EE 3 Final Report

Lawrence Liu
lawrencerliu@ucla.edu

Inesh Chakrabarti
inesh33@g.ucla.edu

Electrical and Computer Engineering
University of California, Los Angeles
United States of America
June 4th 2022

Acknowledgements

We would like to acknowledge and extend my sincere thanks to Dr. Dennis Briggs who made this project possible. His knowledge of the subject, guidance, and support through the whole quarter carried us through this project. We would also like the acknowledge our TAs Dhruv Srinivas and Xin Li for your brilliant comments and suggestions. Furthermore we would like to thank from the bottom of our hearts our wonderful mentors Alexis Aleksandrovich Samoylov and Vaibhav Gupta for their guidance and support.

Finally we would also want to thank our families, without them none of this would be possible.

Introduction

The goal of this project was to design a closed loop controll system for a small "car" robot to follow a line. The car was a Texas Instrument (TI) Robotics Systems Learning Kit (RSLK).
This car was controlled by a TI MSP-EXP432P401R launchpad microcontroller. This microcontroller is part of TI's MSP432P401x family of ultra low power microcontrollers.

The CPU is a ARM 32-bit Cortex-M4 RISC engine with a frequency of up to 48 MHz. Furthermore the microcontroller has 256KB of Flash Main Memory, 16KB of Flash Information Memory, and 64KB of SRAM.

Testing Methodology

Our development followed two routes, we tried both a basic PID developmental route and a attempt to develop a ML model to controll the car through Deep Q reinforcement learning.

Unfortunately, the Deep Q reinforcement learning could not be made to work. Our code had memory leaks and it was difficult working around the microcontroller's small memory size.

So we result in using a basic PID controller. We realized that since the car is always moving, there is no steady state. Therefore we did not need the Integral part of the PID controller since there would be no steady state error to eliminate. Therefore the closed loop transfer function from the input sensor fusion value to the car movement with the car's plant transfer function being $G(s)$ was

$$\frac{(k_p + k_d s)G(s)}{1 + (k_p + k_d s)G(s)}$$

Where k_p and k_d are the proportional and derivative gains respectively. Let us call the output from the controller to be d, and assume that the car was set to travel at a base speed of V_{base}, then because of our controller, the left and right velocites of the wheels would become

$$V_{left} = V_{base} - d$$

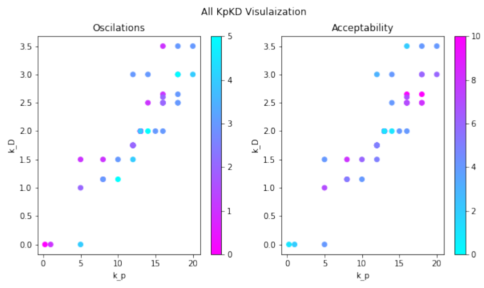
$$V_{right} = V_{base} + d$$

Therefore the parameters we would want modify would be k_p and k_d and the base speed. To determine the optimal values for these parameters, we would measure whether the car completed the track, and if it did, the time it took the car to complete the track.

Analysis

To facilitate the analysis of the system we developed two metrics to quantify how the car performed. The first metric was Oscillations, quantified from 5 (full oscillations) to 0 (no oscilations). . The second was acceptability quantify from 10 (fully acceptiblity) to 0 (not acceptable). This criterion was based on both the speed of the car, the time it took to complete the track, and how close it came to completing the track if it did not.

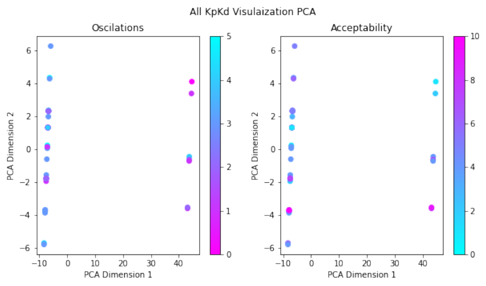
To analyze how the k_p and k_d values affected these two metrics we plotted a scatter graph of the oscilation and acceptablity for all the values of k_p and k_d we experimented with, regardless of the base speed.



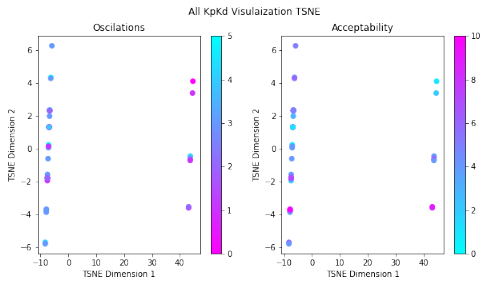
However this does not take into account our development path. We first attempted to control the car with PID at 50 speed. Once this succeeded we increased the speed to 100 speed, however the car was unable to keep on the track at this speed. So we focused on making a peicewise PID controller with different base speeds and k_p and k_d values.

Since we already had a working set of values for k_p and k_d for 50 speed, we focused on developing another set of them for 100 speed that could complete everything but the initial chicane.

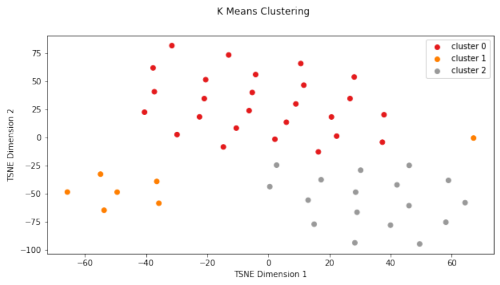
Therefore we would want to take into account these factors, such as the velocity, the battery voltage, and whether we were testing over the entire track. However this would make the data we would want to visualize to have more than 2 dimensions, so we would have to use a dimensionality reduction algorithm to reduce the data to 2 dimensions that is plottable. We first try to use Principal Component Analysis (PCA) to reduce the data to 2 dimensions.



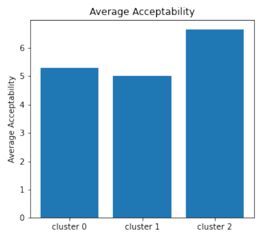
As we can see, the PCA algorithm did not work well. Therefore lets try to use t-SNE to reduce the data to 2 dimensions.

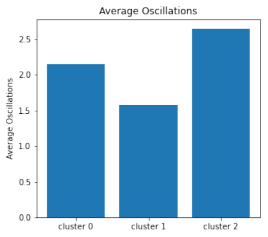


This algorithm seems to work well, and it seems like the data is organized into 2-3 clusters. Therefore let us cluster it with K-means at see the average acceptablity for each cluster. K means for 3 clusters classifies the data as the following:



Therefore, K means does successfully classify the data into the 3 clusters we would expect. The average for the Acceptibility and Oscillation metrics for each cluster are plotted below as a bar graph.





Interpretation

A notable run was our 3rd run with $k_p = 5$ and $k_d = 0$ as we had not implemented a derivative controller yet this run showed that we needed such a term, or we would have to reduce the speed of the car and k_p in order to reduce the oscillations.

Conclusion

From this run mentioned and the runs preceding and after it, we developed the following algorithm to tune the k_p and k_d: increase k_p until oscillations developed, then increase k_d until the oscillations stoppped. and then repeating these two steps until the car was able to achieve the objective for this controller.