Christopher Ho

500840595

[Christopher.j.ho@ryerson.ca](mailto:Christopher.j.ho@ryerson.ca)

Thursday December 17, 2020

CPS 607 Final Exercise Fall 2020 – NOT Tesla’s Autopilot

The first feature is enhanced cruise control. My approach for this algorithm is quite simple, every iteration I take a reading and check the distance directly in front of me using my ultrasonic sensor. I have four distance brackets and depending on how far the object is away from me, I’ll either adjust my speed or change lanes. If the object is < 30cm away I deem it too close and my vehicle changes lanes. Initially, it starts out on the right lane. If the object is < 40cm I slow down my vehicle to a speed of 80. If less than 100, I set my speed to its original default of 100. Any distance greater than 50 and I set my speed to 150.

For switching lanes, I used lines sensors to track how many lanes I have crossed in order to avoid using timing or delays and guessing how long it takes for the vehicle to move forward. Once an object is < 30cm away the vehicle switches lanes. To start the procedure, I turn left for 1.2 seconds then move forward until my left line sensor is triggered. At this point I’m in the middle lane with my left sensor touching the right boundary of the left lane. I then turn right to straighten my vehicle. My vehicle is now in the middle in between “lane” facing forward. I repeat the sequence of steps again to end up on the left lane. Changing to the right lane is the same logic just with inverted actions (switching left to right and right to left).

Another aspect of cruise control is staying in its current lane while being in near-continuous motion. To achieve this, I took the principles of line following and inverted them continuing to always move forward while keeping my vehicle bounded between the two lines of tape representing my current lane. Here is how I accomplished this behaviour. First is the case to move forward, I wrote that if all three lines sensors (left, middle, right) are NOT triggered (touching a lane boundary line), then move forward. If the left line sensor is triggered, move right while its still touching the left line. If the right line sensor is triggered, then turn left while its still touching the right line. The ending result of this inverted line following logic is that the vehicle moves around the track staying in its lane and adjusting to bends or turns.

The second feature is self-parking, specifically parallel parking. I wanted this to mirror the steps an actual car would follow in order to parallel park. My vehicle’s ultrasonic sensor is directly facing right (0 degrees), if an object is < 15cm I begin the parallel park sequence. I start by turning left for 1.2 seconds. I then reverse for 1 second. I then turn right until my middle sensor is triggered. At this point my vehicle is facing straight, then I turn my servo directly in front of the vehicle at 90 degrees to measure the distance in front of me and make adjustments forward or back until 5cm < distance < 10cm and stop. I used boolean values to determine what action I should take next with my last state “final position” set to true and my car parked.