Environment-Perception-For-Self-Driving-Cars

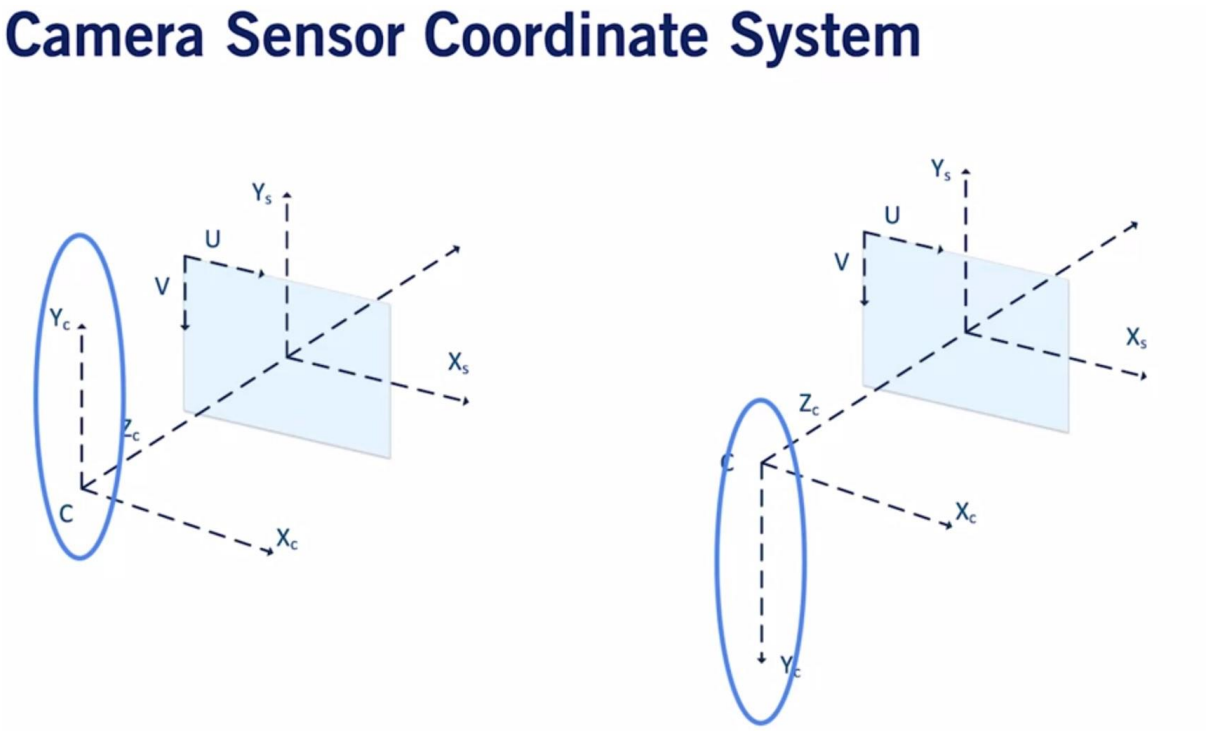
Using the output of semantic segmentation neural networks to implement drivable space estimation in 3D.

Using the output of semantic segmentation neural networks to implement lane estimation.

Using the output of semantic segmentation to filter errors in the output of 2D object detectors.

the filtered 2D object detection results to determine how far obstacles are from the self-driving car.

The Camera Sensor



All the required 3D estimation will be performed in the camera coordinate frame

The camera as a sensor is usually oriented with its y-axis pointing downward.

The only thing you will need to be careful about, is the sign of the height value of pixels.

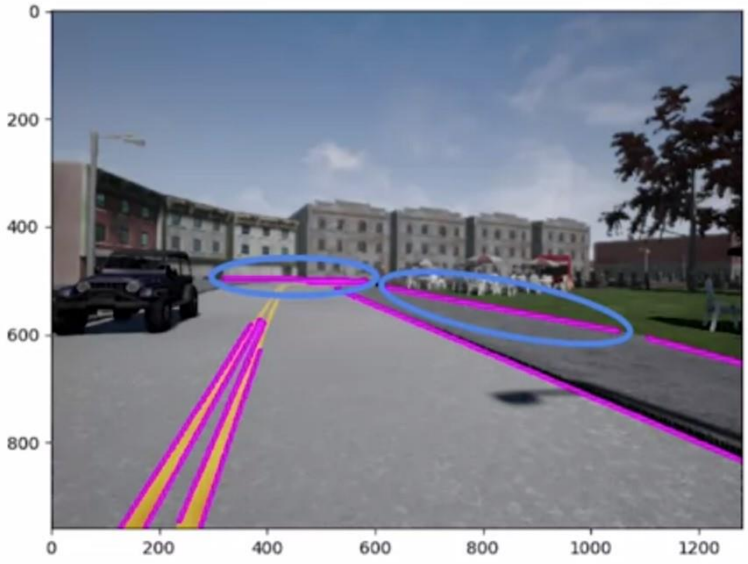
Points higher than the camera will have a negative height, while points lower than the camera will have a positive height.

Plane Estimation

- compute_plane(xyz):
  - Custom function provided to compute a plane estimate using SVD
- numpy.linalg.lstsq(A, b):
  - Solves the equation Ax = b by minimizing the Euclidean 2-norm
  - Be careful of minimum number of points required when randomly choosing points for RANSAC
  - Might fail if chosen points result in a poorly conditioned system
- numpy.linalg.svd(A):
  - Better numerical stability, and reliability over normal equations
  - SVD does not provide the final solution, you will need to manipulate the results yourself

Filtering Horizontal Lines

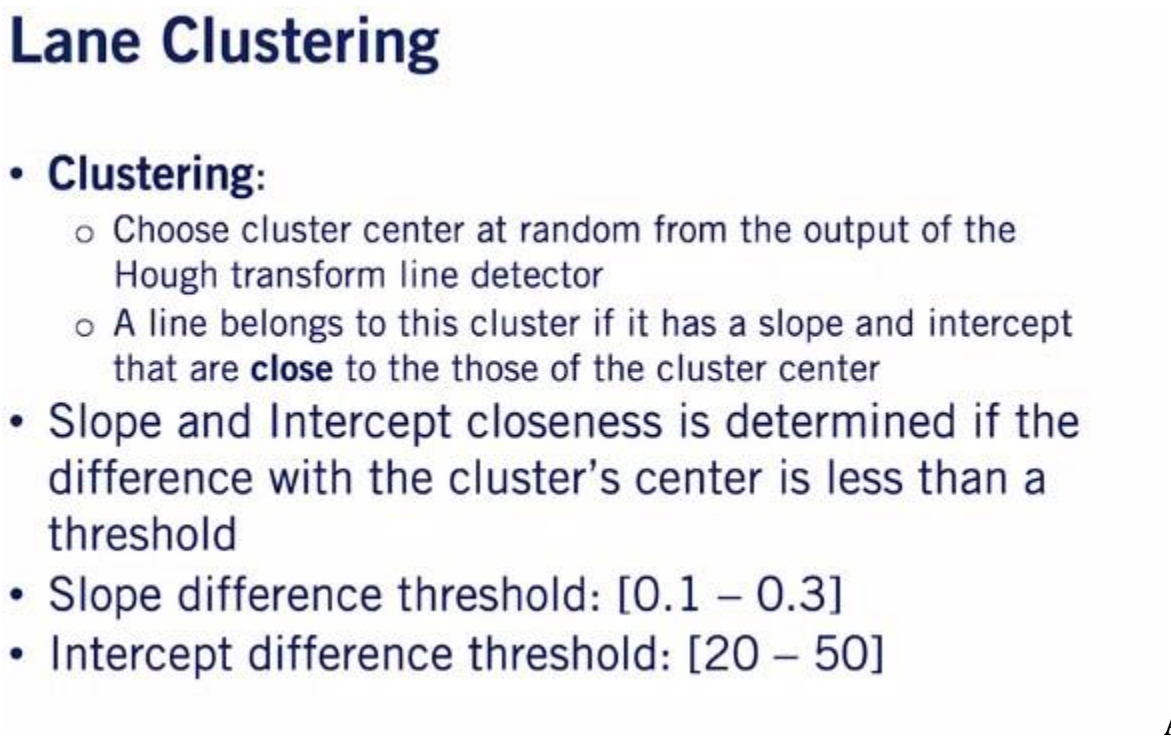
- **Filtering:**
  - Remove any line with slope in the range $[0, \eta]$
  - Try η in the range of 0.1- 0.3 for best results



To filter out the horizontal lines, we can rely on the slope of the estimated lines.

Horizontal lines and images tend to have a slope very close to zero. However, we also want to remove heavily slanted lines.

As such, a threshold is introduced as a lower limit of allowed slopes for the output of this filtering step. The exact value of this threshold needs to be determined empirically, try values between 0.1 and 0.3 for best results.



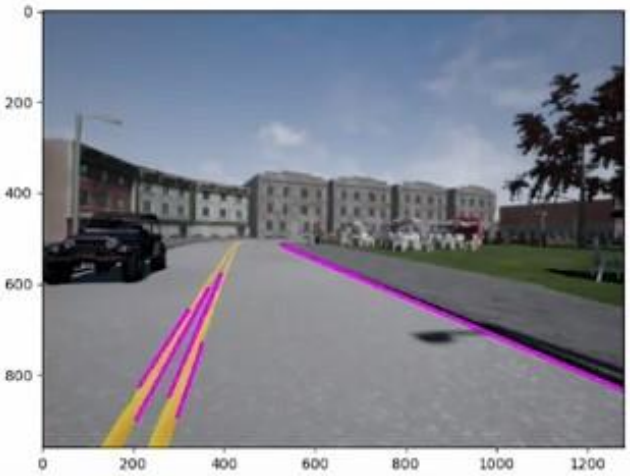
A simple clustering algorithm would be to first choose a cluster center at random from the remaining filter lines, then add to the cluster any lines that have a similar slope or intercept to the cluster line.

A line is considered close to the cluster center if the difference between its slope and slope of the cluster center is less than a specific slope threshold, and the distance between its intercept and the intercept of the cluster center is less than a specific intercept threshold as well.

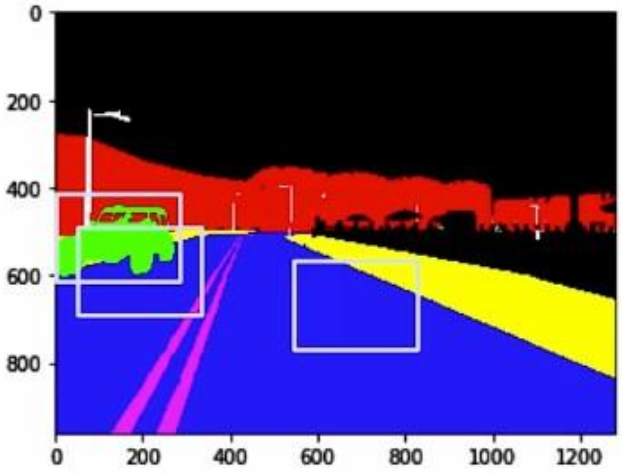
The slope difference threshold is usually chosen to be a maximum of 0.3, while the intercept difference threshold is defined in pixels, and is usually chosen between 20 and 50 pixels.

Lane Merging

- For each cluster, compute the average slope and intercept!



Filtering Uncertain Output Of Object Detectors



Filtering Uncertain Output Of Object Detectors

- Count number of pixels belonging to the car category from the output of semantic segmentation

The output of object detection is usually reliable. But for this assessment, we are given a high recall low precision detector that detects all objects in the scene, but also provides some false positives.

We are required to use the output from semantic segmentation to eliminate these false positives before estimating the distance to the obstacles. The results should be bounding boxes that reliably contain obstacles. To perform this filtering, we will need to use the semantic segmentation output to count the number of pixels in the bounding box that have the same category as the classification output from the 2D object detector.

The trick here, is that this number will depend on the size of the bounding box. We will need to normalize the pixel count by the area of the bounding box before attempting to filter out the detections with a threshold. The final normalized count is equivalent to computing the area inside the bounding box occupied by pixels belonging to the correct category.