EECS 461 Final Report

Team Members: Minzhe Zhang, Cheryl Zhang
Date: April 15th, 2016

Introduction

We were assigned the task of developing a simulink model that models a simple, virtual vehicle with features such as adaptive cruise control and automatic steering. We worked on these tasks over the span of three weeks in three separate lab sessions.

During the first week, we implemented the "Vehicle Model" subsystem in our simulink model, which allowed us to control the vehicle manually by using the haptic wheel as our steering wheel and adjusting the potentiometer connected to the ADC to control the vehicle speed.

During the second week, we implemented the "Automatic Cruise Control" subsystem in our model, which could be activated/deactivated by toggling one of the digital input dip switches. When activated, according to the pick lead logic which is described in the following section, the vehicle would operate in either velocity or position mode. We also modified our CAN network to transmit and receive positions and speeds of cars from each lab station. If the vehicle were in velocity mode, the vehicle's speed would no longer be controlled by the potentiometer, but rather by the lower eight digital input dip switches. On the other hand, if the vehicle were within a critical distance behind the lead vehicle (the vehicle immediately in front), position mode would be activated, and our vehicle would maintain a specified distance behind the lead vehicle.

During the final week, we implemented the "Automatic Steering" subsystem in our model, which allowed our vehicle to maintain a certain distance from the center of the road and steer autonomously.

Pick Lead Logic

We used an S-function to implement our pick lead logic. We took in the manual enable signal, the positions and speeds of all cars, and the critical distance H as inputs for our S-function.

To find the lead car, we created a lead_car_i variable that stored the index of the lead car. It was initialized to 0, meaning that our vehicle started as the lead car. We also created an s_temp variable that stored the position of the current lead car. It was initialized to 3000, which was longer than the virtual track length of 2000. We then traversed through all the car positions and checked if there was a car in front of us yet closer to us than s_temp. If so, we would replace lead_car_i and s_temp with the index and position of that car. At the end of the process, if lead_car_i were 0, then we would be the lead car and operate in velocity mode. Otherwise, there would be a lead car ahead of us, and we would compare our velocity and position to the lead car to determine which mode to activate. If the lead car's speed were less than or equal to ours and we were within distance H behind the lead car, then position mode would be activated. Otherwise, velocity mode would be activated. However, if the manual enable signal was 1, neither velocity or position mode would be activated, and the vehicle would operate in manual mode.

Automatic Steering Controllers

The formulas for our PD and PID controllers are as follows:

$$C1 = Kd\_PD \times \frac{Z-1}{TZ} + Kp\_PD$$

$$C2 = Kd\_PD \times \frac{Z-1}{Tz} + Kp\_PD + Ki\_PID \times \frac{T}{z-1}$$

To implement our automatic steering controllers, we used a PD controller in series with a PID controller. First, we initialized the gains of the PD controller and PID controller as the midpoint of each suggested range that was given, and tested the vehicle's stability. By observing the amount of oscillation exhibited by the vehicle, we first tuned the PD controller, then the PID controller, and carefully made sure the vehicle operated smoothly at a speed of 25 mph, especially at sharp turns.

For the PD controller, we first tuned Kp_PD and Kd_PD in order to decrease the rise time without a large amount of overshoot. For the PID controller, we implemented PI and set Kd_PID = 0 to make the system stable and respond as fast as possible. We choose Ki _PID = 0.002, a relatively small value, because a larger Ki _PID would cause greater instability. After that, we tried different Kd_PID values within the range of 50-100 and observed the response of the system. We chose Kd_PID = 75, because although a large Kd_PID could mitigate the negative effects caused by delay, too large of a value would result in more oscillation in the output of the system.

The gains of the PD controller and PID controllers are as follows:

K_{d_PD}	K_{P_PD}	K_{P_PID}	K_{I_PID}	K_{D_PID}
0.03	0.4	550	0.002	75

Encountered Problems

When implementing our lead pick logic, we first tried to traverse through the position vector until we found an arbitrary car that is in front of our car and set it as our lead car. Then we continued to traverse through the position vectors of the remaining cars and replaced the lead car index with the car that was in front of us but behind the current lead car to find the nearest car in front of us. Though our logic seemed correct, the output of our ACC logic test cases did not match up to the expected output. We realized that our break clause did not break us out of our for loop, but rather just the if bracket immediately surrounding the break clause, so the code wasn't doing what we thought it was. In the end, we changed our code to what was explained in the "Pick lead Logic" section.

We also faced issues tuning our PID parameters due to the long sample period and nonlinear behavior of the haptic wheel. It took us a long time to tune the Kp, Kd, and Ki values. Finally we were able to find a way of tuning the PID parameters more efficiently, which was described in the previous section.

Work-load Distribution

Both members collaborated to complete the prelab work together for this project, and contributed equally during lab. This lab report was also an equal, collaborative effort by both members.

Course Evaluation Receipt

Dear Cheryl Zhang,

This message is a confirmation of your Teaching Evaluation submission for EECS 461 015 LAB. It was submitted on Thu Apr 14 17:26:24 EDT 2016. Thank you for helping the University maintain and improve the quality of its teaching.

Please save this message for your records.

Note that all student responses are kept confidential and you may edit your evaluation responses up until the close date for this evaluation.

Dear Minzhe Zhang,

This message is a confirmation of your Teaching Evaluation submission for EECS 461 015 LAB. It was submitted on Thu Apr 14 17:35:40 EDT 2016. Thank you for helping the University maintain and improve the quality of its teaching.

Please save this message for your records.

Note that all student responses are kept confidential and you may edit your evaluation responses up until the close date for this evaluation.