**Reflection**

**1. Describe your pipeline. As part of the description, explain how you modified the draw\_lines() function.**

My pipeline consisted of 6 steps. First, I converted the images to grayscale where RGB are converted into grayscale images. 2nd stage is applying the smoothening the image to supress the noise before applying the canny edge detection algorithm. 3rd stage is applying the canny algorithm which detects edges and spits outs edges in the image. 4th stage is the selection of region of interest where a quadrilateral was selected with appropriate vertices. 5th stage was the application of hough transform to detect lines from edges in the region of interest. 6th stage was a post process step where the detected lines were overlapped with initial image using appropriate weights.

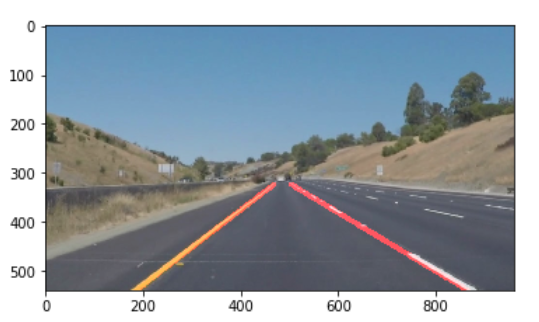
Inorder to draw a single line on the left and right lanes, I modified the draw\_lines() function by doing the following things:

1. Identifying the lanes in thee basis of their slops. Note: left lane has negative slope (image coordinate system is centred at left top corner instead of left bottom corner.) The left and right lanes are identified using this concept. (Slope>0 – right lane, else left lane)
2. Slopes values for each of the points selection for both left and right lanes are stored in lists. Finally, an average of these slope values is selected as the slope of the lane line. The respective coordinate are also stored.
3. Now we have the slope and the coordinates, from these values the intercept is also determined. This gives an equation of a straight line (Y = Slope\*X+Intercept). By basic understanding, the lane starts from the bottom most point of the image recorded by the car. Therefore, choosing the bottom most point of the image and feeding it to the equation of the straight line to get X. Finally extrapolating from the extreme points viz. bottom most point and top most point. This gives us the required line.
4. The odd values of slopes are thrown out because logically thinking lanes will mostly be in a range of slope of 30 deg to 60 degrees unless there’s a really steep curve (highly unlikely on roads).

Image: Before



Image: After



### 2. Identify potential shortcomings with your current pipeline

One potential shortcoming would be what would happen when there is a steep curve, since this is a straight line interpolation, the lane line detected will not be a curve line but a straight line.

Another shortcoming could be where the region selected by the algorithm is sometimes very large such that the surrounding areas are also included in the lanes detected (wrong detection) like in the optional challenge.

Another shortcoming could be when there is uphill/downhill road, since the region of interest doesn’t take this change in height into account, the lanes detection wouldn’t work.

### 3. Suggest possible improvements to your pipeline

A possible improvement would be to reduce the lanes detection to near lines only not very far. This way we can restrict the region of interest closer to the car and a better and accurate detection.

Another potential improvement could be to use non-linear interpolation technique intend of a straight-line interpolation method.

I tried the optional challenge with color selection technique as well, but was unable to integrate color selection code into the pipeline. This could also be another improvement in the algorithm.