Finding Lane Lines on the Road

Project Goals

1. Create a pipeline to detect lanes on an image of a road

The project involved first setting up the development environment that would have been conducive for the development of the first term projects and learning.

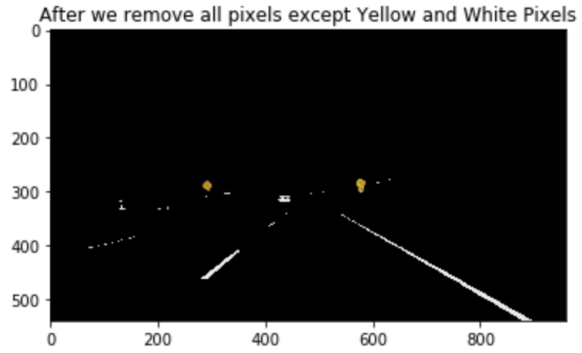
1. Given an image of a road detect the lane marks on the image

- To achieve this goal I created a number of variables that would be used by the helper functions they included kernel_size low and high threshold, parameters to be used by the Hough transform and the definitions of the point of interest.
- The Helper functions were already provided hence to achieve the first goal I only needed to define the pipeline to be used.

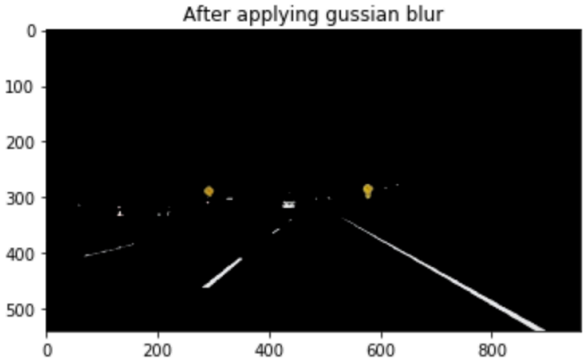
The function takes an image and takes moves it through a series of steps each geared towards being able to detect the lane marks.

Pipeline stages

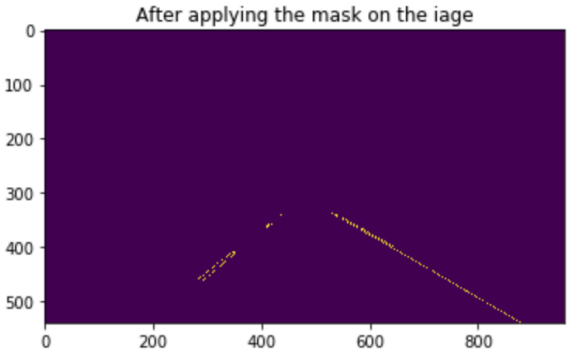
1. Color detection function (color_det_image())



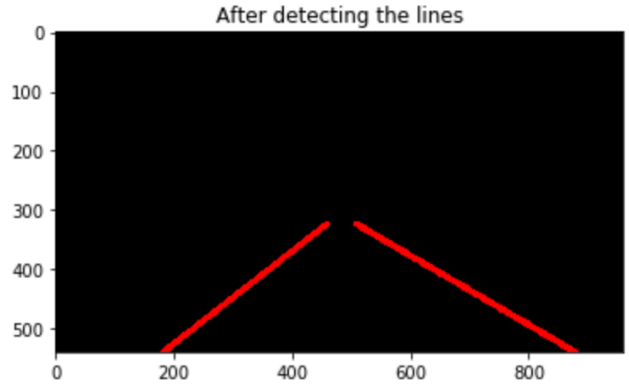
2. Apply gussian blur



3. Apply the mask to get out area of interest



4. Draw the lanes using Hough transform



5. Bringing it all together

The heart of the transformation draw_function

The challenge with just working with the a few images was that they did not show all the short comings that the initial draw function had. To make amends I did the following

1. Determine the gradient on every two points to be drawn (Y=mx+b) = (y2-y1) / (x2-x1)
2. Only keep the the slopes that is above our threshold
3. Run a outlier function to elimate those slopes that are way too off from the rest
4. Collect the lanes both left and right
5. Determine the points to be drawn
6. Only draw the lines if they are either left or right

Using the Pipeline for the lane detection find lanes from a video

For this challenge the program uses video frames that are passed to a function process_image() which takes every frame of the image and passes it to the pipeline as an image then using the defined algorithm shown above.

Potential Shortcomings

1. The pipeline works well on still images but the right lanes are abit too noisy and will need to fine tune the way the lanes are collected and slope to be fine tuned to ensure that the noise is not that much.

Improvements to pipeline

1. The code written would not be very good as far as time and space complexies are concerned and hence would be better if we reduce the number of for loops and the number of variables used to reduce time and space requirements.
2. The smoothening of the lanes for a better experience.

References

1. http://www.math.com/school/subject2/lessons/S2U4L2DP.html
2. https://pythonprogramming.net/color-filter-python-opencv-tutorial/
3. http://stackoverflow.com/questions/28789614/x-and-y-intercepts-from-slopes-python
4. http://stackoverflow.com/questions/19406049/extrapolating-data-with-numpy-python
5. http://stackoverflow.com/questions/22354094/pythonic-way-of-detecting-outliers-in-one-dimensional-observation-data