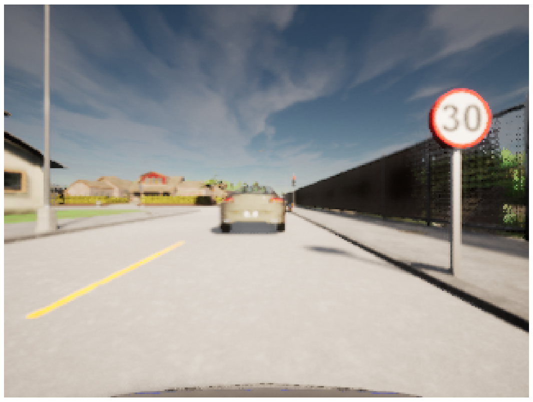
Robot Learning HW 2 Report

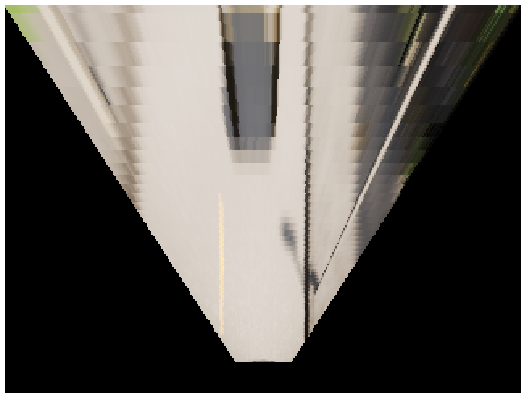
Linden Adamson & Jason Calalang

2.1 Lane Detection

A: Homography Transfer

Using sample data from the last assignment, we were able to transfer images from carla into BEV images



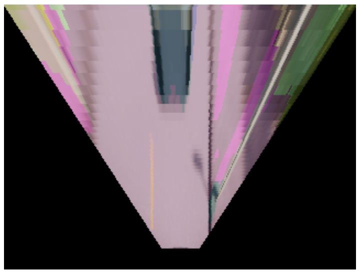
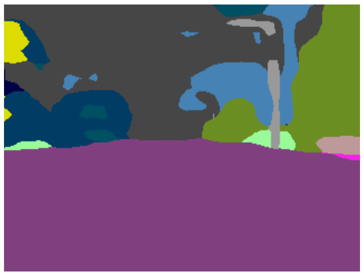
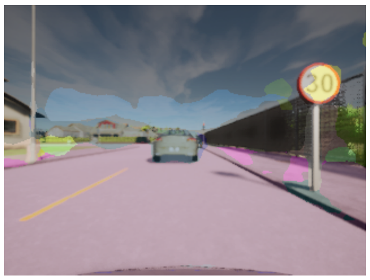


B: Edge Detection

Our BEV already cut out anything above the horizon, so while a cut is implemented, it is not used and the BEV image is turned to grayscale only

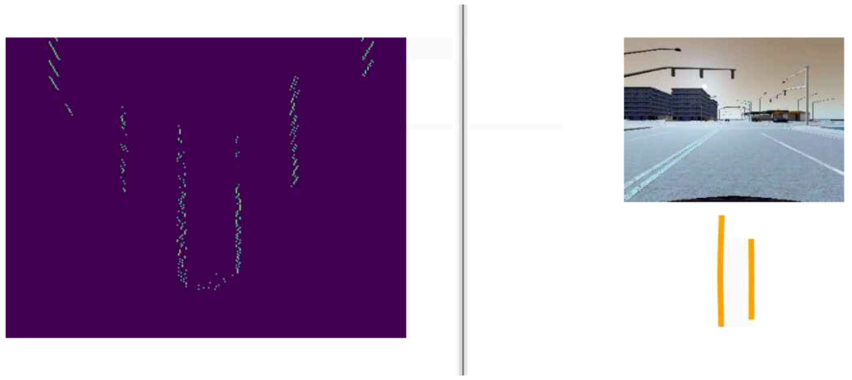
C: Obstacle Detection

To make obstacles more clear, we employed a real-time segmentation model that we trained on the cityscapes dataset and used BiSeNet v2. The first image is the segmentation model applied to the BEV image, the second is the objects on one frame from carla, and the third is the test image with the segmentation model applied.

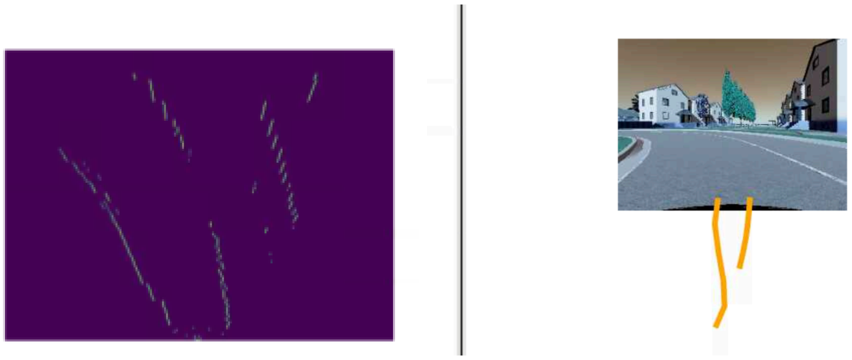
D: Assign Edges to Lane Boundaries

We use a custom method we found worked better in most base situations. We divide the output of the edge detection into the right and left side of the screen, then use a modified k-means algorithm to put the edge data into clusters centered around their x value. After cleaning out outliers, the cluster closest to the center of the screen is output as the lane boundary. On the left you see the edges as seen by the dp-means algorithm, and on the right the splines atop the image they're fit to.

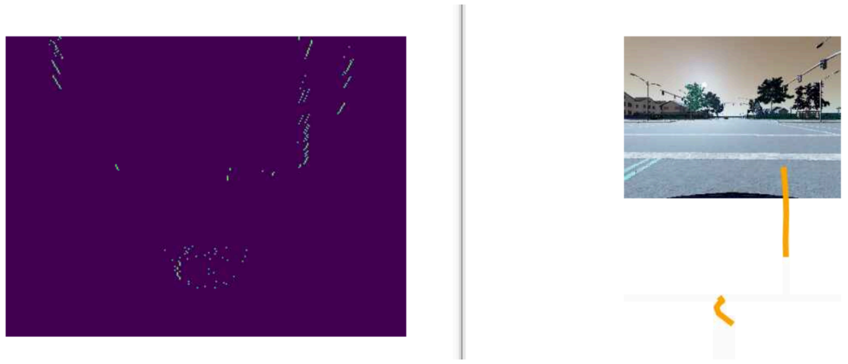


F: Testing

Our lane detection algorithm can struggle with cases such as traveling around this roundabout.

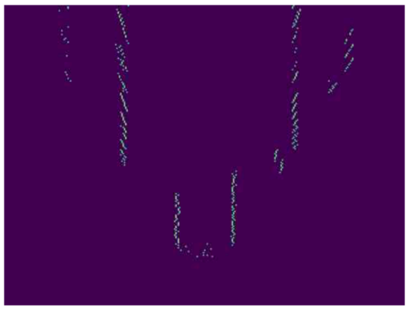


Additionally, the model can struggle when dealing with intersections where the continuity of lines stops.



2.2 Path Planning

A: Road Center



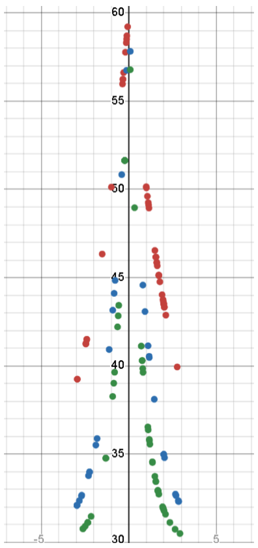
Here is a plot running the simulation. The blue dots are centered waypoints while the orange lines are the splines

B: Path Smoothing

The second term, as described in the report, minimizes the angle between two consecutive line segments. It does so by calculating the dot product of the segments as normalized vectors and then summing all of them to represent all the angles in a consecutive segment. As the term gets minimized, the dot product and the angles also get minimized, so the path as a whole will become smoother.

C: Target Speed Prediction

In this case, the goal is to increase the speed to near v_max (60) when straight (curvature is low) and decrease the speed to a minimum of v_min (30) when curving. To get a more responsive and accurate target speed, we made adjustments to K_v.
Here is a plot of the target speed assumptions over different parameters. Red = 0.4, Blue = 0.9, and Green = 1.4. We found that the red is the most optimal for faster speeds since a majority of the time the road will be straight so higher speeds are favored



2.3 Lateral Control

A: Stanley Control Theory

$$\delta_{SC}(t) = \psi(t) + \arctan\left(\frac{k \cdot d(t)}{v(t)}\right)$$

The goal of the Stanley Control Theory is to command steering so that d(t), or the cross-track error, converges to 0. The cross-track error is the distance between the center of the front wheels and the nearest point on the trajectory. In Stanley Control, the cross-track error optimally converges exponentially. ψ(t) denotes the angle of the nearest path segment, so in an error-free situation it would make sense if d(t) is 0 and **δ**_sc(t) = ψ(t) or the front wheels are parallel to the path. K is a gain coefficient, and the larger the error, the larger the proportional response will be toward the trajectory.

B: Stanley Controller

The gain we arrived at was .5, an order of magnitude lower than what was previously default. We found that this reduced the amount of clipping, giving much more reasonable steering values. The vehicle's behavior can still be erratic, however, in some cases overreacting to otherwise small errors in the waypoints.

C: Damping

We found the default value of .6 to be the best performing. This value helps keep the control angles relatively smooth by shifting the new control angle somewhat near the one before it by a fraction of the difference. This also helps against overcorrection where the control angle zigzags around the path.

2.4 Longitudinal Control

B: Parameter Search

The parameters we arrived at: KP=0.01, KI=0.001, KD=0.01

2.5 Competition

Test_agent.py, modified from the provided file, will drive based on waypoints from an rgb image.