1. Pipeline description:

My pipeline consisted of 5 steps.

First, I converted the images to grayscale, then I used Gaussian blur to blur the image. After Gaussian blur, I apply canny function to the image to display only the edges where color changes drastically.

Gaussian blur and canny parameters are shown below:

- Kernel size = 5
- Low_threshold = 50
- High_threshold = 150

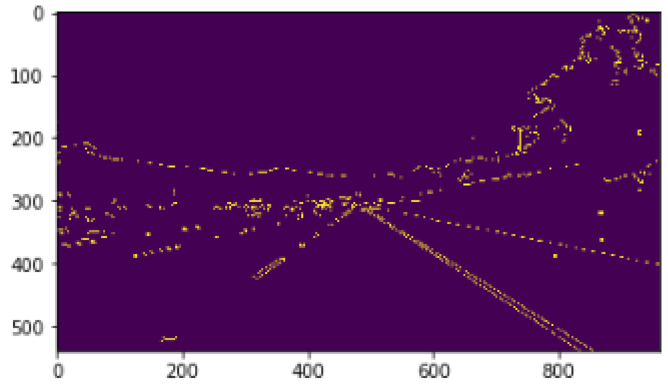

Fig 1. Image after canny function

Then, I define a four sides polygon to mask the area I am not interested.

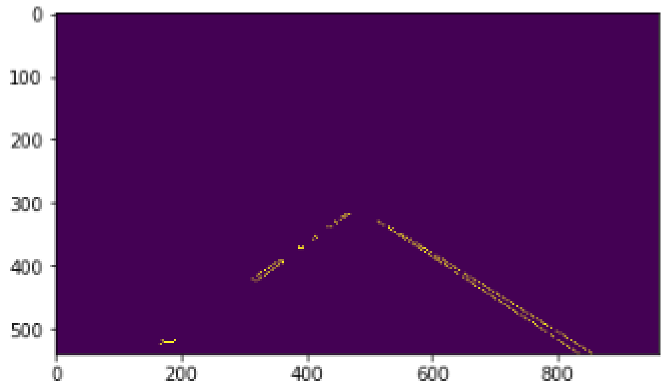

Fig 2. Masked image

After masking the image, I applied Hough Transform algorithm with the edge detection image from canny function as my input. Following is the parameters I used:

- rho = 1

- theta = np.pi/180
- threshold = 15
- min_line_len = 40
- max_line_gap = 20

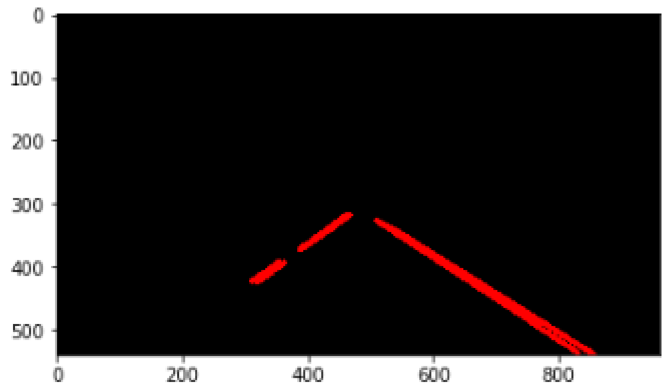

Fig 3. Hough lines

Finally, I overlay the above image with the original image:

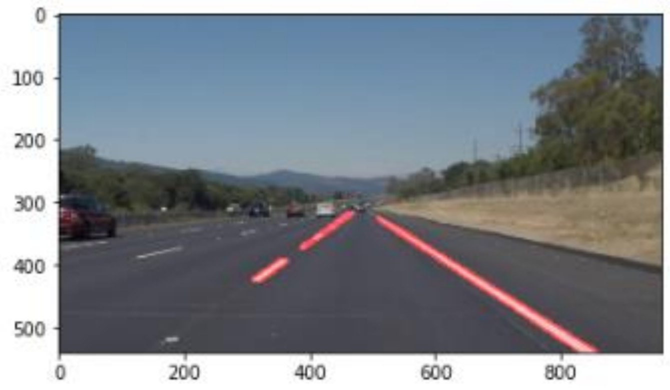

Fig 4. Edges detection

In order to draw a single line on the left and right lanes, I modified the draw_lines() function by taking the average slope and interception points of the lines detected by Hough transformation. Then use the average interception points to draw two single solid lines.

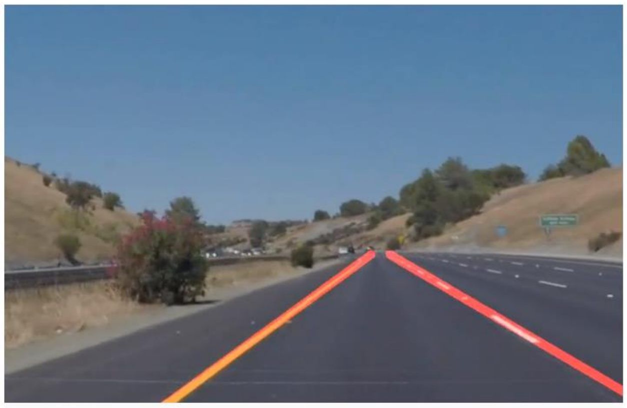

Fig 5. Edge detection (Solid lines)

2. Shortcomings with current pipeline:

1. One of the the shortcoming of the code would happen when the lines on the road are not straight. The current hough_lines function can only detect the lines that are straight.
2. The other shortcoming of the code drawing single line on both side would be left and right lines are not always accurately captured in every frame of the video.

3. Suggestion to improve the pipeline:

1. One of the suggestions would be to have a hough line function that can detect polynomial lines and use the polynomial parameters to print out the lines.
2. For drawing the single line, one suggestion would be to eliminate the slope if the slope value is far away from previous detected slope value.