Professor Fearing EECS192/Assignment #2 Car Steering Spring 2016

Due by 6 pm Fri. April 1 on BCourses in .pdf.

(One assignment per team.)

For this assignment you will be using `V-rep` for dynamic simulation of a car. (V-rep Pro EDU V3.3 is available in 204 Cory, and can be downloaded for Mac/Ubuntu/Windows at `http://www.coppeliarobotics.com/downloads.html`. The car model `car-scale.ttt` is on Piazza. V-rep is the server, and you will modify a Python function `control_loop`() to drive the simulated car. The simulator estimates `lat_err` = `yDist` = y_a, the lateral error from the track at a distance approximately 2 car lengths in front, and takes 2 inputs: 1) commanded steering angle `steerAngle` = δ and 2) longitudinal velocity `xSpeed` = V. The V-rep simulation server needs to start before `carTest.py`. `carTest.py` will run the car but it is not tuned. You will modify and extend the code to get a good car controller. The simulator runs ok at 10 ms time step (dynamics are updated at 10X).

For all plots, time axes should be in seconds, and lateral errors in m or cm. Do not use a white-on-black screen dump (as it looks unprofessional). Matplotlib (see iPython notebook from hw1) is suggested to make nice plots. Combine all plots into a single .pdf file to upload.

For each part below, plot on 1 page a) actual x vs y position of car b) lateral error y_a as function of time, c) steering angle δ as function of time. For part 4, below, also plot d) car longitudinal velocity as a function of time. For each part, list constants used. (k_p should have units of radians/meter, etc.)

(25 pts) 1. Steering Simulation- proportional control

(22) a. Using pure position control, $\delta = k_p y_a$, choose a fixed speed V and k_p that allows the car to successfully complete the track without hitting any cone(s). Report k_p and V.

(3) b. Specify worst-case overshoot (cm), and note on plots where this occurs.

(25 pts) 2. Steering Simulation- proportional + derivative control

(20) a. Using PD control $\delta = k_p y_a + k_d \dot{y}_a$, choose a fixed speed that allows the car successfully complete the track without hitting any cone(s). Report k_p, k_d, and V. (Estimate $\dot{y}_a \approx \frac{y_a[n]-y_a[n-1]}{20ms}$.)

(3) b. Specify worst-case overshoot (cm), and note on plots where this occurs.

(2) c. Briefly comment on any differences in performance observed between P and PD type control, such as overshoot or maximum speed.

(25 pts) 3. Sensor Resolution Effect

(10) a. For your line camera mounting location and orientation, estimate track position resolution Δ (in cm) which is limited by the 128 elements. (Optionally, use your algorithm from HW1). Standard deviation due to position quantization can be estimated as $\sigma = \frac{\Delta}{\sqrt{12}}$.

(15) b. Repeat problem 2 above, but adding the noise into estimated y_a.

i.e. `yDist = yDist + np.random.normal(0.0, sigma)`.

(25 pts) 4. Steering simulation with PD, quantization, and speed control

(15) a. Modify `carTest.py` to use car speed control rather than a fixed speed. For example, decelerate when the lateral error is large, and speed up when tracking well. (Or choose a velocity setpoint depending on error.) Plot best car performance.

(5) b. Specify the velocity controller used, and include relevent controller code section.

(5) c. What is worst case overshoot? What is laptime? Has the best time improved compared to fixed car speed?