/20$ vs. $\theta_1 = -\pi/20$, information about the curve parameters of the previous frame could solve this dilemma. If the previous frame was known to have a configuration with $\theta_1 = -\pi/22$, the system could use this information to choose the most logical parameters between two visually similar configurations. This modification will improve our network so that it can accurately predict all four parameters at a speed fast enough for real time control of a continuum robot.