# Finding Lane Lines on the Road

Nicolas Keck

## 1. Pipeline Description

For this assignment I created two pipelines, one simpler pipeline to be used with the solidWhiteRight and solidYellowLeft videos, and a more complex pipeline to be used for the challenge video. My complex pipeline consisted of 6 steps. Firstly, a frame from the video is read and split into its BGR channels. These channels are then re-merged with the exception of the green channel such that the lane line in the challenge video is more easily detected. Secondly, the image is blurred with a gaussian blur effect in order to enhance lane line detection. Third, canny edge detection is run with predefined lower and upper boundaries to filter out extra edge noise. Fourth, a predefined polygon mask is applied to the image to further reduce noise and ensure that lane lines are only "found" upon the road. Fifth, a hough transform is applied to the image to draw lines on the outlines of the lane lines. Finally, the lines from the hough transform are processed and each line that is found to be too horizontal is discarded. The remaining lines are then used to find single lane lines for the right and left side of the road. These lines are drawn onto the frame and the finished frame is written to the output video.

In order to draw a single line onto the left and right lines, I did not modify the draw_lines() function. Rather, I decided to in-line any of the useful helper function code and write all my code without the use of user-defined functions instead. My method for creating single lane lines for the left and right side of the roads revolved around using linear regression in order to determine a line of best fit for each side of the road. For this method to work I first had to reduce noise from any outlier lines, and I did so by removing lines with slopes between -0.25 and 0.25. Then, I used the x values of the points making up each line to determine if the points were on the left or the right side of the image. I then performed a linear regression on each group of points to come up with an equation for a left and right lane line, and I used those equations to draw lane lines from the bottom to the top of my polygon mask. I could have also averaged the slopes of each line making up a lane as well, but I felt that my linear regression approach worked well enough.

For my code for my simple pipeline see: **videoAnnotator_final.py**

For my for my complex pipeline see: **videoAnnotator_color_final.py**

## 2. Issues with Current Pipeline

There are a couple of issues with the current pipeline, the most prominent being that the pipe has a lot of difficulties dealing with curved lines, and the fact that the polygon mask is currently not dynamic. This leads to a number of visual issues which I find are most present in the annotated challenge video. As can be seen, the fact that the system can't handle curves demonstrates itself

in the visual effect that the lane lines are constantly shifting back and forth attempting to match the lane lines. Additionally, the lack of a dynamic polygon mask demonstrates itself when the car starts to drive into the curve, as the system detects part of the wall to be a lane line and the left lane line veers off the road for a second. This can also be seen when a car occasionally drives too close to the left lane and the line freaks out a bit too.

Some issues I found with the pipeline when trying some of the optional videos was that the pipeline handled motion blur and shadows very very poorly. As can be seen in **hough_1.avi**, the hough transform hardly picks up anything in the video, and it barely even manages to detect the lane line. Meanwhile, **hough_2.avi** demonstrates the issues that shadows play with the camera, as each shadow is seen as a bunch of edges through canny edge detection. Blur again also plays a big role in the second video, as the camera movement is too rapid and jerky for the algorithm to detect anything. Because the detection on these videos was so poor, I couldn't even properly annotate them, and therefore left them only partially-annotated with hough lines.

For my annotated videos see: **annotated_videos**

For my hough transform videos see: **hough_videos**

## 3. Possible Improvements to Pipeline

Some possible improvements to the pipeline all stem from the current issues with it. First and foremost, adding a method to draw curved lane lines would improve the pipeline significantly, as the current results are marred a bit by the jerky way in which the straight line attempts to handle a curved line. Additionally, applying some of cv2's anti-blur features, while difficult, would also make the pipeline significantly improved. This would allow the algorithm to see the road even when the camera is bumping around, and would therefore allow me to attempt examples 1 and 2 again. Finally, adding some sort of dynamic polygon which reacts to upcoming curves and cars would also greatly improve the pipeline, as the lane lines would no longer be confused by other features.