Bridge of Doom: Quarantine Edition
QEA Spring 2020

Sam Kaplan

April 16, 2020

1 Introduction

The goal for this project was to have a simulated NEATO robot traverse a dangerous volcanic bridge using an open loop control system. This was accomplished mainly through MATLAB and its ROS toolbox. The centerline of the bridge the NEATO robot had to traverse is defined by the following parametric curve:

$$\mathbf{r}(u) = 4 * [0.3960\cos(2.65(u+1.4))\mathbf{i} - 0.99\sin(u+1.4)\mathbf{j}].(u \in [0, 3.2])$$

2 Methodology

First we have to find the unit tangent and unit normal vectors for the given parametric curve. This was accomplished using MATLAB's symbolic notation. The unit tangent vector and unit normal vector are defined by the following equations:

$$\hat{\mathbf{T}} = \frac{\mathbf{r}'}{|\mathbf{r}'|}$$

$$\hat{\mathbf{N}} = \frac{\hat{\mathbf{T}}'}{|\hat{\mathbf{T}}'|}$$

Because we must modulate our parametric curve to have our wheel speeds remain under 2 m/s, we set

$$\mathrm{u} = \beta\ {}^*\mathrm{t}$$

where β for the predicted model is found through trial and error, and define 't' as a one dimensional matrix with 1000 numbers between 0 and $3.2/\beta$. Using the appropriate formualas we can then calculate our position vectors, unit tangent vectors, unit normal vectors, linear velocity, angular velocity, and our two wheel

speeds. To do this we substitute the values in the array 't' into our symbolic equations using MATLAB's 'subs' function. The equations for the necessary vectors are as follows:

$$\omega = \hat{\mathbf{T}} \times \frac{d\mathbf{T}}{dt}$$

$$V_{left} = V - \omega \frac{d}{2}$$

$$V_{right} = V + \omega \frac{d}{2}$$

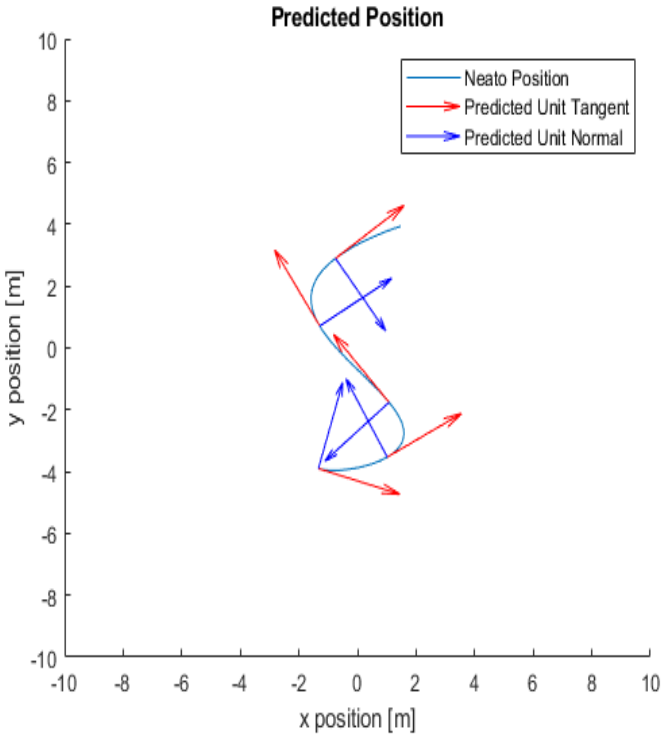

Figure 1: Unit tangent and unit normal vector directions for the parametric curve.

Please note that V is linear speed which can found by calculating the magnitude of the first derivative of the parametric function.

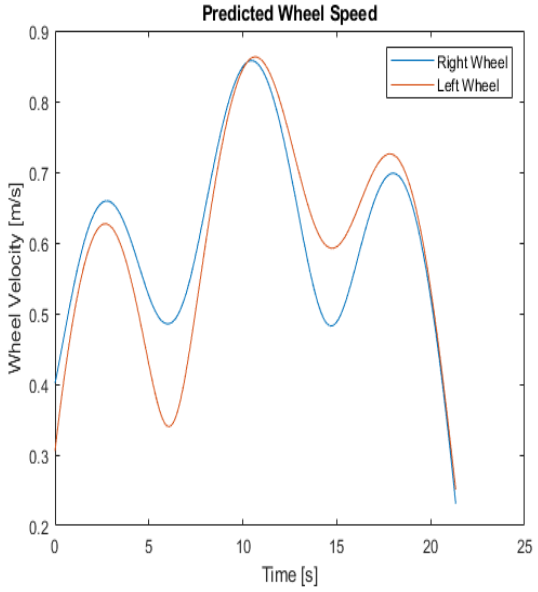

Figure 2: Predicted left and right wheel velocitie.

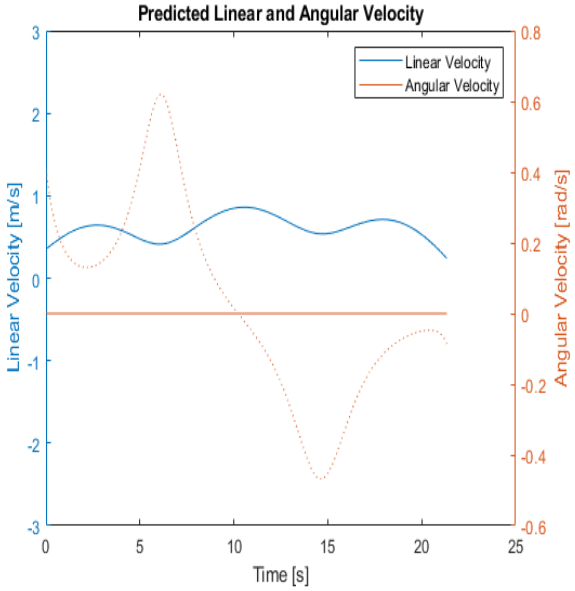

Figure 3: Predicted linear speed and angular velocities. Please ignore the weird line in the middle.

In order to get the simulated Neato to traverse the Bridge of Doom, it needed to die. Many times. Due to the fact that the program recording the wheel encoder data delays server communication, the β previously found is now useless, because the NEATO is perceiving time differently. Therefore, I iterated over hundreds of possible β's and sat infront of my computer for several hours until one of them worked. Inside each sweep of beta was another for-loop iterating over the velocity vectors each wheel. Thank you to the teaching team for providing startercode to place the NEATO on the bridge at the correct starting position. Using this method, I found a β of 0.32367 to get the NEATO to cross relatively consistently.

Once a successful beta was discovered, we used the data collected script to record the position of the NEATO's wheels over time. Using this data, we cas easily find the left and right wheel velocites over the given time interval. The rest of the information we need to reconstruct the measured linear velocity, angular velocity, and ultimately position, can be found using the following equations:

$$V = \frac{Vleft+Vright}{2}$$

$$\omega = \frac{Vright-Vleft}{d}$$

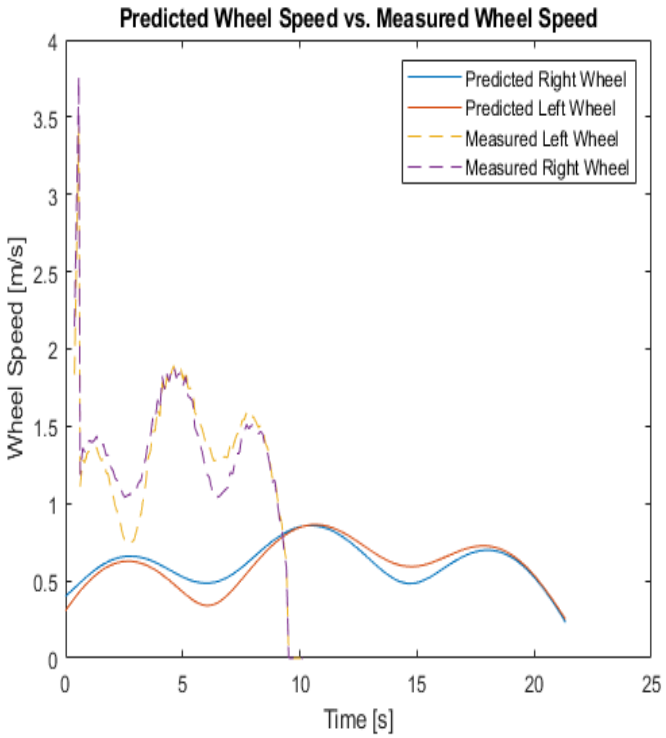

Figure 4: Predicted and measured left and right wheel velocities.

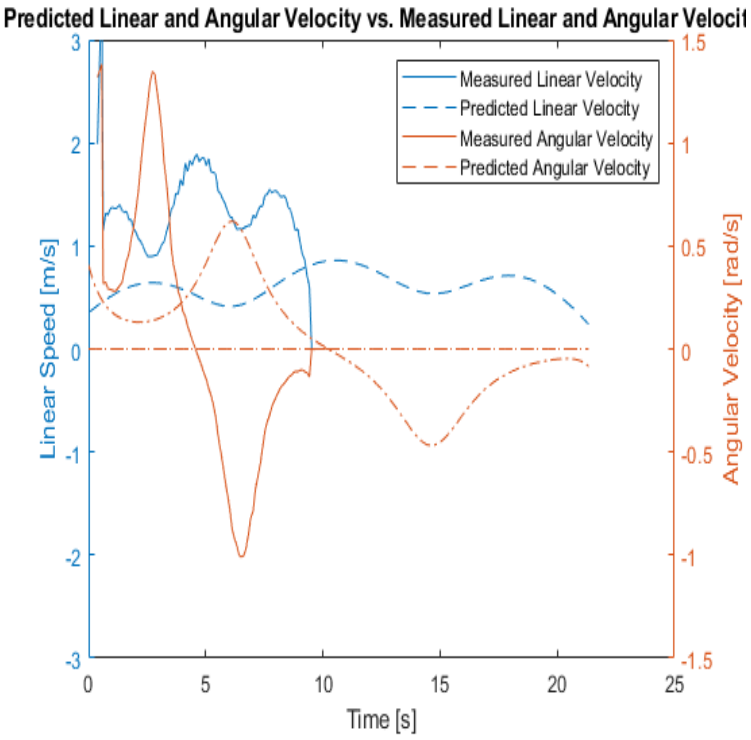

Figure 5: Predicted and measured linear speeds and angular velocities.

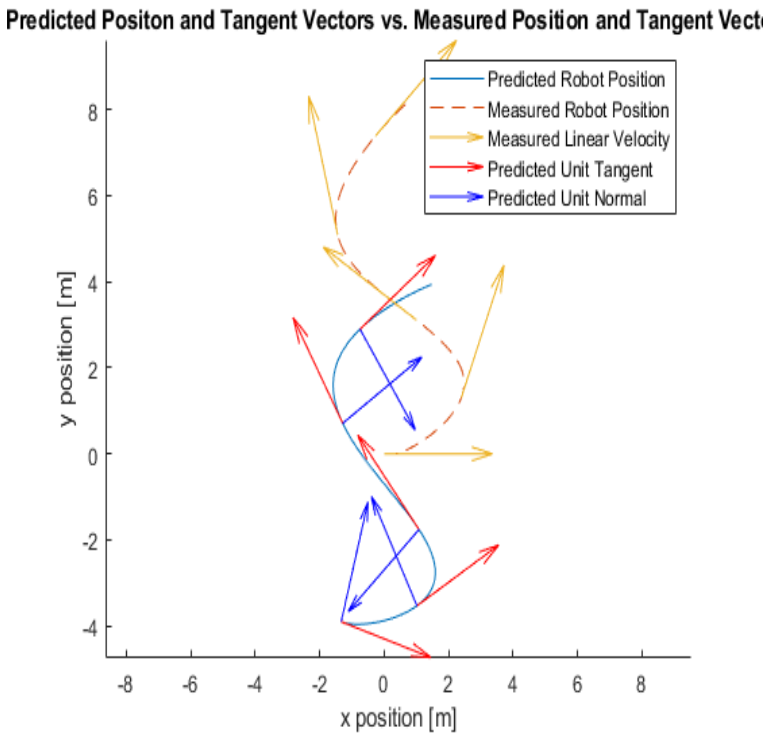

Figure 6: Predicted and measured NEATO paths, along with their respective unit tangent and unit normal vectors.

Watch the perilous journey for yourself: `https://youtu.be/pFaZ9D6f-rY`

Look at my MATLAB code for this project:
`https://github.com/slkaplan/BridgeOfDoom-QEA-Spring-2020`