Finding Lane lines on Road

Reflections:

The project aims at identifying the lane lines in an image and then extend the algorithm to a video to identify the left and right lanes effectively irrespective of their shape (solid vs dashed) or colour (white vs yellow) or slope (straight or angled or curved). The image/ video used in the project is from a highway where the lanes are clearly marked and they are mostly straight and not much curved.

Following features have been used for the identification:

1- **Colour**: Lane lines are generally light coloured (white/ yellow) compared to the road (dark grey). So a black/white image works better as the lanes can easily be distinguished from the background.

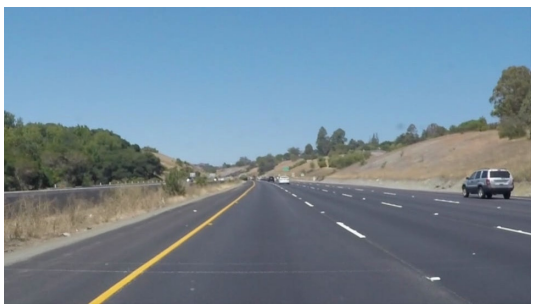
2- **Shape**: Lane lines are usually solid or dashed lines which can be used to separate them from other objects in the image. Edge Detection algorithms such as Canny can be used the find all the edges/ lines in the image. Then we can use further information to decide which edges can be qualified as the lane lines.

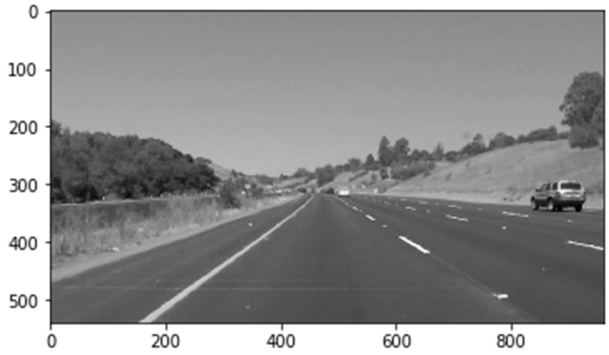
3- **Orientation**: Highway Lane lines are more closed to vertical than they are to horizontal. So the slope of the lines detected in an image can be used to check whether it is a possible candidate for the lane or not.

4- **Position in Image**: In a regular highway image taken by a dash cam mounted looking ahead on the car, Lane lines typically appears in the lower half of the image. So the search area can be narrowed down to the region of interest to reduce any noise.

The pipeline does the following steps to mark the lane lines:

1- **Converting the image to grayscale**: Image is converted to grayscale by using OpenCV function cv2.cvtColor (). This made it easier to identify lanes and also helped with the next step of edge detection. Gausssian Blur is applied to average out the cell values in order to reduce noise before taking gradient.





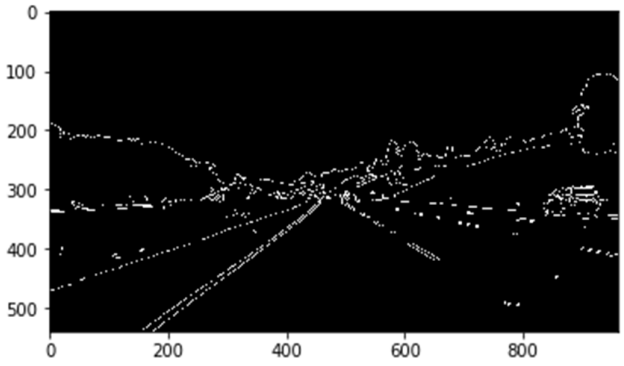
2- **Canny Edge Detection:** Next the gradient of the image is taken to compute the boundaries or edges of the objects in an image.

cv2.Canny(blur_gray, low_threshold, high_threshold) is the function used to perform canny edge detection where low and high thresholds are the thresholds for how strong an edge should be in order to be detected.

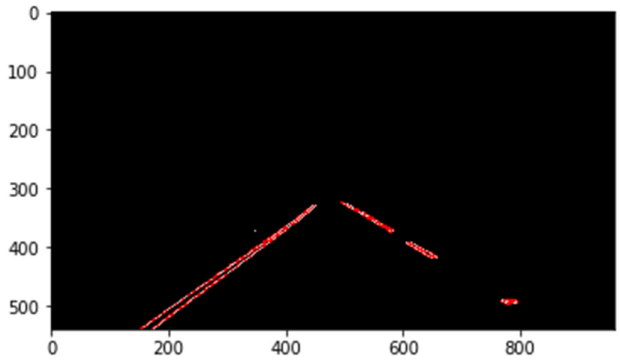
We have used the values as

low_threshold = 50

high_threshold = 150



3- **Region of interest**: Next a four sided polygon is defined to mask the region of interest. As mentioned before, the parts of the image where lane lines can be found are identified which in our case was a polygon in lower half of image and the other regions of the image are masked to limit our search within that area.



4- **Hough Transform:** Once the edge pixels are detected in an image, hough transform is applied to connect these edge pixels to form lane lines.

cv2.HoughLinesP is the function used which takes input image from Canny and a few other parameters in order to define the lane lines.

The output from Hough Transform contains all the lane lines (solid/dashed). Next step is to find a common average line to smoothen them

5- **Finding average equation of line and smoothing:** Once the lines are identified in the image, they are passed on to draw_lines(line_img, lines) function which finds the average equation of the line.

It divides the left and right lane line by calculating slope. Negative values correspond to right positive to the left (based on image coordinates).

Once you have separate left and right lines, the slopes and intercepts for each left/right lines are averaged out by accumulating all the points in each left/ right and finding an average fit through them.

6- **Extrapolation:** Once you have average left /right line, they are extrapolated to generate a common solid average lane line.

Once the pipeline starts working on the test images. The same is done for each frame of the video to generate the lane lines for the whole video.

All the output images are in output_images folder and the final video is also attached

Shortcomings

- Doesn't work very well on the challenge video as the lines are more curved and we are fitting lines in the equation
- As we are only operating on grayscale images, colour scheme is sometimes confusing for the pipeline particularly at the frames in the video where dashed lines are not clear or where other lines are present.

Possible Improvements:

- Use of second degree polynomial in order to account for the curved lanes
- Use of different lane identification techniques in order to be more accurate.