Writeup report on Finding Lane Lines on the Road

Description of Pipeline

These are the steps in the pipeline that I used.
Step 1: Make image Grayscale
Step 2: Apply Gaussian blur on the image
Step 3: Detect edges using Canny Edge Detection Algorithm
Step 4: Apply your choice of region of interest. I chose trapezium with vertices being

    (imshape[1]*1/10,imshape[0]),
    (imshape[1]*4/10, imshape[0]*6/10),
    (imshape[1]*6/10, imshape[0]*6/10),
    (imshape[1]*9/10,imshape[0])

where imshape = image.shape

Step 5: Apply Hough Transform algorithm to detect lines in the image
Step 6: Filter lines based on slope. Lines with abs(slope) between 0.4 and 0.8 are chosen
Step 7: Get lines that represent lanes and span the region of interest

One strategy that I used here was to create segments that would join broken segments

- a.) Take edges that were say of positive slope (right lane)
- b.) Create a smaller list of edges from above edges that do not have multiple edges for x point.
  That is: a list of edge [(0,0,100,100),(5,5,10,10)] would become [(0,0,100,100)]
- c.) Use above to create segments that connect edges by interpolation
- d.) Use extrapolation to create segment for corners (max_y and min_y cases)
- e.) This strategy did not yield good results. So, it was ditched

Another strategy was to use the segments (say positive slope ones) and come up with a line that would best fit those points. The choice of model to create a line segment was Linear Regression

- a.) Take edges that were say of positive slope
- b.) Create arrays that would contain X and Y points
- c.) Run linear regression on X and Y to get the line of best fit
- d.) Use slope and intercept from regression to calculate X values for max_y and min_y values
- e.) max_y = 540, min_y = 324 (imshape[0] and imshape[0]*6/10 respectively)
- f.) Return these lines

Step 8: Use lines detected after hough transform and the two lines from Step 7 and create an image
Step 9: Combine original image and image from Step 8 using cv2 addweighted method
Step 10: Plot the final image

Potential shortcomings and possible solutions

a.) I think using just a single line to represent a lane line is a shortcoming. The reason is when a car would be making turns, the lane would be some what circular in shape. This curve can be represented by a set of smaller segments or perhaps a parabola.
The algorithm can run regression of order one or two and choose the model with the least amount of error. So, for a straight road, the algo would detect a straight line and for a curve, a parabola would be detected and plotted.

b.) Another shortcoming to the existing pipeline is failure to detect lines when the car enters an areas on the road that is under shade. (This is evident when I run my pipeline on challenge.mp4)
This shortcoming can be improved by tuning edge detection and hough transform algorithms