# Finding Lane Lines on the Road

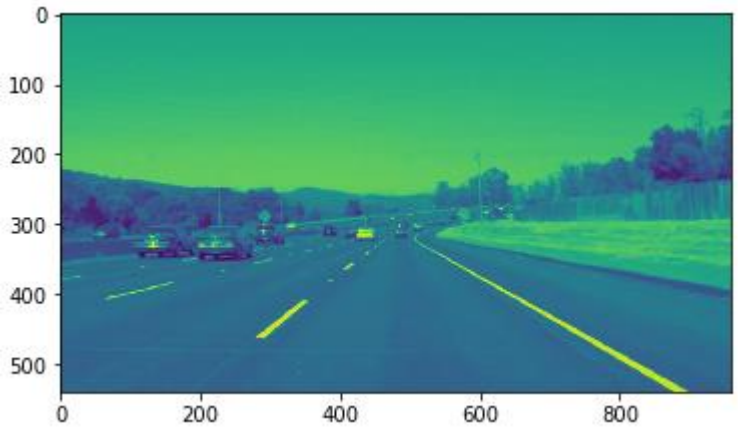
## Reflection

The proposed pipeline consists of 7 steps:

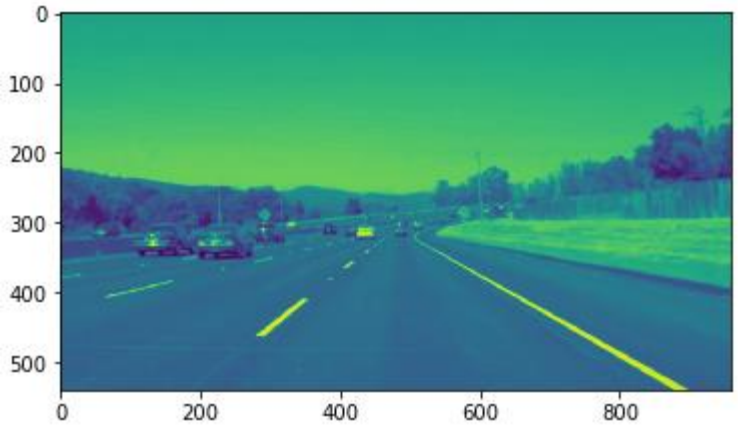
**The solidWhiteCurve.jpg image is used throughout this brief description of the pipeline to resemble each step.**

1. Took a copy of the image, to avoid affecting the input image.

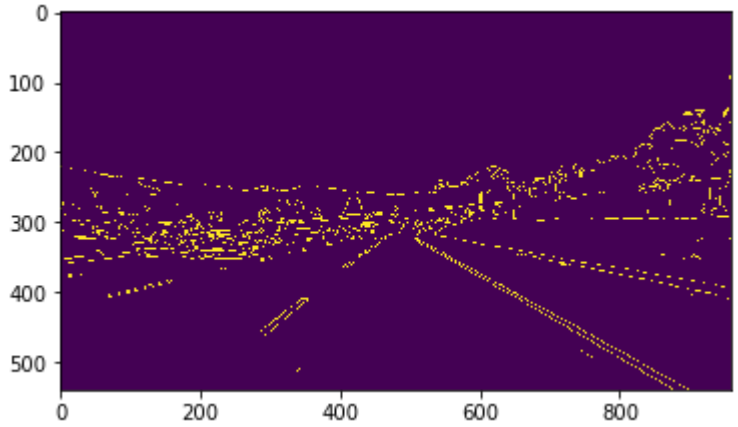

2. Applied the Grayscale transform by using helper function grayscale() on the image.



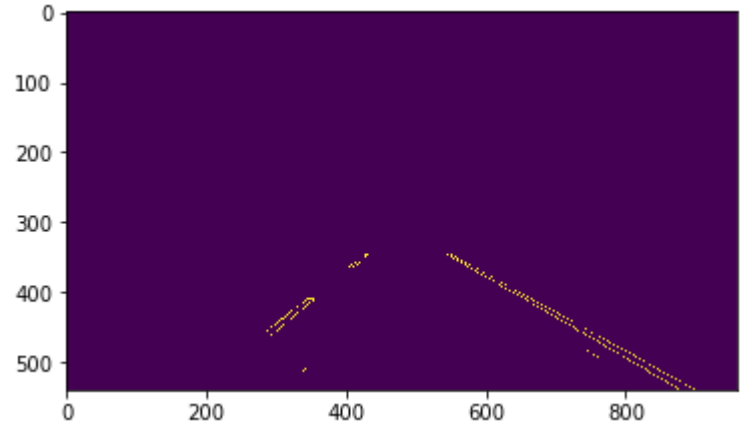

3. Smoothed the image using the Gaussian Blur function on the image using a Kernel Size of 3x3



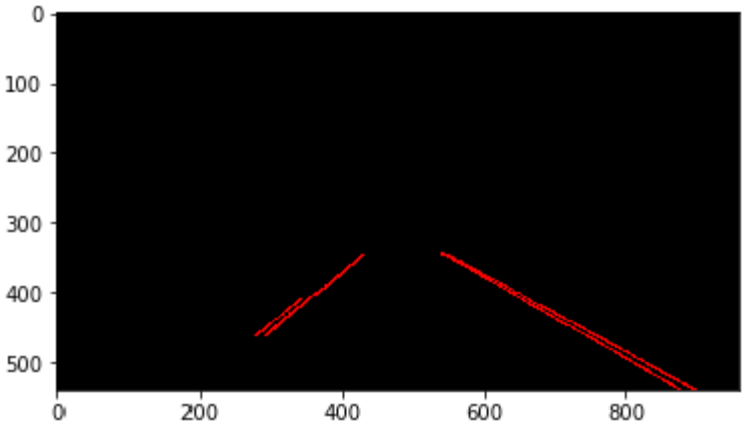

4. Detected the image's edges using the Canny Edge Detection function, with a ratio of 1:2 through a low threshold (=50) and a high threshold (=100).




5. Constructed a trapezoid mask for every image by determining 4 vertices across the image. Using approximate ratios with respect to the image's specific dimensions calculated by variables xsize and ysize.




6. Transformed the masked part of the image to a Hough Space, using the Hough Lines function, determining the lines connecting the Edge points previously collected through the Canny function.




7. Merged the lines drawn and input image together using the weighted image function, to display the drawn lanes marked by the red lines on the input image.


## Potential Shortcomings

The pipeline might have several shortcomings, which would cause indecisiveness in determining the lanes.

These shortcomings might occur under certain conditions: (a) Extremely high or low lighting. (b) Shadows, etc. All these conditions would might cause a distortion in the pipeline's algorithm since it detects the lines available in a specified region in front of the car, and it might be inaccurate.

## Possible Improvements

This algorithm needs a lot of improvement, one of the ways is using machine learning to know where the lanes are, instead of hard coding the ratios determining the region of interest. Moreover, we could use convolutional neural networks to detect more precisely the lane location.