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
- Apply this pipe line to detect the lines in a video form the road

1. Describe your pipeline:

The pipeline of consist of following steps:

- Loading the image of the road, as follow:

Figure 1. Sample loaded image.

- Apply Adaptive Histogram Equalization on the loaded image to histogram equalization on 8x8 squares to adjust the contrast in the image. It is useful for the challenge video when there is the tree shadow on the road. By applying the Adaptive Histogram Equalization it is still possible to detect and follow the line in this situation.
- Using inRange form OpenCV for the color selection. By using the selected threshold value of [205, 205, 0] it is possible to detect the yellow lines too, which that is helpful for the video with yellow line. It is also possible to convert yellow pixels to white and use threshold values to just detect the white pixels. The output of cv2.inRange is a gray image then no need to change it to gray.
- Use Gaussian filter with kernel size of 5 to smooth the previous step output.
- Use the Canny algorithm with low and high threshold of 50 and 150 to detect the edges in the image.
- Pick the region of interest by choosing the size of polygon. It is possible in the definition of the vertices use the image dimension divided by a value, which is useful to select the region of interest for images and videos with sizes different from 540 x 960.
- Use Hough transformation to detect lines. The Hough transformation parameters are as follow: rho = 2, theta = 1 radian, threshold = 15, min_line_len = 15, max_line_gap = 35. The hough_lines function uses the draw_lines to add the detected lines in the image.

- Shows the image containing lines on top of original image.

2. Make continues left and right lines:

Using the draw_lines adds all detected lines to the image and it can highlight whatever line on the road. To create a single continues left and right line from all detected lines (figure 2) I defined the draw_lines2 as follow:

- Separates the left and right lines based on the values of x2, if x2 be bigger than image image.shape[1]/2 then it is a right side line, otherwise it is a left side line. And it store the right and left detected line in different lists.
- Keeps the lines that have slopes in certain range to avoid some horizontal lines in the road.
- There can be several detected lines in left and right, it averages on each of them to get one line for each side.
- Based on the coordinates of the line on each side it can do an interpolation for that side. It uses two points to interpolate: one y=540 or bottom of the image, other y = 325 the top side of polygon. Here instead of y = 325 I used image.shape[0]/1.66 which is useful for images have different size as the challenge video. And then it appends the interpolated lines to the line repositories, which is defined outside of the function to store the detected lines from different video frames.
- Always keep the last 15 interpolated lines in the repo and remove the rest to prevent memory overload for long videos, and average over these interpolated lines in the last 15 frames to make line movement on the video smoother.

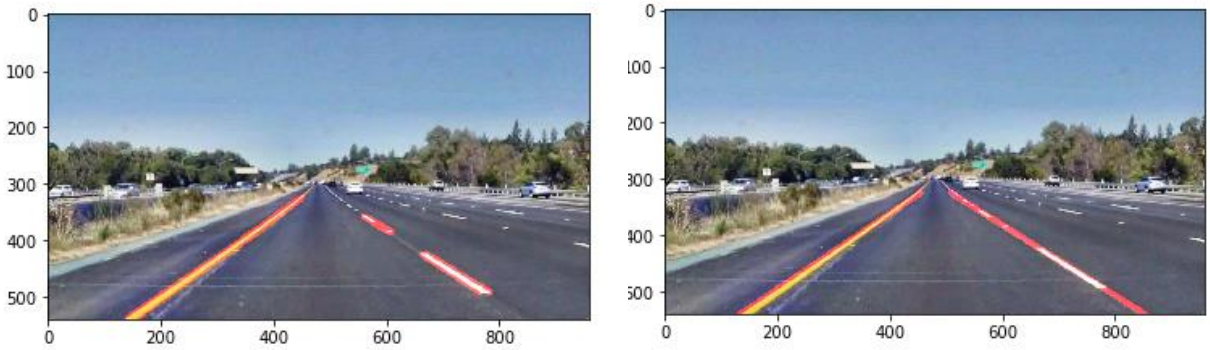

Figure 2. Left image: added all detected lines, Right image: added averaged interpolated lines

Summary:

To implement the algorithm I have used the Adaptive Histogram Equalization, which is very effective in adjusting the contrast of image and be useful to detect line in the areas of the image have different contrast (figure 3). Also using inRange with lower threshold of [205, 205, 0] facilitates detecting the yellow lines in the image. I he defined draw_lines2, which can separate left and right lines, interpolate

to find a single line for each side and, average them on the previous 15 frames of the video to make smoother movement of the detected line in the video.

Figure 3. Detected line without (left image), with (right image) adding Adaptive Histogram Equalization in the pipeline image: added averaged interpolated lines.