1) Converting image to grayscale:

2) Applying Canny edge detector:

Instead of using the regular canny edge detector and hardcoding the thresholds, I instead used a function called autocanny() which automatically calculates the threshold to give the best edges.

3) Gaussian Blurring:

Gaussian blurring is done to reduce the noises and focus more on the important pixels.

4) Region of Interest:

I have used a triangular polygon as the Region of Interest.

5) Drawing the Hough lines:

Hough Lines are drawn to the edges obtained within the area of Region of Interest.

6) Drawing lines over the lane:

1. Initially I found the segments whose slopes are between -0.6 and -0.9 to be averaged and then found the average left lane line slope.
2. Then, I found the segments whose slopes are between 0.45 and 0.75 to be averaged and found the average right lane line slope.

3. Average the left line segments points to get an average point that left lane line will go through, and the same method is applied to right line segments.
4. After getting the average point and slope, I then calculated the 2 points position along the line on bottom and half height of the image. Same method is applied to both the left lane line and right lane line.
5. Finally, I connected the 2 points to extrapolate the line segments.

7) Potential Shortcomings:

The lines were not efficiently drawn when the slope values I meant went out of range. Also the model didn't perform well in cases where the lane lines had slight shadows.

8) Possible improvements to pipeline:

Instead of just dealing with the RGB color space, I think considering other color spaces like HSV, HLS and other aspects like gradients could be of a great help during lane identification.