PID and beyond

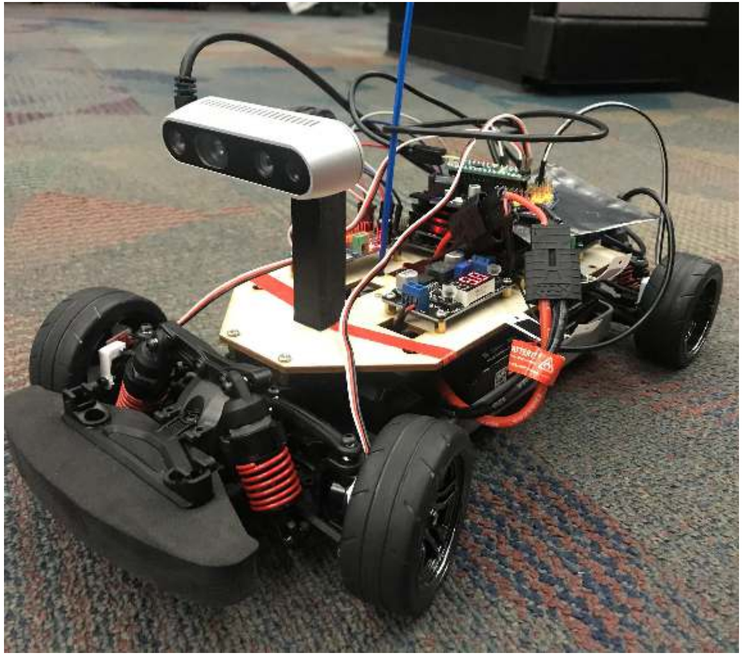
Control Approaches for Autonomous Racing

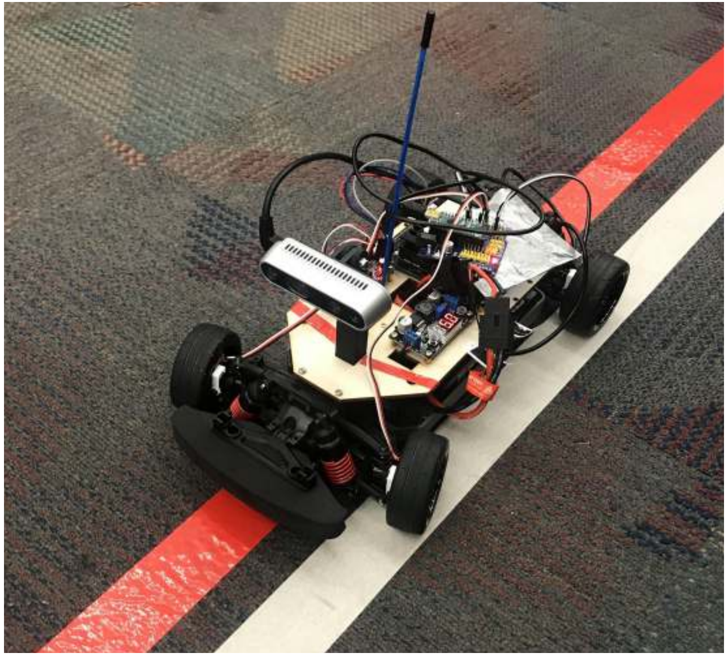
Mike Estrada

Berkeley
UNIVERSITY OF CALIFORNIA

[redacted]

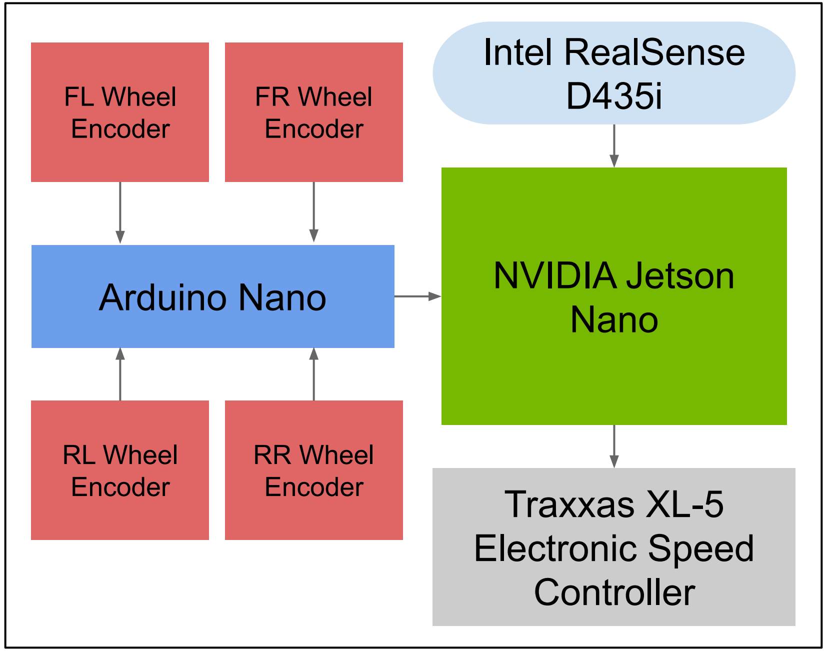
Setup



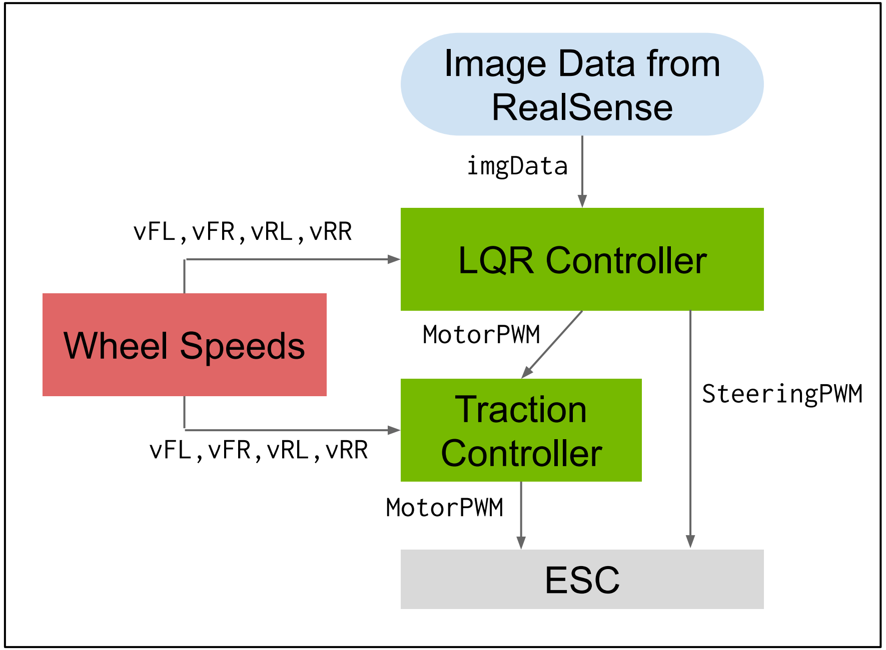


Architecture

HARDWARE

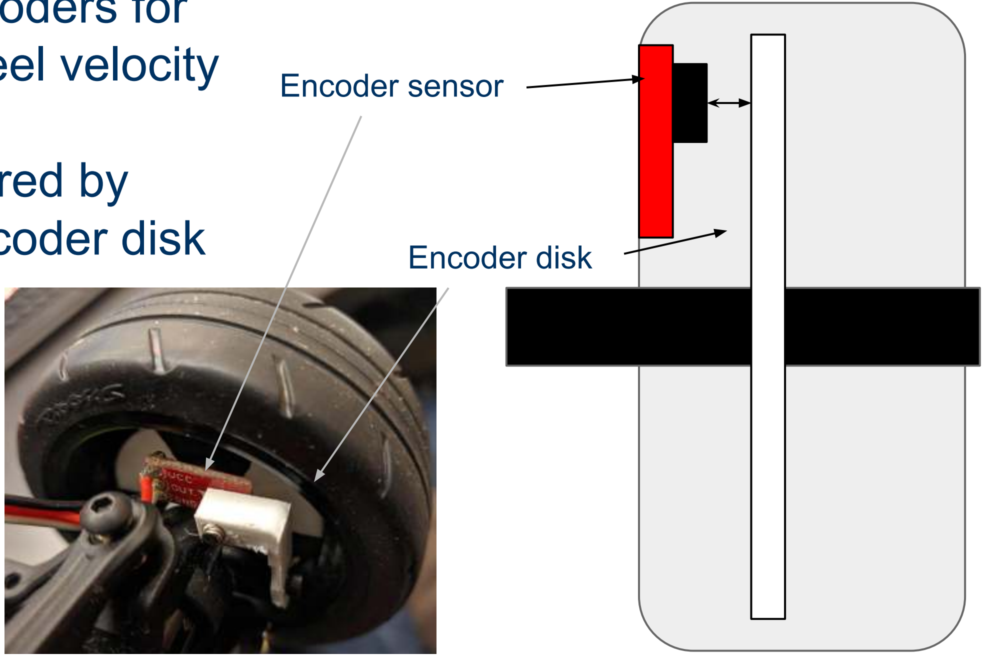


SOFTWARE

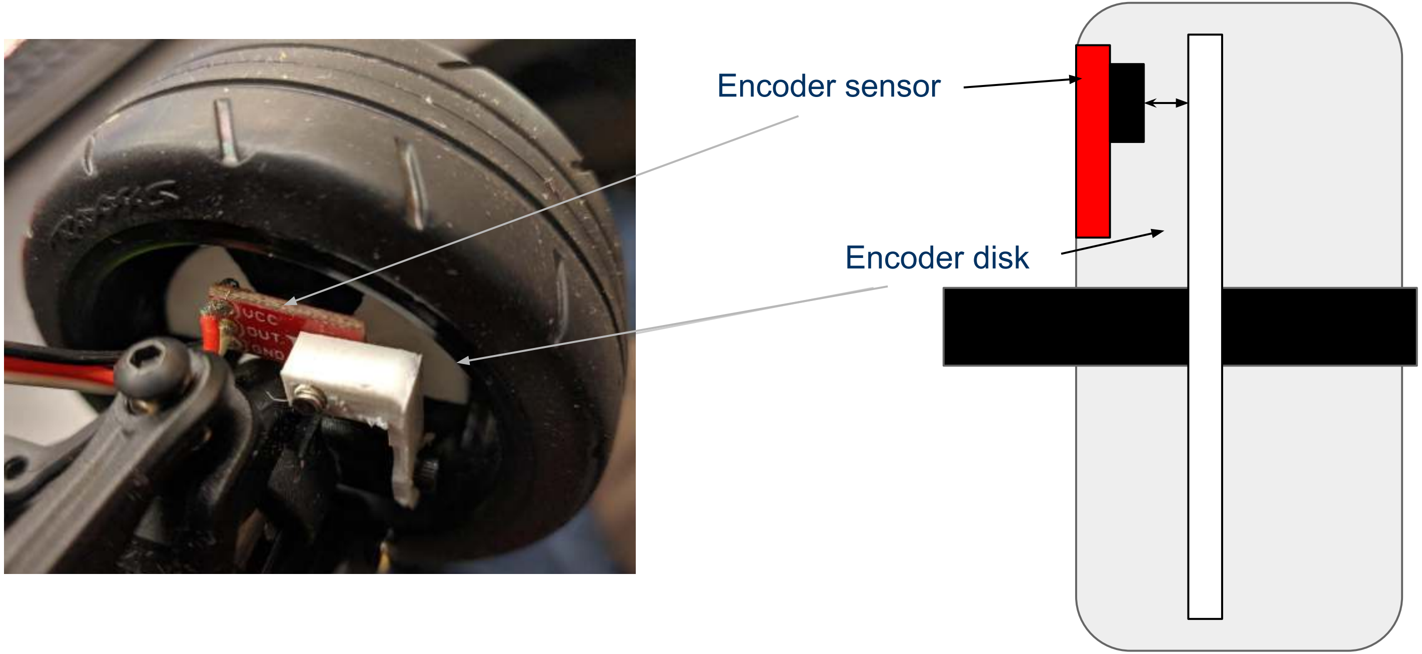


Wheel Encoders

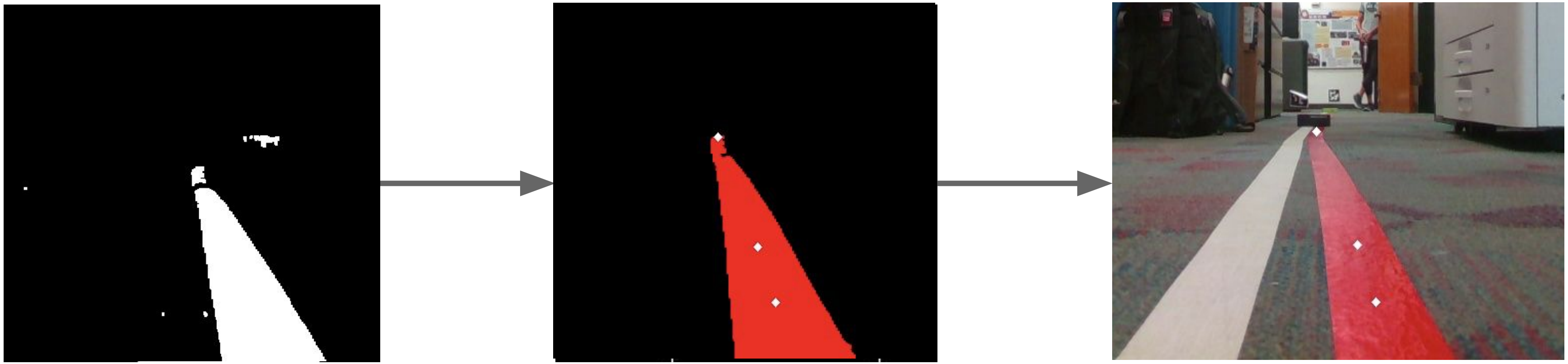
- Designed, prototyped custom encoders for independent measurement of wheel velocity
- Encoders use a line sensor triggered by change from black to white on encoder disk
- Independent wheel velocities calculated by Arduino Nano
- Velocities communicated to Jetson via serial port



Wheel Encoders



Lane Tracking

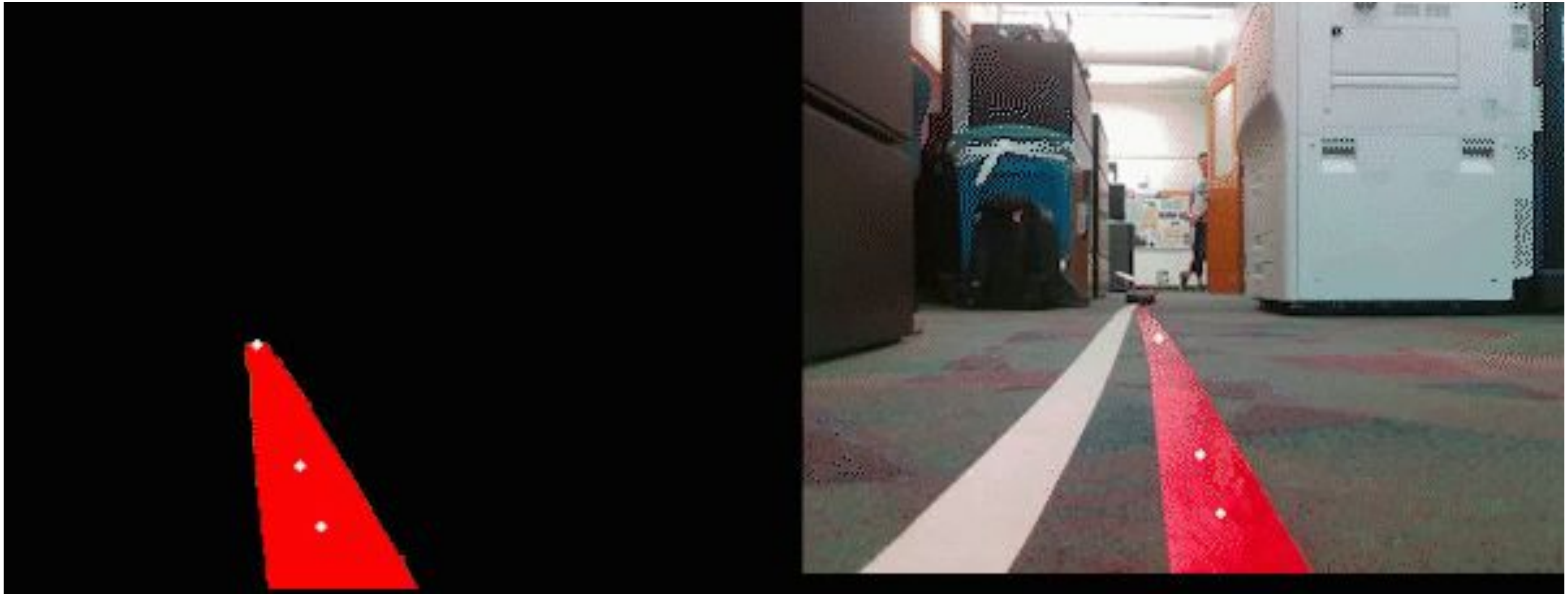


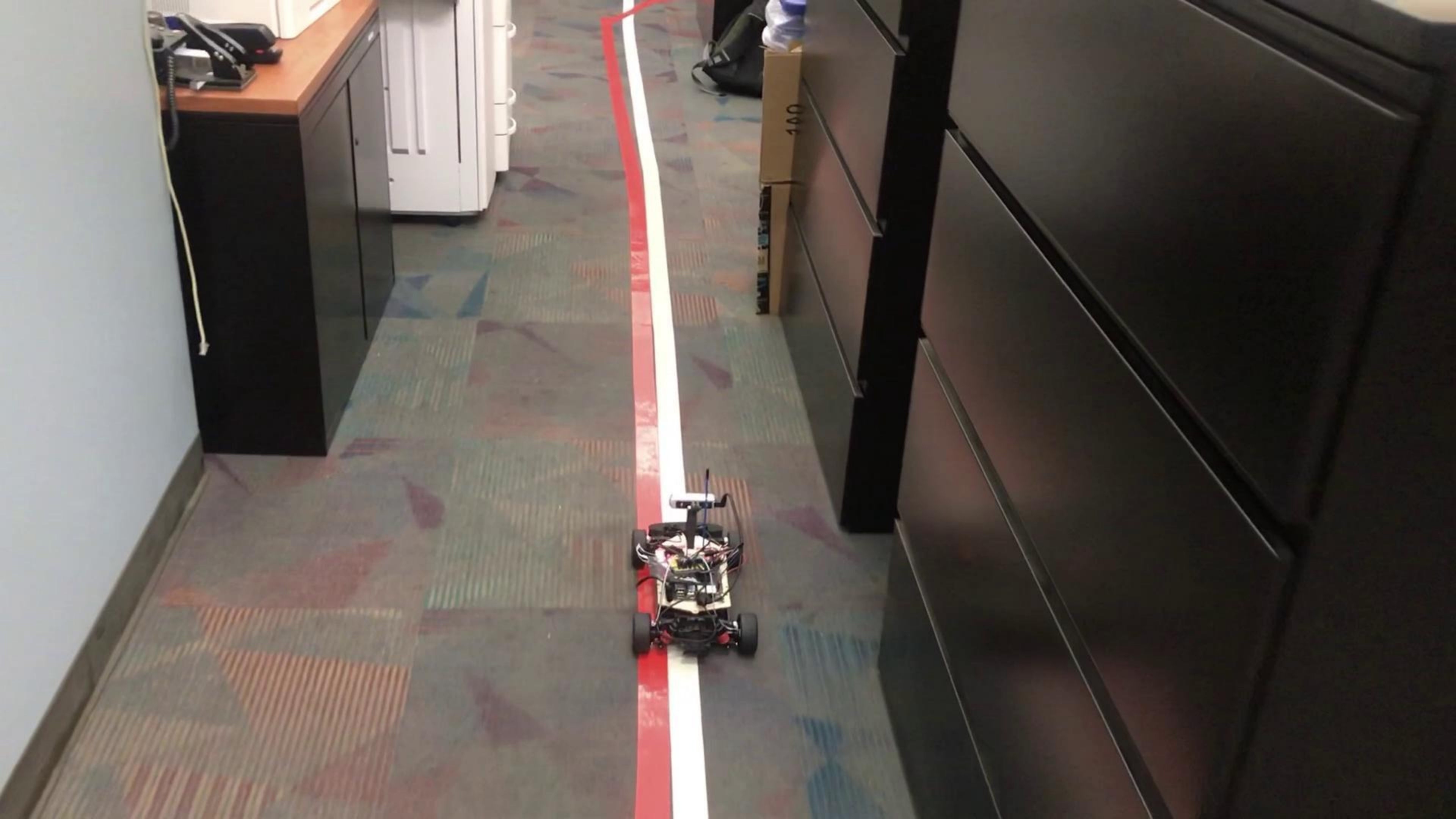
Step 1: Set the threshold in HSV space to get the red part in the video

Step 2: Get the lane by finding the largest connected component

Step 3: Find the three way points on the Lane by averaging the coordinates with certain y-value

Lane Tracking





Simplified Bicycle Model

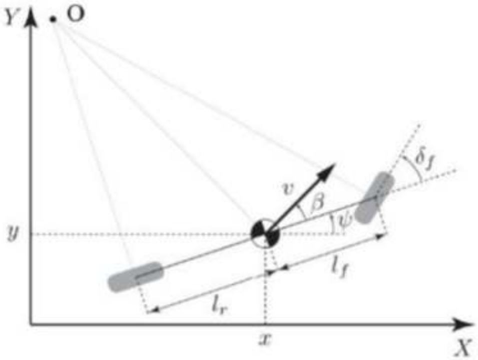


Figure 1: Simplified Kinematic Bicycle Model

Recall the simplified kinematic bicycle model shown in Figure 1:

$$z_{ref} = \begin{bmatrix} x_{ref} \\ y_{ref} \\ \psi_{ref} \end{bmatrix} = \begin{bmatrix} x_{ref} \\ y_{ref} \\ \frac{v_{ref}}{R}\Delta t \end{bmatrix} \qquad \begin{aligned} \dot{x} &= v\cos(\psi+\beta) \\ \dot{y} &= v\sin(\psi+\beta) \\ \dot{\psi} &= \frac{v}{l_r}sin(\beta) \end{aligned}$$

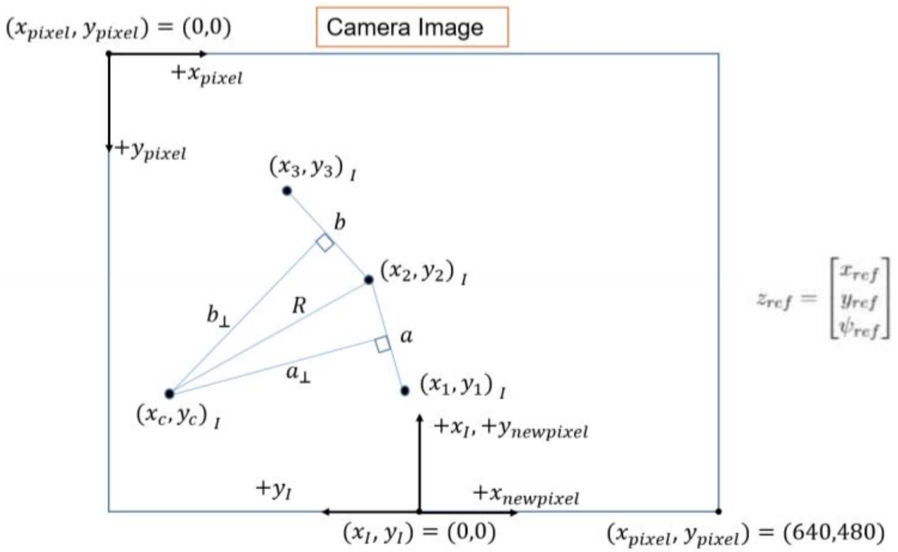
$$u_{ref} = \begin{bmatrix} v_{ref} \\ \beta_{ref} \end{bmatrix} = \begin{bmatrix} v_{ref} \\ \sin^{-1}(\frac{l_r}{R}) \end{bmatrix} \qquad \delta_f = \tan^{-1}\left(\frac{l_r + l_f}{l_r}\tan(\beta)\right)$$

$$f_1(z,u) = u_1\cos(z_3+u_2)$$
$$f_2(z,u) = u_1\sin(z_3+u_2)$$
$$f_3(z,u) = \frac{u_1}{l_r}sin(u_2)$$

$$A = \begin{bmatrix} \frac{\partial f_1}{\partial z_1} & \frac{\partial f_1}{\partial z_2} & \frac{\partial f_1}{\partial z_3} \\ \frac{\partial f_2}{\partial z_1} & \frac{\partial f_2}{\partial z_2} & \frac{\partial f_2}{\partial z_3} \\ \frac{\partial f_3}{\partial z_1} & \frac{\partial f_3}{\partial z_2} & \frac{\partial f_3}{\partial z_3} \end{bmatrix}\Bigg|_{\substack{z=z_{ref} \\ u=u_{ref}}}$$

$$B = \begin{bmatrix} \frac{\partial f_1}{\partial u_1} & \frac{\partial f_1}{\partial u_2} \\ \frac{\partial f_2}{\partial u_1} & \frac{\partial f_2}{\partial u_2} \\ \frac{\partial f_3}{\partial u_1} & \frac{\partial f_3}{\partial u_2} \end{bmatrix}\Bigg|_{\substack{z=z_{ref} \\ u=u_{ref}}}$$

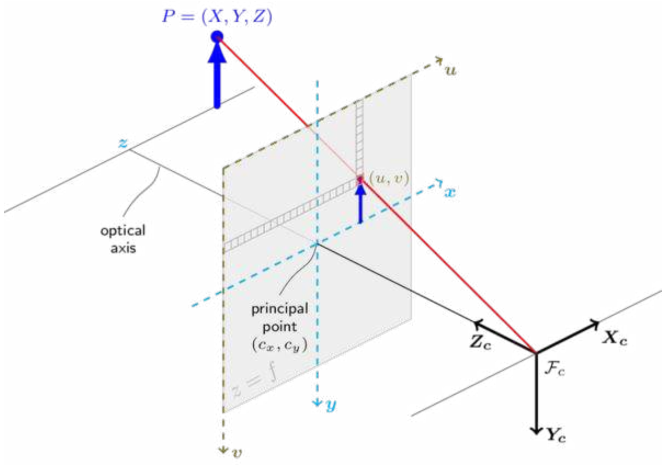
State Estimation



- Waypoints give x and y coordinates along the track relative to the car
- Radii of turns are calculated to determine a relative inertial heading (ψ)

State Estimation

$$s\begin{bmatrix}u\\v\\1\end{bmatrix}=\begin{bmatrix}f_x & 0 & c_x\\0 & f_y & c_y\\0 & 0 & 1\end{bmatrix}\begin{bmatrix}r_{11} & r_{12} & r_{13} & t_1\\r_{21} & r_{22} & r_{23} & t_2\\r_{31} & r_{32} & r_{33} & t_3\end{bmatrix}\begin{bmatrix}X\\Y\\Z\\1\end{bmatrix}$$



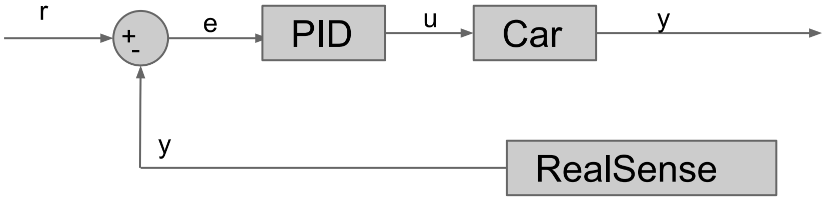
Calculating world coordinates from pixel coordinates

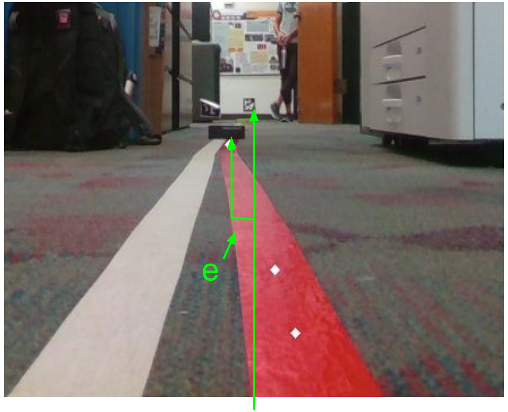
- From the RealSense camera, we have access to the intrinsic matrix for the camera
- Solve for [X Y Z], Z (depth) given by camera

“Smart” PID Controller

The first iteration for our lane following controller was a “Smart” PID Controller:

- Constant straight line velocity
- Tries to control the x position of the furthest waypoint to be 0 (inline with car)
- Increasing radius lowers velocity





MPC controller w/dLQR

- Initially solved MPC problem with dLQR using Ricatti equation

$$\min_{x,u} \sum_{i=0}^{N-1} [(x - x_i)^t Q (x - x_i) + u^t R u] + (x - x_f)^t P_t (x - x_f)$$

$$s.t.: L_f h + L_g h u + \lambda h \geq 0$$

$$x_{i+1} = A x_i + B u_i$$

$$x_0 = x(0)$$

$$Q, R, P_t > 0, diagonal$$

$\Rightarrow$

$$\min_{x_{qp}} \frac{1}{2} x^t P x + q^t x$$

$$s.t.: G x_{qp} \leq h$$

$$A x_{qp} = b$$

$$x_{qp} = [x_0, \dots, x_N, u_0, \dots, u_{N-1}]^t$$

$$P = block\_diagonal[Q, \dots, Q, P_t, R, \dots, R]$$

$$q = [-2x_0 Q, \dots, -2x_{N-1} Q, -2x_f P_t, 0, \dots, 0]$$

- Inclusion of CBF requires constrained optimization
- Converted standard dLQR with state tracking into QP form

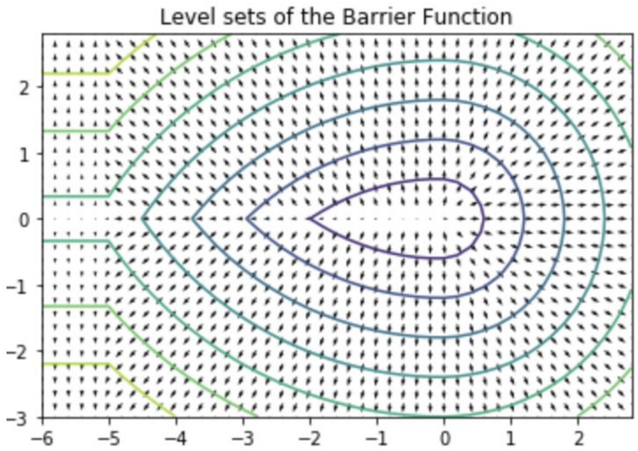
Control Barrier Function

Reachable Set for 12 cm radius cylinder

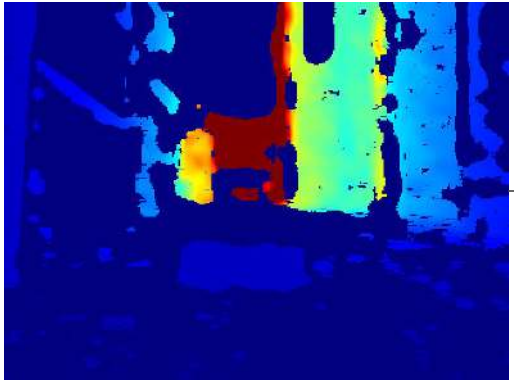
Using code from Sylvia Herbert in Claire Tomlin’s Lab

Visualization of Level sets

Using code from David McPherson

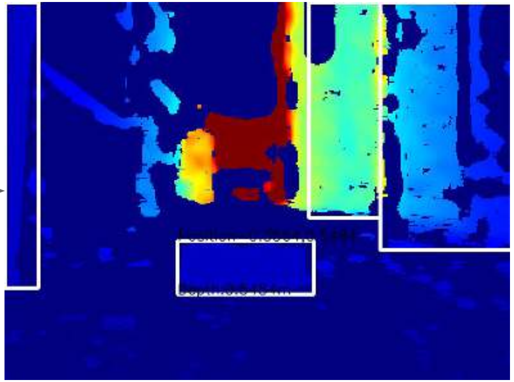


Obstacle Detection



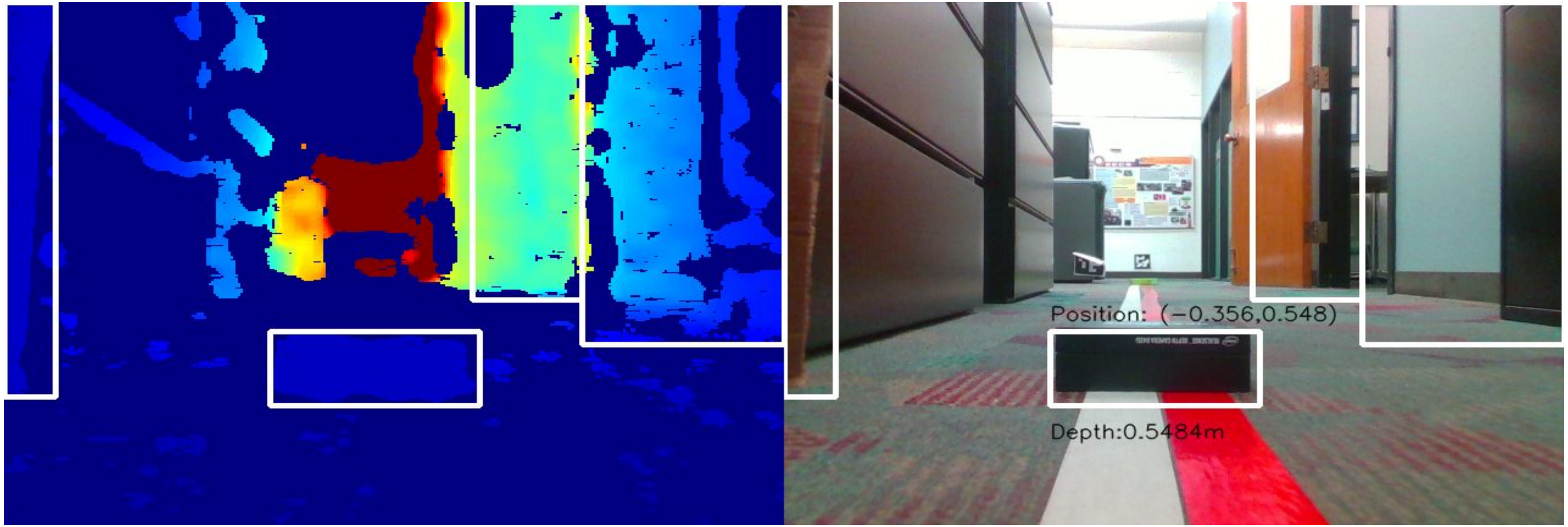
Step 1: Start with Realsense Camera depth data. Image colorized for visualization purposes

Step 2: Perform image segmentation based on the distance from objects to the camera



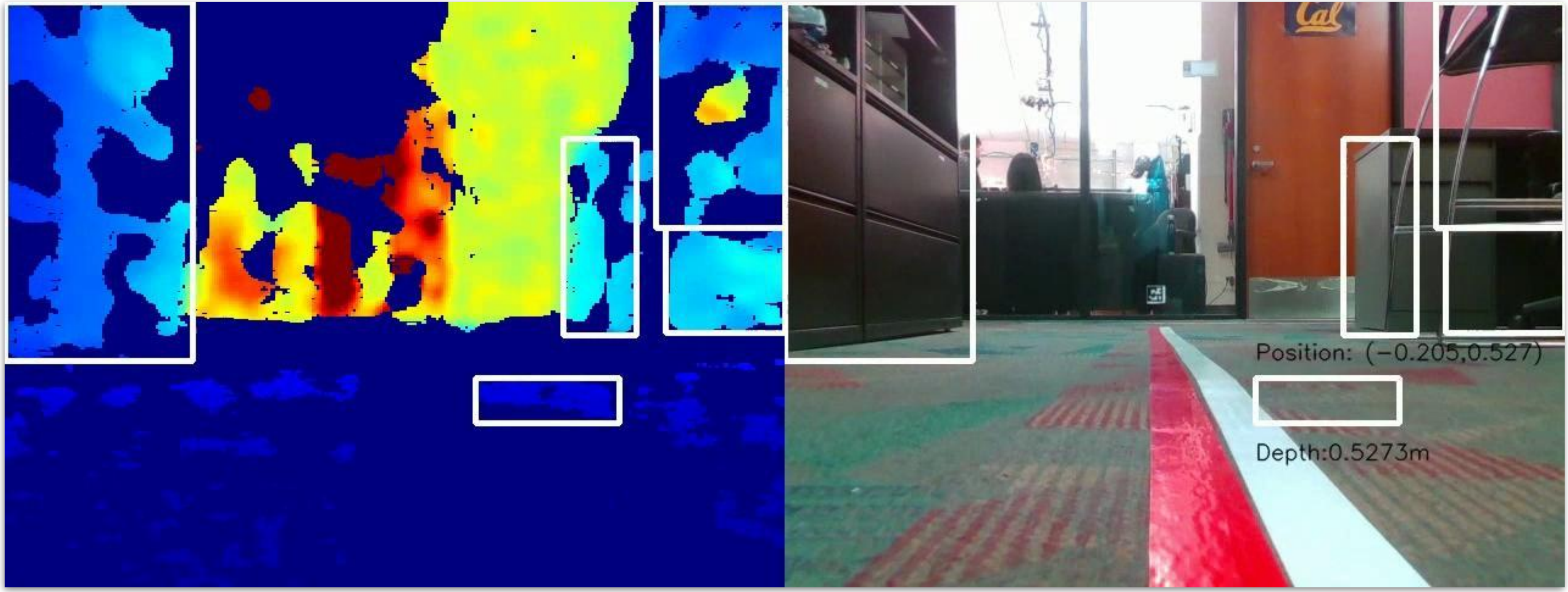
Step 3: Find contours and bounding boxes of all objects (white areas in step 2) after filtering by object position, size, distance, and object height

Obstacle Detection



Berkeley
UNIVERSITY OF CALIFORNIA

Obstacle Detection



Appendix

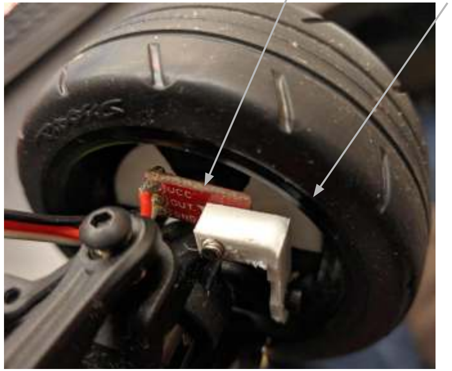
Goals

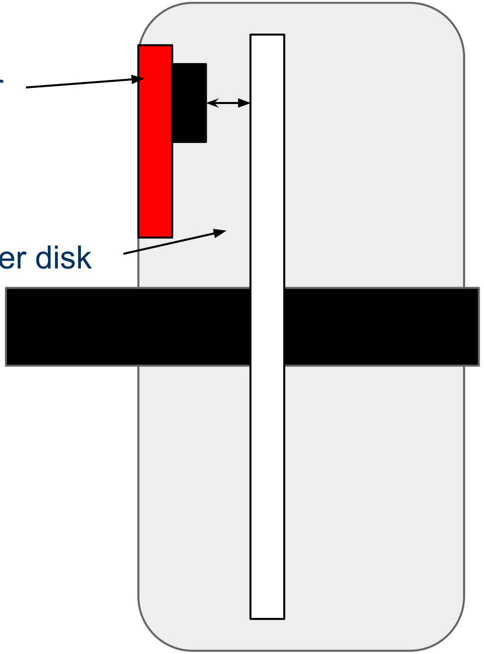
Build an autonomous Traxxas RC car capable of navigating a track at high speed by

1. Using computer vision to detect lane markers and obstacles
2. Implementing an MPC controller and control barrier function (CBF) to follow the track and avoid obstacles
3. Implementing a traction control algorithm to maximize vehicle traction and acceleration

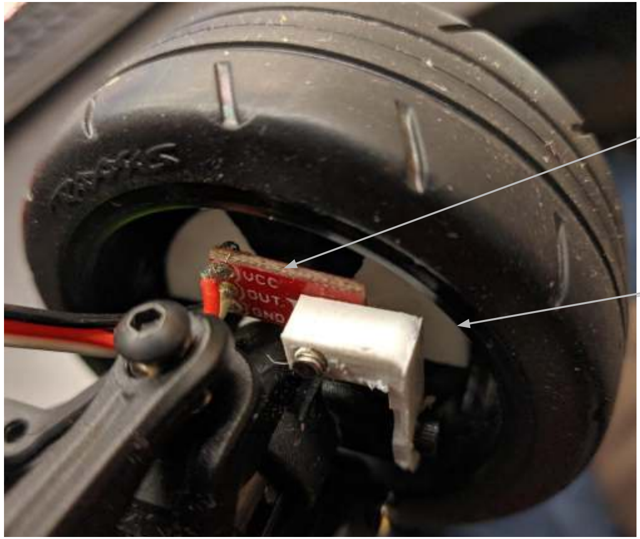
Wheel Encoders

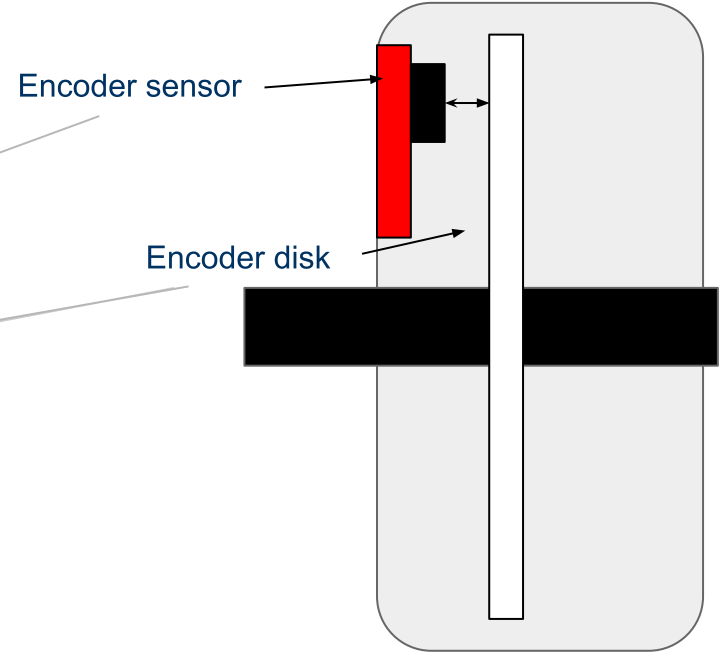
- Designed, prototyped custom encoders for independent measurement of wheel velocity
- Encoders use a line sensor triggered by change from black to white on encoder disk
- Independent wheel velocities calculated by Arduino Nano
- Velocities communicated to Jetson via serial port





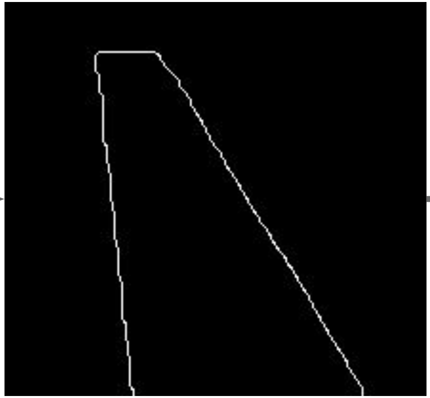
Wheel Encoders



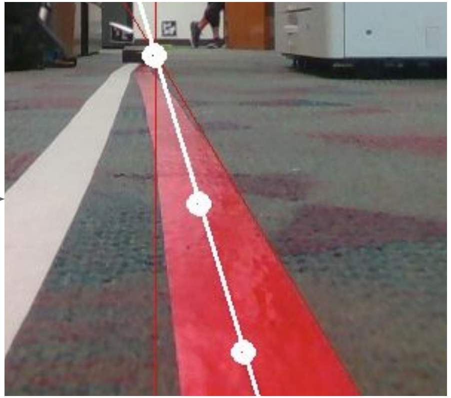


Lane Tracking - Initial Method

Step1: Set the threshold in HSV space to get the red part (Followed by extra image processing)



Step2: Extract the edge of lane using a canny edge detection



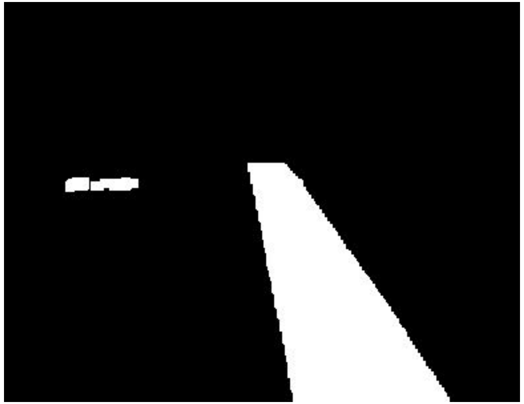
Step3: Detect two edges of the lane (thin red lines), get the middle line (the bold white line), select three points on it.

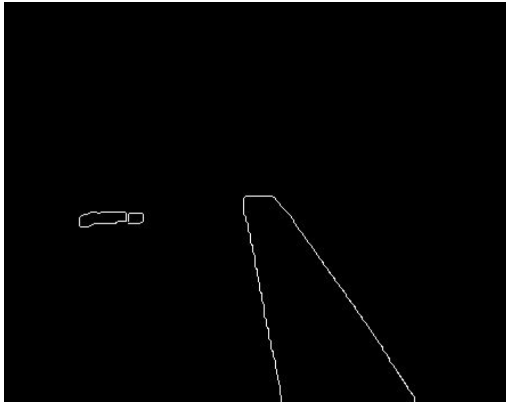
Lane Tracking

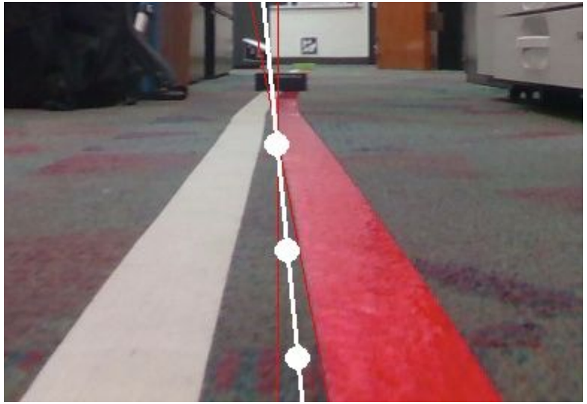
Video for Initial Method

Lane Tracking

- Existing Problems in Initial Method:
  - Easy to be influenced by noisy pixels
  - Two edges of the lane are not accurate, the point might shift out of the lane







Berkeley
UNIVERSITY OF CALIFORNIA

Lane Tracking

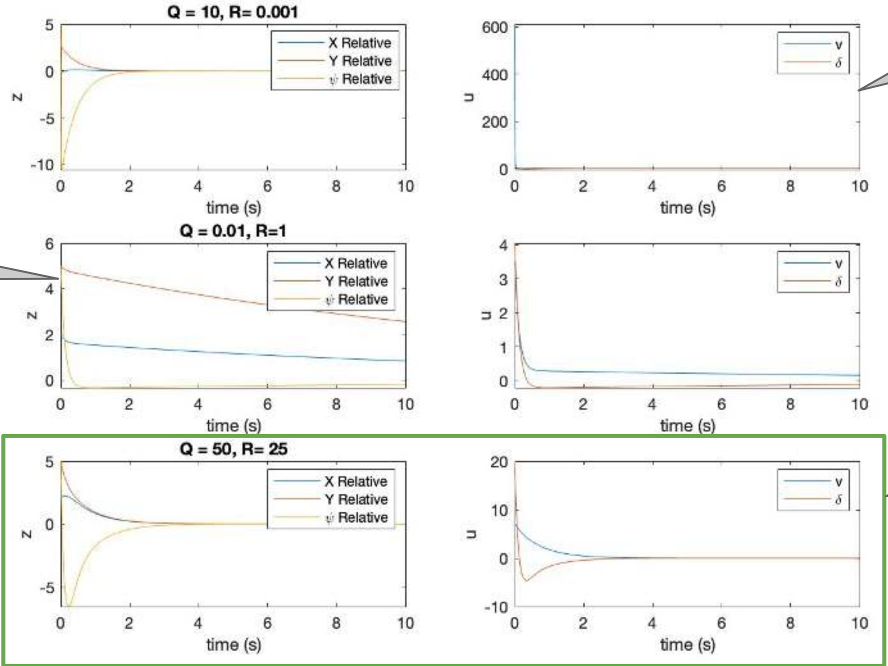
Video in RBG

Video showing processed img

Lane [redacted]

LQR Controller: Tuning

Simulated LQR Controller on a Simplified Bicycle Model



Unachievable Inputs on Real Car

Long Time to Achieve Goal

Chosen Q and R Values Considering Input Thresholds

Berkeley
UNIVERSITY OF CALIFORNIA

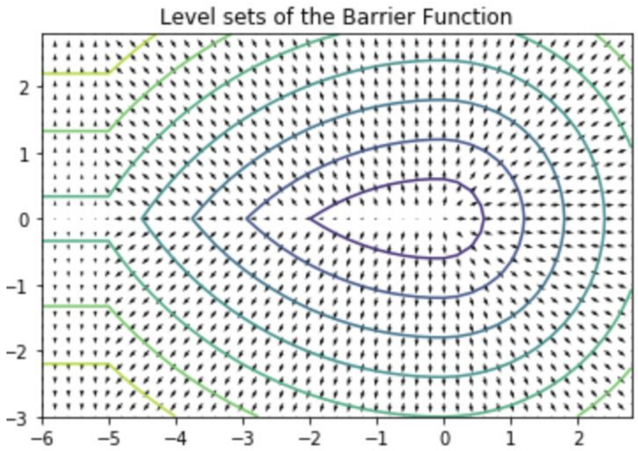
Control Barrier Function

Reachable Set for 12 cm radius cylinder

Using code from Sylvia Herbert in Claire Tomlin’s Lab

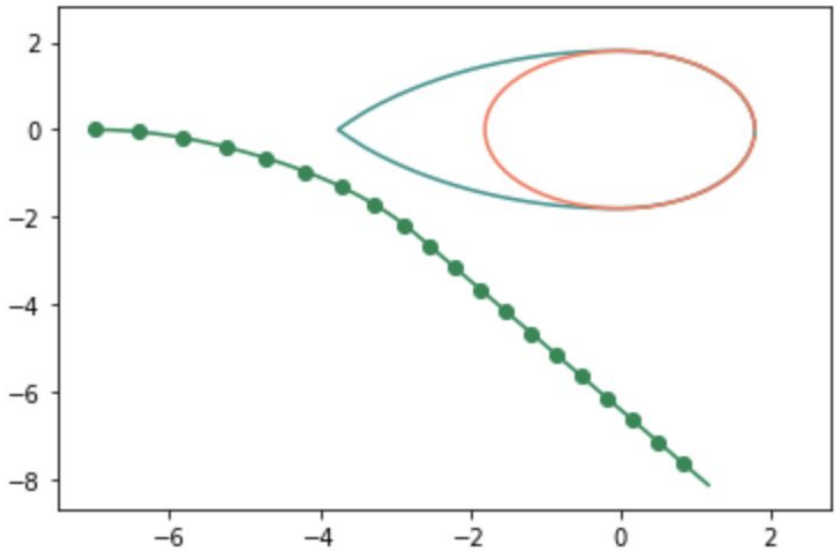
Visualization of Level sets

Using code from David McPherson

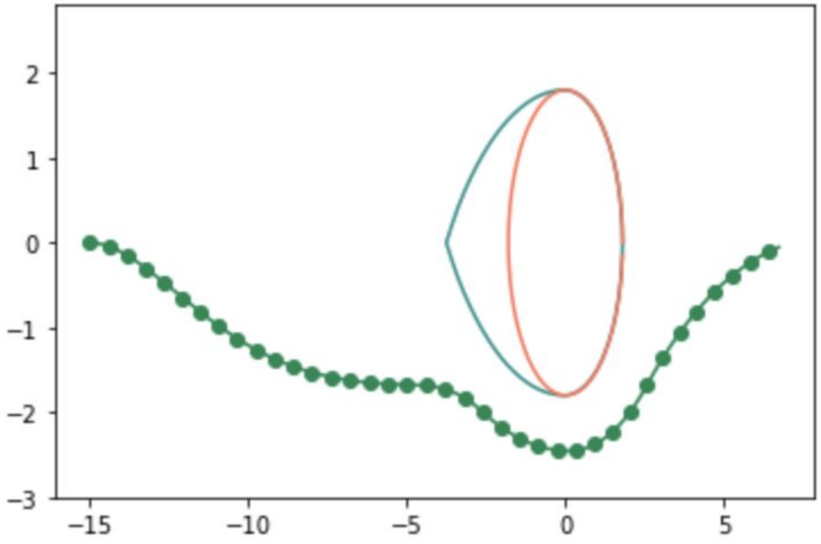


Control Barrier Function

Safe Control Path



Safe Path/LQR Hybrid Controller



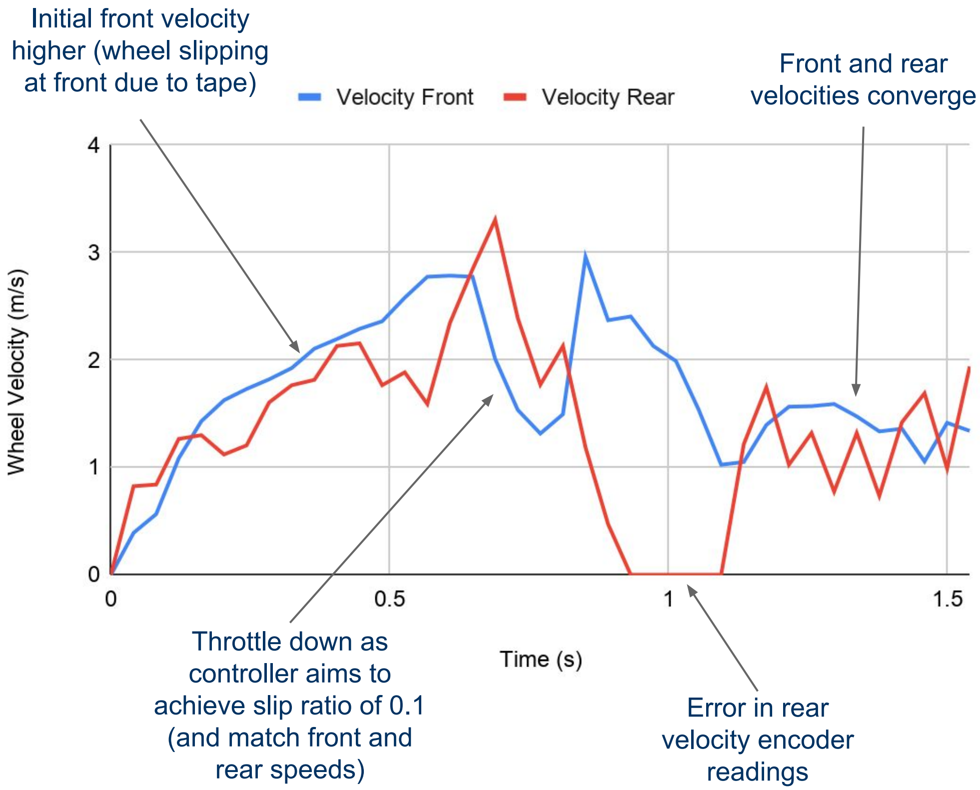
*Plots adapted from David McPherson's code

Traction Control

No traction control: example of wheel slip

Traction Control

- Constant throttle PWM value sent to ESC
- Front wheel velocity and rear wheel velocities recorded



Berkeley
UNIVERSITY OF CALIFORNIA