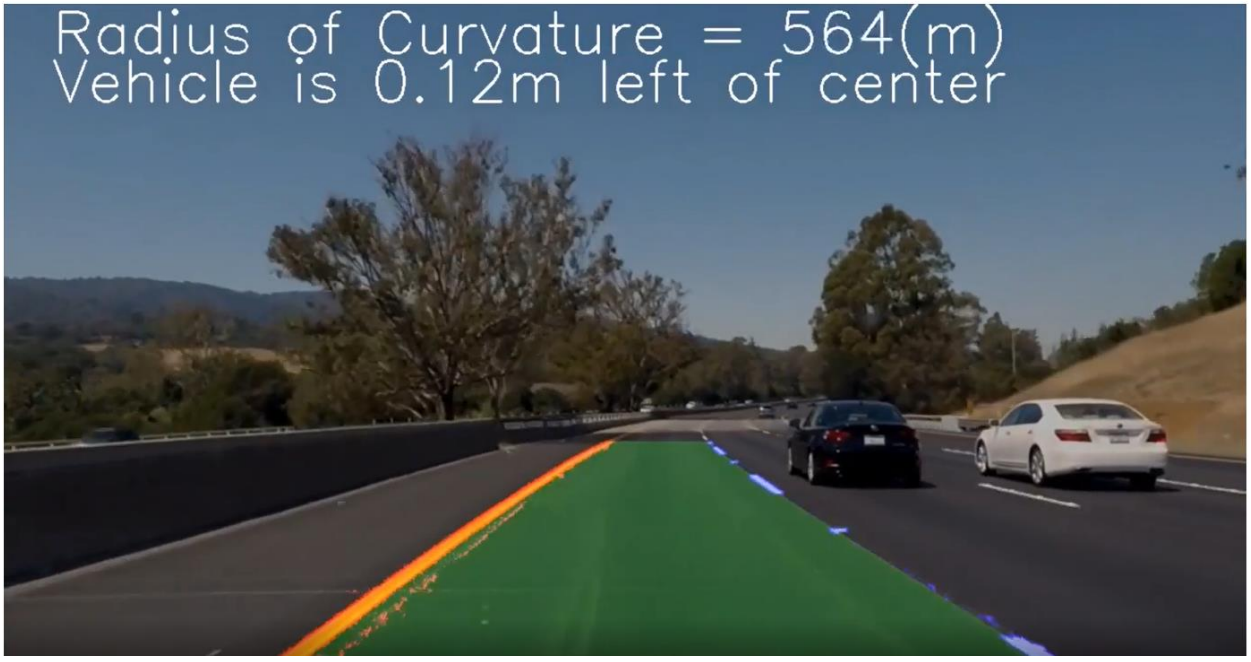
Simple Perception Stack for Self-Driving Cars

Self-driving cars have piqued human interest for centuries. Leonardo Da Vinci sketched out the plans for a hypothetical self-driving cart in the late 1400s, and mechanical autopilots for airplanes emerged in the 1930s. In the 1960s an autonomous vehicle was developed as a possible moon rover for the Apollo astronauts. A true self-driving car has remained elusive until recently. Technological advancements in global positioning (GPS), digital mapping, computing power, and sensor systems have finally made it a reality

In this project we are going to create a simple perception stack for self-driving cars (SDCs.) Although a typical perception stack for a self-driving car may contain different data sources from different sensors (ex.: cameras, lidar, radar, etc…), we're only going to be focusing on video streams from cameras for simplicity. We're mainly going to be analyzing the road ahead, detecting the lane lines, detecting other cars/agents on the road, and estimating some useful information that may help other SDCs stacks. The project is split into two phases. We'll be going into each of them in the following parts.

Phase 1 - Lane Line detection:

In this first phase, your goal is to write a software pipeline to identify the lane boundaries in a video from a front-facing camera on a car. You're required to find and track the lane lines and the position of the car from the center of the lane. As a bonus, you can track the radius of curvature of the road too. By all means, you are welcome and encouraged to use any technique that you see fit. You can assume the camera is mounted at the center of the car, such that the lane center is the midpoint at the bottom of the image between the two lines you've detected. The offset of the lane center from the center of the image (converted from pixels to meters) is your distance from the center of the lane.



Expected output:

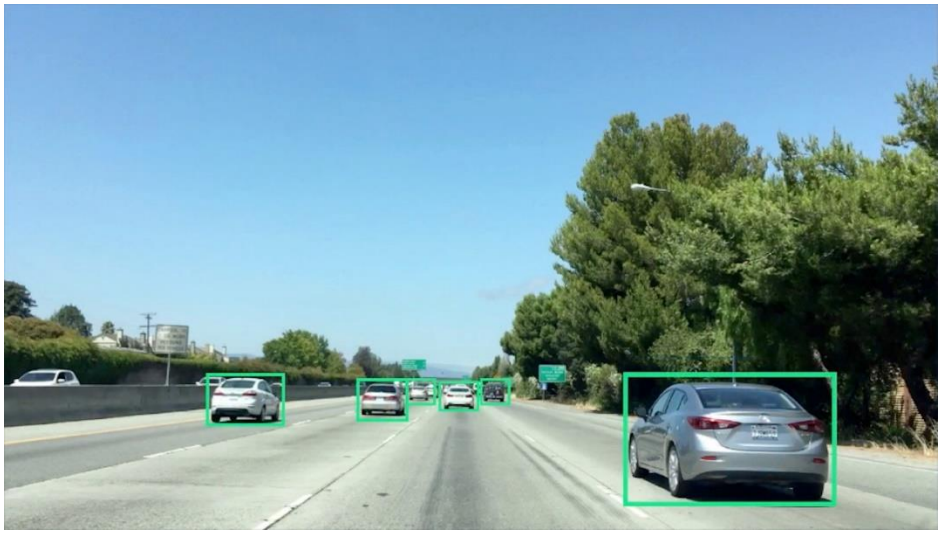
For the first phase, the expected output is as follows:

- Your pipeline should be able to detect the lanes, highlight them using a fixed color, and pain the area between them in any color you like (it's painted green in the image above.)
- You're required to be able to roughly estimate the vehicle position away from the center of the lane.
- As a bonus, you can try to estimate the radius of curvature of the road as well.

Figure 1 In this example, the pipeline consists of roughly 5 stages. the output of each stages is concatenated with the original image.

Stage 2:

In this stage, you're required to locate and identify the cars on the road.



Mainly, you'll be using HOG features from to search across the image. The dataset for this task will be provided right after the deadline of stage one.

What to be delivered:

- The code well written and commented.
- A Github repository with your code on it. The Github repository should be your main tool for different stages of the project (You shouldn't only push the code when you're done with each phase). The readme should also have clear instructions on how to run your code
- A jupyter notebook showing the result of your pipeline on the provided test images.
- The output of your pipeline on the provided test videos.
- Finally, your code should support a debugging mode whether it was a video or a single image. When this mode is activated, your pipeline should be showing all the stages that your code is going through.

Teams formation:

You're encouraged to work in teams. The maximum of each team is 3.

Deadline:

Phase one:

April 20th 2022 11:59 pm

Phase two:

May 15th 2022, 11:59 pm

The late policy is as follows: for each late hour of the first four hours, 20% of the total mark will be deducted. After the 4 first hours, only 20% of the total marks will be given without any further deductions. A late hour can be anything from one second to sixty minutes. No exceptions will be made.

Good luck