Finding Lane Lines on the Road

The goals / steps of this project are the following:

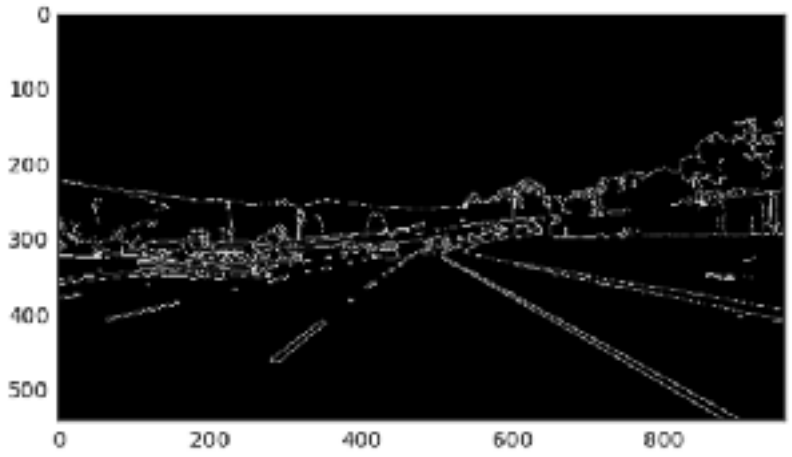
Make a pipeline that finds lane lines on the road

Reflection

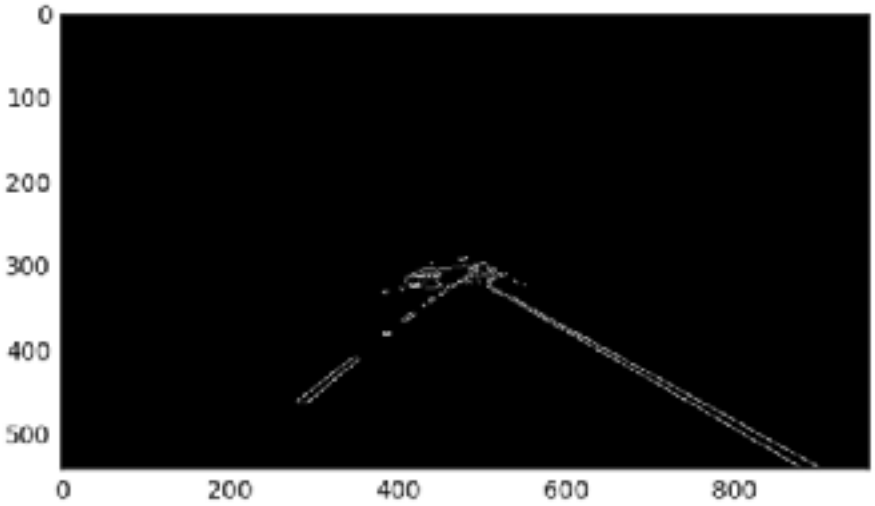
1. Describe your pipeline. As part of the description, explain how you modified the draw_lines() function.

My pipeline consisted of 5 steps. First, I converted the images to grayscale, then I ....

1) Grey scale conversion, Gaussian Blur w/Kernel Size of 9 vs. 3. Canny for edge detetion with 50,150. This results in a black background with white lines.



2) Apply ROI to canny with trapezoid shape. This is a hard coded shape and eliminates the lines which are not lane lines. This region of interest is a trapezoid:
[[(0,imshape[0]),(450,290),(490,290),(imshape[1],imshape[0])]], the image below is the same as the one above with the ROI applied.



3) calculate the lines in the above image using the proabilistic hough transform. Observe the left and right slopes. Filter into 2 lists and caculate average. Take longest segment and interpolate to where y=0 and calculate vanishing point.

Algorithm for calculating vanishing point:

y=mx+b for both left and right lines
bleft = yleft -mleft*xleft. Use the tuplepoints for x,y
bright = yright -mright*xright. Use the tuplepoints for x,y

equations foe lrft/right lane lines

yleft=mleft*xleft+bleft
yright=mright*xright+bright

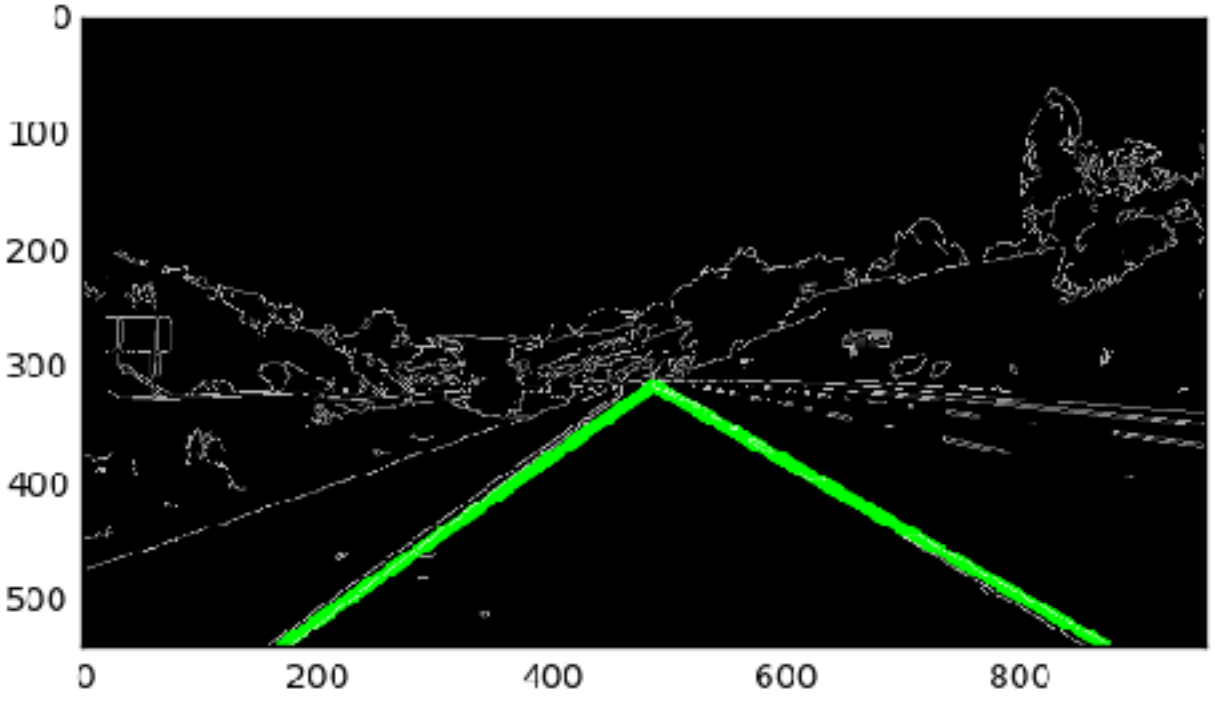
bleft = yleft - mleft*xleft
bright = yright - mright*xright

the vanishing point is when yleft=yright=yvp and yright=xright=xvp

y_vp=mleft*x_vp+bleft
y_vp=mright*x_vp+bright

mleft*x_vp+bleft = mright*x_vp+bright
mleft*x_vp - mright*x_vp = bright - bleft
x_vp(mleft - mright) = bright - bleft

x_vp = (bright - bleft)/(mleft - mright)
y_vp = mright* x_vp + bright



The vanishing point represents the point on the horizon where anything above that should be filtered out.

4) Draw the lane lines using the interpolated/calculated vanishing point using 60% of the value from y=0 to the vanishing point. Adjust to smaller value as the car starts moving. Average lines together to prevent sudden jumps.

2. Identify potential shortcomings with your current pipeline

1) the ROI is a static dimension. One shortcoming would be if a car was in front of the vehicle before the vanishing point. This would create more vertical and horizontal line artifacts. This problem would get worse with more cars. Also for situations like lane changes or u turns this method would not work well.
2) These images are all daytime images. Another shortcoming are lines created from shadows or no edges in low light or if there is rain.

3. Suggest possible improvements to your pipeline

1) Improvement: add a dynamic ROI region. Compensate for traffic or cars to the side and in front of the camera.