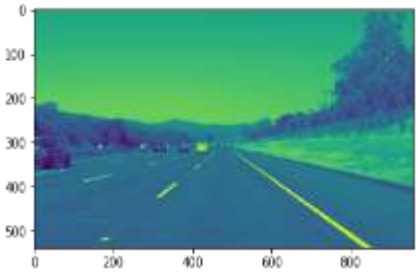
Finding Lane Lines on the Road:

This Report Explains the Pipeline implemented in this project to “Recognize and Highlight the Lane Lines” on Road.

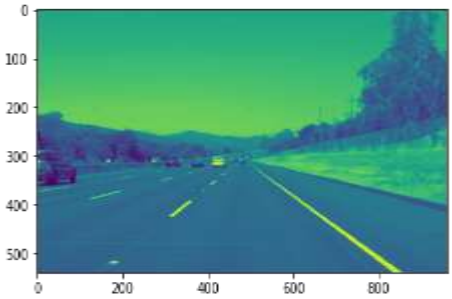
1. **Pipeline and draw_lines() function:**

   The pipeline for this project involves six steps. Below is the original image. Following the image are the six pipeline steps implemented on the image.

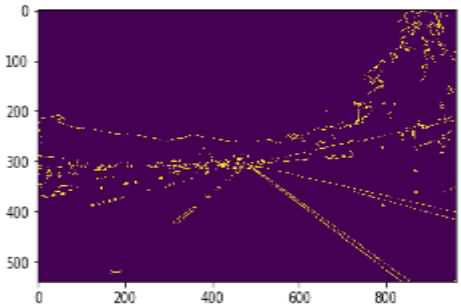
   i. The input image is converted to a Grayscale Image as shown below.



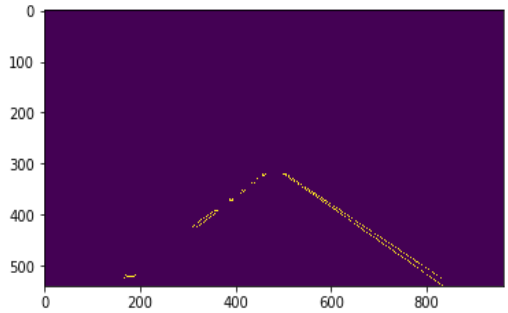
   ii. Gaussian Blur is applied on the image above to smoothen it.



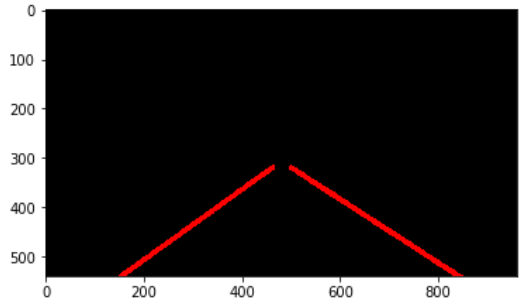
   iii. Canny transform with a lower threshold of 50 and the upper threshold of 150 is applied on the image from step 2.



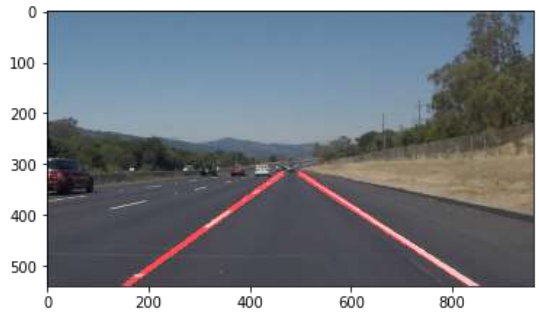
iv. Region of interest, a quadrilateral, was defined by its vertices and the above image was masked.



v. Hough Transform was applied on the above image, with parameters rho = 1 pixel, theta = pi/180 radians, threshold = 40, min_line_length = 10 pixels and max_line_gap = 10 pixels. The lines generated by the Hough Transformed image were given to draw_lines() function to mark continuous lines as shown below.



vi. The above highlighted lane markings are marked on the original image.



draw_lines() Function:

The lines generated by Hough Transform are not continuous in the region of interest. This function takes the image and the lines as parameters. It identifies the line segments that have positive (Line to the Right of Car) and negative (Line to the left of Car) slopes and groups them accordingly. An absolute threshold with slope 0.4 is applied while performing the grouping mentioned, this eliminates the near horizontal lines from appearing on the image. Using np.polyfit() function coefficients for the line equation y = mx + c are obtained from the points grouped for left and right lanes. These coefficients are used to find starting and ending points of the lanes (in region of interest) and the lines are marked in RED.

NOTE: All the test images were similarly processed and written to the folder "test_images_output".

NOTE: The above-mentioned pipeline was also applied on test videos. Processed videos were written to the folder “test_videos_output”.

2. **Potential Shortcomings:**
   i. The current pipeline is not able to identify lanes on an image taken in shade.
   ii. The region of interest had to be hardcoded.

3. **Potential Improvements:**
   i. The code pipeline must be improved so that lanes can be detected even on images taken in shade.
   ii. The current implementation has region of interest (a quadrilateral) hardcoded. This can be allowed to be chosen dynamically.