Shape Computation for Concentric Tube Robots
Project Progress Report for Introduction to Machine Learning

Ahmet Hatip ASHATIP@LIVE.UNC.EDU
Chris Hemmer CAHEMMER@LIVE.UNC.EDU
Paul Song BIGMSONG@LIVE.UNC.EDU
Eric Xin ERICXIN@LIVE.UNC.EDU

Abstract

Concentric tube robots are the future of non-invasive surgery. However, a major problem with these robots is that it is very computationally expensive to calculate the position of the robot (Torres, et al., 2015). We attempted to solve this problem from a machine learning perspective, by employing the use a simple sequential neural network to predict the position of the robot arm. Based on the results we have attained, in respect to accuracy and efficiency, we believe that the use of neural networks to solve this problem is the optimal way forward.

1. Introduction

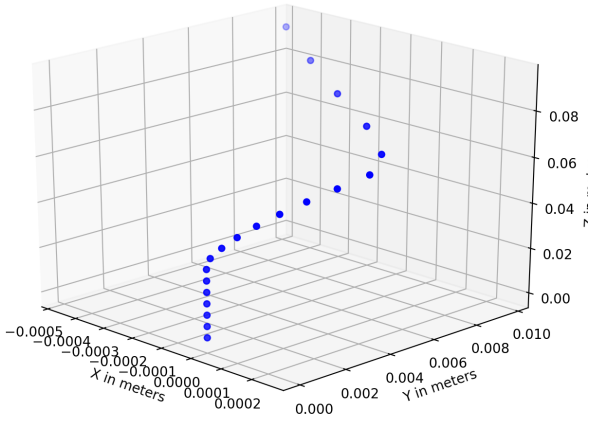
1.1. Literature Review

Concentric tube robots are robots with different re-curved telescoping tubes. Each tube is controlled by rotation and translation. Because of the tubes size being small, these robots are the future of minimal invasive surgery. However, computing the shape of these tube based on the rotation and translation is computationally difficult. Because the movement of the whole robot is continuous it takes a longer time to calculate the location when compared to robots with discrete movements (Torres, et al., 2015).

Neural networks may lower the computation time. It is proven by Hornik et al. that our model can be built using neural networks because of the Universal Approximation Theorem, which states that a neural network with one hidden layer can approximate any bounded continuous function.

1.2. Description of the Problem

The problem that we are trying to solve with our work is given 6 inputs (3 translational positions of motors, and 3 rotational positions of motors) predict the position (in a 3D space) of the robot arm. Not only do we need to predict the position accurately, it needs to be faster than the current method of calculation (.001 seconds).



This is a sample of what the 20 points look like in 3D space.

1.3. Limitations of work

With the purpose of our model being for use in a medical instrument, accuracy is extremely important. A poor prediction could lead to a disaster, so it is crucial that we strive for a model with high accuracy. However, we want to stray away from creating a model that is slow, but accurate, as that defeats the purpose of our project.

2. Baseline Results

The traditional method of calculating the position of the arm via ODEs yields an incredibly accurate prediction but with a low efficiency. (Torres, et al., 2015) It takes roughly 5 milliseconds to calculate the position of the arm, which when this calculation needs to be done incredibly often, as

is required with surgical instruments, is not fast enough. This shortfall of concentric tube robots is the entire motivation behind this project, where our end goal is to create a model capable of accurate predictions in an amount of time significantly less than the current approach.

3. Initial Step

3.1. Initial Model

The data we received contained 1 million data sets of 6 inputs and the x,y,z values for 20 equi-arc-distant points along the robot arm. Before trying to predict the position of the whole robot arm, we tried to see the feasibility of accurately predicting just the final coordinate at the end of the arm using a neural network. We first built a model that took in 6 inputs (the robot configuration) and gave 3 outputs (x, y, z of the endpoint coordinate). We split our data into a training set and a test set 80/20. The training set is further split into 80/20 for cross validation. For our initial model, we scaled our three target variables to have zero mean and unit variance. We used a Keras Neural Network Sequential Model with two hidden layers, each having 64 neurons and using ReLU as its activation function. We used mean squared error as our loss function, RMSProp with a learning rate of 0.001 as our optimizer, and mean abs error as our error metric.

$$\boxed{Input : 6\ parameters\ (l_1, l_2, l_3, \theta_1, \theta_2, \theta_3)}$$

$$\Downarrow$$

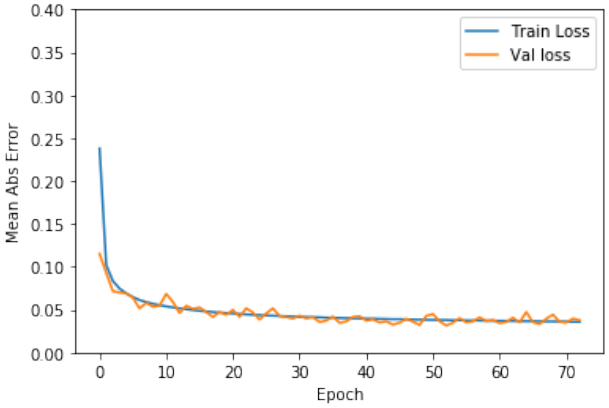
$$\boxed{\begin{array}{l} Layer\ 1 : 64\ nodes,\ Activation :\ relu \\ Layer\ 1 : 64\ nodes,\ Activation :\ relu \end{array}}$$

$$\Downarrow$$

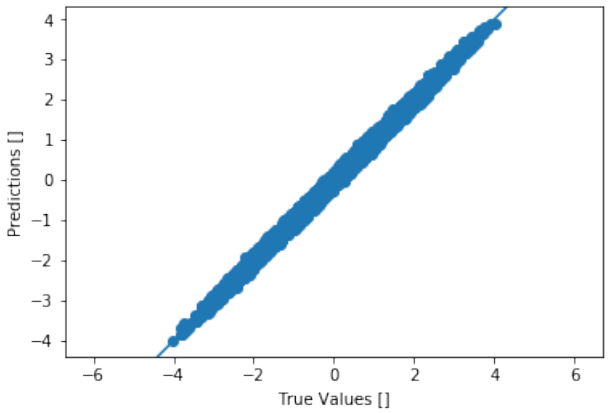
$$\boxed{Output : 20^{th}\ point\ (x_{20}, y_{20}, z_{20})}$$

3.2. Initial Results

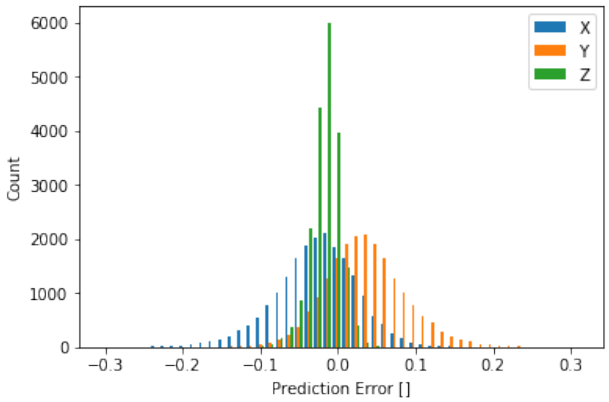
Our model produced a scaled mean abs error of about .029.



This graph shows the "loss" as it trains. The model stops early if the val loss is roughly stable for 20 epochs.



This graph plots the actual values of the test set with the values predicted by our model. A perfectly accurate prediction would fall on the line given y=x.



This graph shows the error distribution of each coordinate: x is blue, y is orange, and z is green. Our model predicts the z value well, but it has more variance with the x and y coordinates. This makes sense as the z is the forward distance that mostly depends on the translational components of the motors.

Overall, our model, though not the best nor most accurate, shows that it is feasible for a neural network to model the final position of the arm.

3.3. Linear Regression

We built a linear regression model with rotational and translational motor configurations as inputs and 20 3D points as outputs. The model had around 18 coefficients, each of them representing dimension and a motor. The training error was 0.4182 and test error 0.4180, yet the testing time was 0.1872 sec, and the training time was 1.8295 sec.

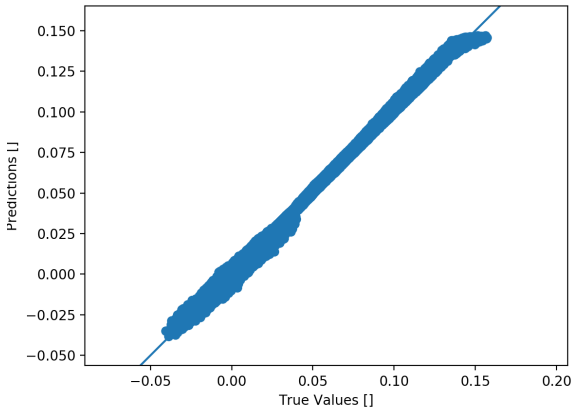
3.4. Mapping out every point

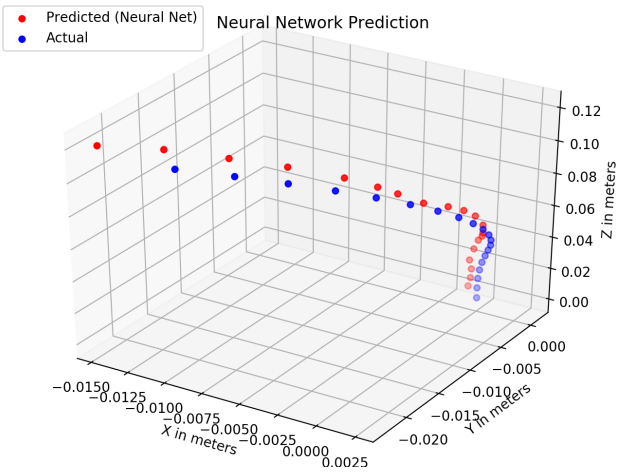
Having created a neural network capable of predicting the position of the end of the robot arm, we went about trying to create a network to predict the entire shape of the arm. Eventually, we settled on simply predicting the 20 equidistant points on the arm of the robot.

Our approach to solving this problem was to add another parameter to the input of our neural network, representing which of the 20 points the model was to predict. Upon this modification, the neural network took up this form.

$$\boxed{\begin{array}{c} \textit{Input} : 7\ \textit{parameters}\ (n,\ l_1, l_2, l_3, \theta_1, \theta_2, \theta_3) \\ \Downarrow \\ \textit{Layer}\ 1 : 64\ \textit{nodes},\ \textit{Activation} : \textit{relu} \\ \textit{Layer}\ 2 : 64\ \textit{nodes},\ \textit{Activation} : \textit{relu} \\ \Downarrow \\ \textit{Output} : n^{th}\ \textit{point}\ (x_n, y_n, z_n) \end{array}}$$

Upon training this model with 80% of our data, the network achieved a mean squared error of 0.0000024. This accuracy comes at a cost however, as the time required for the model to predict all 20 points on the curve is 0.019 seconds, which we hope we can improve.



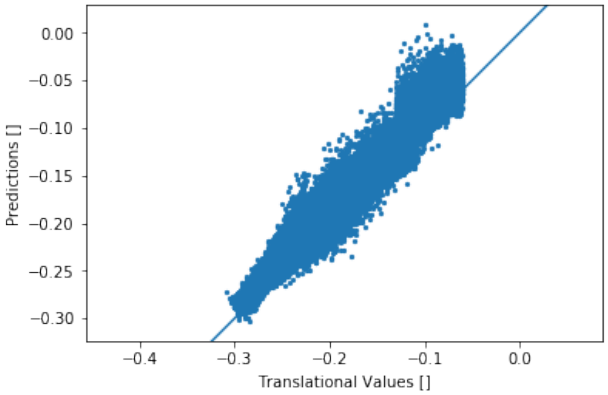


3.5. Reverse Neural Network

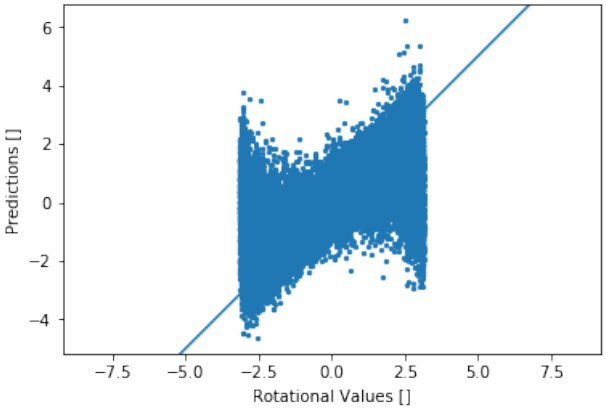
Now that we have achieved a neural network that mapped out the position of the arm using 6 motor inputs, we tried to do the same but in reverse. We made a neural network that can predict the 6 motor inputs using all 20 positions. Since we have a larger input, we created a bigger network with 3 layers of 256 ReLU nodes.

$$\boxed{\begin{array}{c} \textit{Input} : 20\ \textit{points}\ (x_i, y_i, z_i) \\ \Downarrow \\ \textit{Layer}\ 1 : 256\ \textit{nodes},\ \textit{Activation} : \textit{relu} \\ \textit{Layer}\ 2 : 256\ \textit{nodes},\ \textit{Activation} : \textit{relu} \\ \textit{Layer}\ 3 : 256\ \textit{nodes},\ \textit{Activation} : \textit{relu} \\ \Downarrow \\ \textit{Output} : 6\ \textit{parameters}\ (l_1, l_2, l_3, \theta_1, \theta_2, \theta_3) \end{array}}$$

This neural network model produced a smaller error than other models tested for predicting the motor configurations. We found our mean squared error to be about .58025. The network took on average .0010975 seconds to predict the 6 motor values over 100 sets of data. Below are graphs that shows how accurate our model predicted the translational values and rotational values.

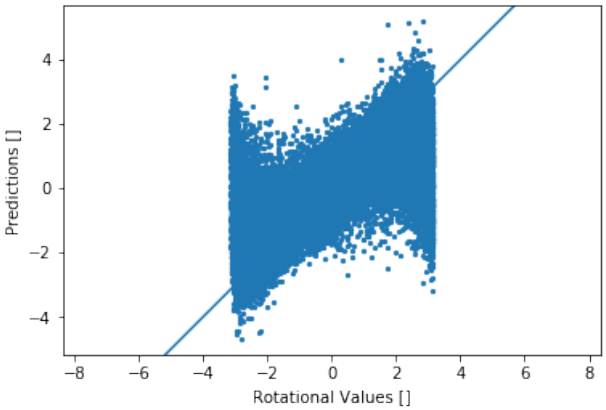


The predicted translational values seem very close to the actual values. However, if we take a look at our predicted rotational values.

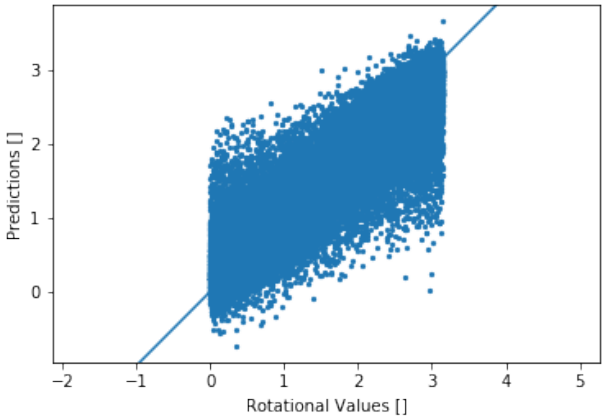


The predicted rotational values are all over the place especially near the limits of the actual values which is around -π and π. This is because -π and π are actually the same rotation angle yet our neural network is unable to smoothly predict values for such a discontinuous break. We also figured that the neural network does not understand how the rotations affect the 3D points. This makes sense as the rotational values have a complex relationship with the position of the robot arm.

We created an identical neural network except it outputs just the 3 rotational values. We hoped to see the model be able to learn and predict better as it only focuses on the rotations. However, our results show that our network was unable to predict the rotational values consistently. In fact, it predicted the values about the same way as our original reverse model.



So, we decided to simply the data by taking the absolute value of each rotational value. We used the same neural network to predict our values and our mean squared error was about .12415.



This improvement shows that our model can indeed be improved if we were able to make the data more comprehensible and easier for our model to learn and predict the rotational values.

4. Going Forwards

Although the results we have achieved so far are promising, there is still much work to be done. We need to fine tune and improve our models in order to increase our accuracy. Also we need to analyze the training and testing errors to see if the model is under-fitting or over-fitting the data. Things we can adjust from our initial models include: number of hidden layers, neurons in each hidden layer, activation functions, loss function, optimizer, and method of standardizing data. Once we believe that our models are at an optimal accuracy, there are two main ways we could further develop our project.

4.1. Increase Complexity

As of now, our model for predicting the curve only predicts the 20 equidistant points on the robot arm's curve. Although this is good for approximating the entire curve, it is not optimal. Our model could be altered to treat the position of the arm as a bounded polynomial or spline function, and use the neural network to find the coefficients for that function. The benefit of this approach is that the set of positions will be continuous rather than discrete.

If the former approach does not pan out, we could also alter our model to be a recurrent neural network. Within this modification, we would still only predict the 20 points on the curve. However, using the position of point x as a parameter for the prediction of point x+1 could significantly improve our predictions and make for a more continuous set of points.

4.2. Predicting Rotational Values

As discussed before, our current "reverse" neural network model does not predict the rotational values consistently. We would need to look into other ways to improve our model. Maybe using sine and cosine to make it easier for our network to interpret the rotational values. We also considered using a more complex model. This can again include adding more hidden layers, more nodes, using a different activation function, or testing recurrent neural networks.

Acknowledgments

UNC PhD student Alan Kuntz has advised us on this project. All of the data points that we have were generated and provided to us by Alan. Alan also demonstrated the use of a concentric tube robot to us, which gave us a deeper understanding of the problem at hand. It is our goal for Alan to find value in the work done, and further develop our models to create a usable end product.

References

Luis G. Torres, Alan Kuntz, Hunter B. Gilbert, Philip J. Swaney, Richard J. Hendrick, Robert J. Webster III, and Ron Alterovitz. (2015). "A Motion Planning Approach to Automatic Obstacle Avoidance during Concentric Tube Robot Teleoperation. Proc. IEEE International Conference on Robotics and Automation (ICRA), pp. 2361-2367

Hornik, K.; Stinchcombe, M.; White, H. (1989). "Multilayer feedforward networks are universal approximators". Neural Networks. 2 (5): 359. doi:10.1016/0893-6080(89)90020-8

Shape Computation for Concentric Tube Robots

Paul Song, Christopher Alan Hemmer, Eric Xin, Ahmet Hatip

[1]Department of Computer Science, University of North Carolina at Chapel Hill, NC 27599, USA

Summary: Concentric tube robots are surgical instruments specifically designed with non-invasive procedures in mind. However, the ability to be non-invasive comes at a cost, calculating the position of the robot is computationally expensive[1]. Our goal for this project was to apply machine learning techniques (specifically linear regression and neural networks) to create models for producing accurate, and time efficient approximations of these calculations in order to make the use of these robots more feasible. Our optimal model for predicting the position of the arm was a neural network, which yielded a test-error of 0.00122, and could plot the whole curve in 0.019 seconds. Our optimal model for predicting the settings of the motors, given the position of the arm, was a neural network, which yielded a test-error of .32225

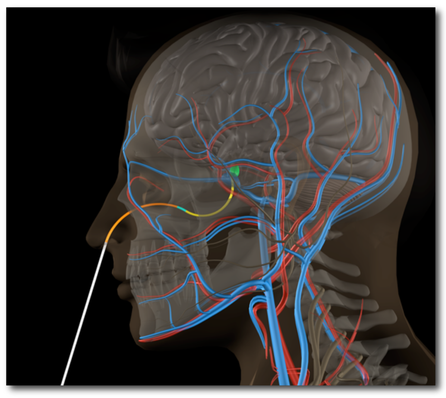
Motivation

- Concentric tube robots are the ideal tool for non-invasive surgery, except for one drawback: calculating the position of the arm is too computationally slow[1]
  - Calculating the position of the arm takes 0.001 seconds using the current approach (solving a set of ODEs)
  - This is a project which allows us to have a large impact on the feasibility of using these robots in surgical procedures
- Goal 1: Create a model for accurately predicting the positions of the robot arm, given 6 input values, in a smaller amount of time than the current methodology
- Goal 2: Create a model for predicting the settings (translational and rotational values) of the motors, given the 20 points along the robot arm

Introduction

- Concentric tube robots are surgical devices specialized for non-invasive usage[1]
- Inefficient to use due to their computationally expensive nature

Concentric Tube Robot [2]



Method

- Model 1: Linear regression
  - A simple linear model with 6 coefficients for every 60 values on the curve

- Model 2: Neural Network
  - 2 Hidden layers of 64 ReLu nodes
  - Input: 6 motor values, and which point on the curve is to be predicted
  - Output: 3D coordinate of the specified point

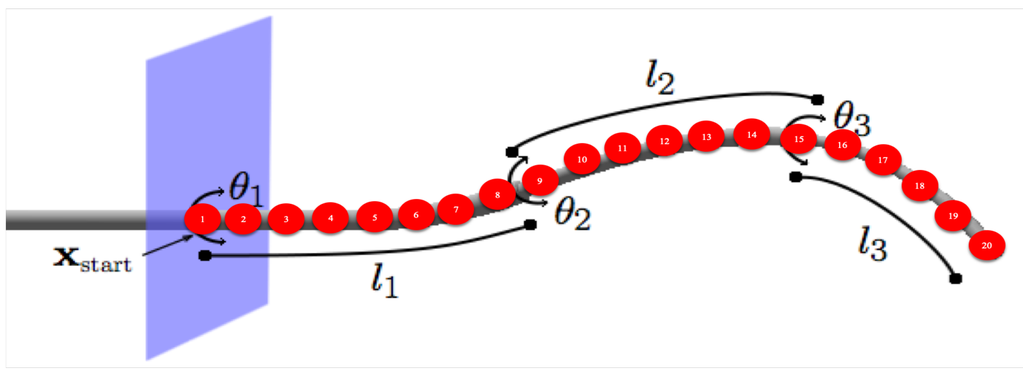
$$Input: 7\ parameters\ (n,\ l_1, l_2, l_3, \theta_1, \theta_2, \theta_3)$$
$$\Downarrow$$
$$Layer\ 1: 64\ nodes,\ Activation: relu$$
$$Layer\ 2: 64\ nodes,\ Activation: relu$$
$$\Downarrow$$
$$Output: n^{th}\ point\ (x_n, y_n, z_n)$$

- Model 3: Neural Network
  - 3 Hidden layers of 256 ReLu nodes
  - Input: 20 3D coordinates
  - Output: 6 motor values

$$Input: 20\ points\ (x_i, y_i, z_i)$$
$$\Downarrow$$
$$Layer\ 1: 256\ nodes,\ Activation: relu$$
$$Layer\ 2: 256\ nodes,\ Activation: relu$$
$$Layer\ 3: 256\ nodes,\ Activation: relu$$
$$\Downarrow$$
$$Output: 6\ parameters\ (l_1, l_2, l_3, \theta_1, \theta_2, \theta_3)$$

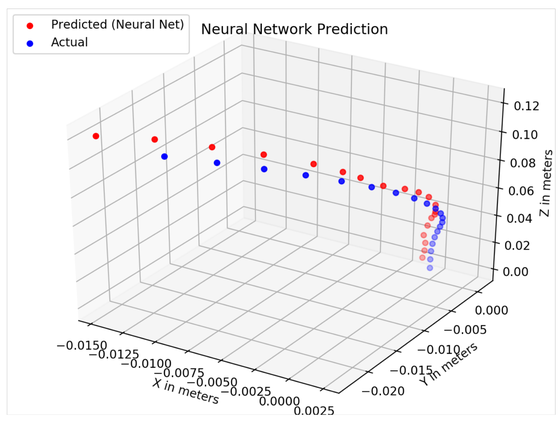
Data

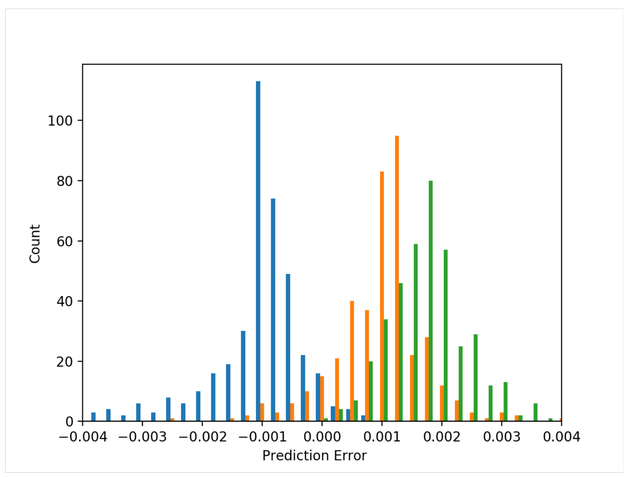
- Alan Kuntz provided us with one million configurations of a concentric tube robot
  - The data was generated on his concentric tube robot in the UNC Robotics Lab
  - Each data point consisted of:
    - 3 translational positions of motors
    - 3 rotational positions of motors
    - 20 equidistant 3D points on the curve of the robot's arm



Experimental Results

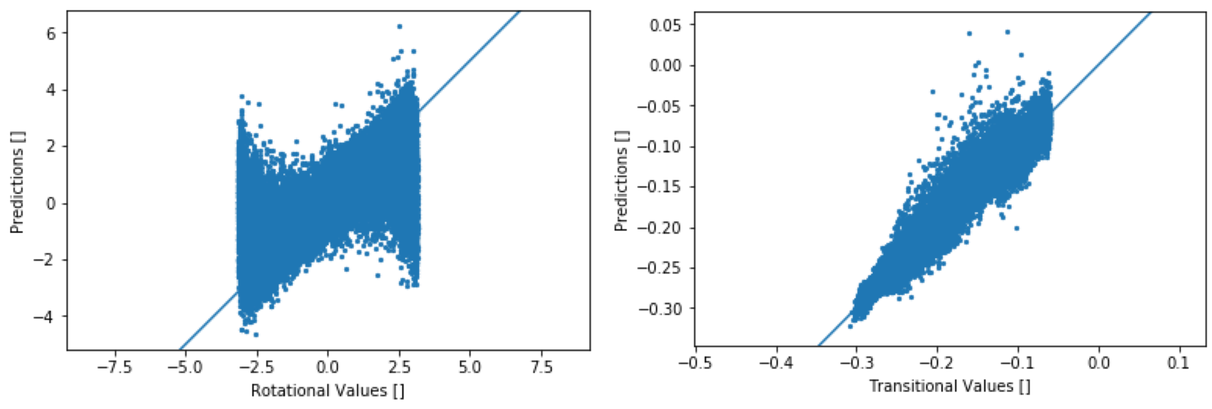
- Mean Squared Errors of the different models
  - Linear Regression to predict the 20 points (Mean Square Error)
    - train error 0.509
    - test error 0.508
  - Neural Network to predict 20 points: 0.0000024005 (MSE)
    - Able to plot the entire curve in 0.019 seconds





- Neural Network to predict motor settings: .58025 (MSE)
- Average Time to predict: .0010975 seconds

Model 3 predictions



Conclusions

- For predicting the position of the arm, we found that our neural network yields better results and has more potential to improve
- For predicting the transitional and rotational values of the motors, our neural network has a problem understanding values at the limits of the motor inputs
- Although our neural networks are promising, the accuracy will need to be improved for the networks' use to be more practical.

References:
[1] Luis G. Torres, Alan Kuntz, Hunter B. Gilbert, Philip J. Swaney, Richard J. Hendrick, Robert J. Webster III, and Ron Alterovitz, "A Motion Planning Approach to Automatic Obstacle Avoidance during Concentric Tube Robot Teleoperation," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 2361-2367
[2] Dr D. Caleb Rucker and Dr. Robert J. Webster III