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 our 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 their 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 whom 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

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

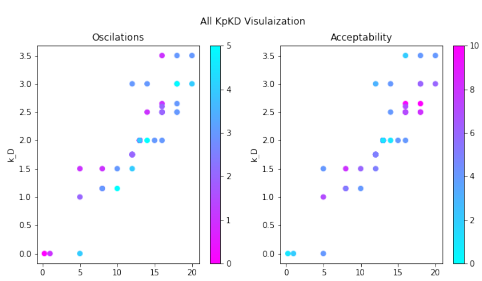
$$V_{right} = V_{base} + d$$

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.

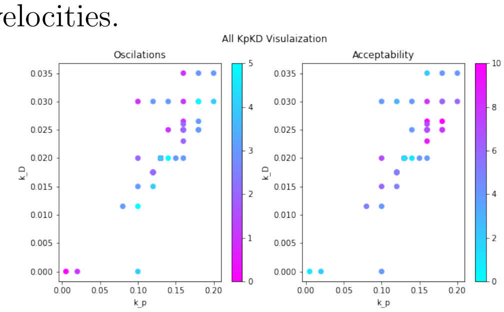
To analyze how the k_p and k_d values affected these two metrics we plotted a scatter graph of the oscilation and acceptablity for all the values of k_p and k_d we experimented with, regardless of the base speed. In other words, the parameters we controlled where k_p and k_d, while the response variables we measured were oscillation and acceptability, both of which were quantified.



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.

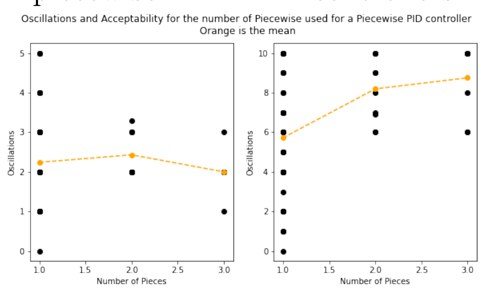
Since we already had a working set of values for k_p and k_d for 50 speed, we focused on developing another set of them for 100 speed that could complete everything but the initial chicane.

To try to determine the optimal k_p and k_d for different velocities we plotted the k_p and k_d normalized by the velocities.



As we can see it seems like the optimal k_p and k_d are around $k_p = 0.17v_{base}$ and $k_d = 0.5$ for 50 speed and $k_p = 0.5$ and $k_d = 0.5$ for 100 speed.

Now we tried the piecewise method, the following plot illustrates, acceptability and oscillation for different number of piecewise PID controllers.



Note that this makes sense due to the three sections of development we had with intiially tuning the car to a singular speed, then with two speeds, and finally with three speeds for the three differnet sections of the track. We see that indeed, the piecewise PID controller with three sections had the best performance. Organizing the data for each of these three method, we get three tables, as shown below:

Table 1: Simple PID results

k_P	k_D	Oscillations	Acceptability	Speed
0.25	0	0	1	50
1	0	1	2	50
5	0	4	4	50
5	1	2	7	50
5	1.5	1	4	50
8	1.5	1	8	50
8	1.15	2	9	50
8	1.15	3	5	100
10	1.15	5	4	100
10	1.5	3	6	100
12	1.5	4	5	100
12	1.75	2	7	100
12	1.75	2	0	100
12	1.75	2	10	100
12	1.75	3	10	100
12	1.75	2	4	100
12	1.75	2	4	100
12	1.75	2	5	100
13	2	1	10	100
13	2	1	10	100
13	2	1	9	100
13	2	1	9	100
13	2	1	10	100
13	2	1	2	100
16	2	3	4	100
15	2	3	4	100
13	2	1	2	100
13	2	1	2	100
13	2	1	5	100
13	2	1	6	100
13	2	4	1	200
13	2	4	1	200

Table 2: PID with two-part piecewise implementation

k_{P_1}	k_{D_1}	v_1	k_{P_2}	k_{D_2}	oscillations	accetability
8	1.15	25	18	2.65	2	9
8	1.15	50	18	2.65	2	10
8	1.5	50	18	2.65	2	8
15	0.5	50	18	2.65	2	10
15	0.5	75	20	2.65	3	7
15	0.5	75	20	2.65	3	9
15	0.5	75	20	2.8	3	6
15	0.5	75	20	2.6	2	6
15	0.5	75	20	2.6	2	6
15	0.5	75	20	2.6	3	9
15	0.5	50	20	2.6	3	10

Table 3: PID with three-part piecewise implementation

k_{P_1}	k_{D_1}	v_1	k_{P_2}	k_{D_2}	k_{P_3}	k_{D_2}	v_3	oscillations	acceptability
15	0.5	50	20	2.6	15	0.5	50	1	10
15	0.5	50	20	2.6	15	0.5	100	3	6
15	0.5	50	20	2.6	10	1	100	2	6
15	0.5	50	20	2.6	10	1	75	2	10
15	0.5	50	20	2.6	10	1	75	2	10
15	0.5	50	20	2.6	10	1	75	2	10
15	0.5	50	20	2.6	10	1	75	2	8
15	0.5	50	20	2.6	10	1	75	2	10

We see from these tables that our best results were indeed, as per our earlier plots, the piecewise PID with three sections.

Next, we attempted to understand how much the voltage of the car had affected our performance. To do this, we only considerd runs that were repeate with different voltage values; however, there were only three such runs, shown below in a table. As such, our data was inconclusive. However, from simple observation, it appeared to us that when the voltage had decreased, the car itself seemed to run slower. Also, although a graph wouldn't be useful in analyzing only five trials, we observe that with higher voltages, oscillations also tend to increase, possibly attributed to the increased speed. However, once again, our data is insuffficent to draw any conclusions.

Table 4: Results for same constants with changes in Voltage

Oscillation	Acceptability	V
3	7	8.7
2	6	8.8
2	6	8.9
3	9	9.1
5	5	9.2

Next, we considered the effect of higher k_D on oscillation given all else was kept constant. Such data entries are in a table below:

Table 5: Effects of Change in k_D on Oscillations

Initial k_D	Final k_D	Initial Oscillations	Final Oscillations
0	1	4	2
3.5	2.5	1	2
2.6	2.5	2	2
2.5	2.65	3	2
2.8	2.6	3	2
0.5	1	3	2

Despite our limited samples, we can rewrite this table, considering the sign of Initial k_D- Final k_D, and whether oscillations increased or decreased.

Table 6: Effects of Change in k_D on Oscillations

Sign of difference	Decrease or Increase in Oscillations
Negative	Decrease
Positive	Increase
Positive	No change
Negative	Decrease
Positive	Increase
Negative	Decrease

We see from our data that when all other factors in the system are kept constant, as k_D increases, oscillations decrease, and as k_D decreases, oscillations increase. Thus, we establish that these two variables must be inversely proportional to each other in some form.

Interpretation

A notable run was our run on the 28th of May, where we were first testing our piecewise PID. The data for these runs is below:

Table 7: Simple PID results

k_{P1}	k_{D_1}	v_1	k_{P_2}	k_{D_2}	oscillations	acceptability	t	V
8	1.15	25	18	2.65	2	10	26	8.9
8	1.15	50	18	2.65	2	10	21	8.9
8	1.5	50	18	2.65	2	8		8.9
15	0.5	50	18	2.65	2	10	20	8.9
15	0.5	75	20	2.65	3	7		8.9
15	0.5	75	20	2.65	3	9		8.9

We had tried the same PID constants for six runs, and the car only succesfully made the path three out of six times. However, these same constants had worked for their individual sections consistently, so we attempted to figure out what the issue was. We noticed the battery voltage was 8.8 V, and we noted that earlier, we'd seen changes in behavior of our car starting st 8.9 V. Next, we consdidered the fact that perhaps the car was changing in velocity too fast—these rapid changes in veloicty were throwing off the PID controller, as we simply change PID constants instantaneously when changing speeds, as well as change speed instantaneously. We also considered the fact that it was consistently failing at the same spot: the large curve. This could possibly mean that we needed another section with new speeds and PID constants for this section for more consistency. With these three options in mind, we continued testing.

Conclusion

From these runs onwards, we see in our data logs that we changed the batteries, apparent from the voltage increasing from 8.8 V to 9.1 V. Also, we slightly changed our code to add a "ramp-up", or a gradual change in speed rather than instantaneously changing speed, giving us the following runs:

Table 8: Simple PID results

k_{P1}	k_{D_1}	v_1	k_{P_2}	k_{D_2}	oscillations	acceptability	t	V
15	0.5	75	20	2.6	3	6		8.8
15	0.5	75	20	2.6	2	9		9.1
15	0.5	75	20	2.6	3	9		9.1
15	0.5	50	20	2.6	3	10	19	9.1

However, even then, we see that our issue was only partially resolved, as acceptability did indeed go up, but we still did not complete the track consistently. At this point, we see in our data logs that we made the decison to implement our three-part PID controller, which we have seen in our analysis is indeed the most effective at consistently completing the track, with the highest average acceptability.