Finding Lane Lines on the Road

First edited on Oct/14/2018 by Bo Li

In this project, the task is completed by the helper functions the udacity has applied beforehand, as well as writing some more helper functions in details. The lane finding pipeline seems to be a nice idea to improve efficiency by combinating those helper functions I have mentioned.

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
- Reflect on your work in a written report

Reflection

1. Describe your pipeline. As part of the description, explain how you modified the draw_lines() function.

My pipeline consisted of 7 steps.

First, I converted the images to grayscale, then I apply Gaussian smoothing, and detect the edges by canny function by setting the the thresholds as the third step.

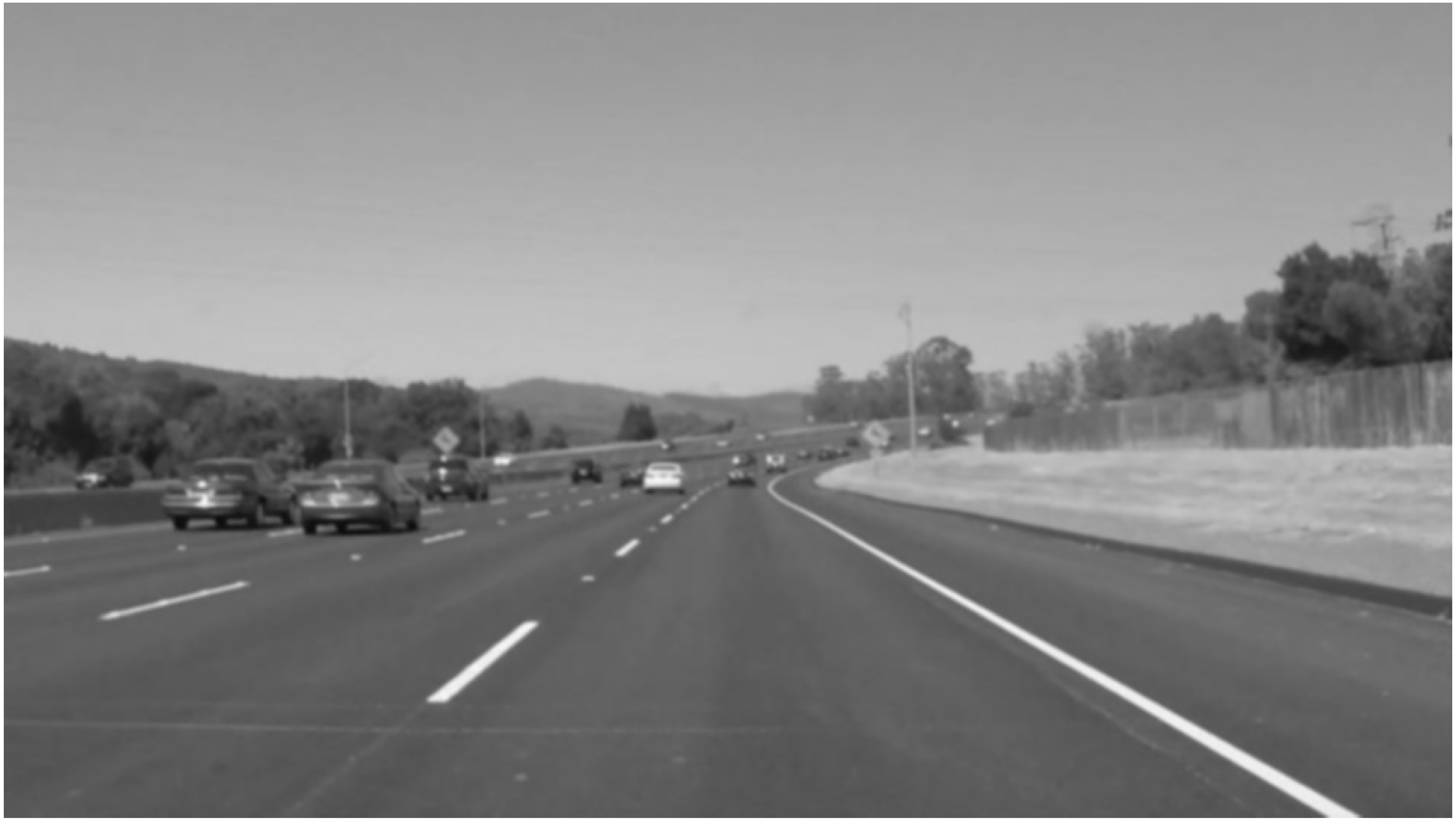
For the fourth procedure, we make use of the get_quad_vertices helper function to get vertices for our mask, so that the lane will be approximately fall in the range of the quadratic region no matter what image appears. The fifth step is a little tricky, I make use of the ordinary least square method to compute the angles for the left and right lanes. The sixth and the last steps are for setting thickness of the line and transparency of plots.

In order to draw a single line on the left and right lanes, I integrate the draw_lines() function into my own helper function draw_lane_line() in order to draw lines on the empty image. And we combine the line and the orginal images together by using the cv2.addWeighted() helper function to show the overall presentation.

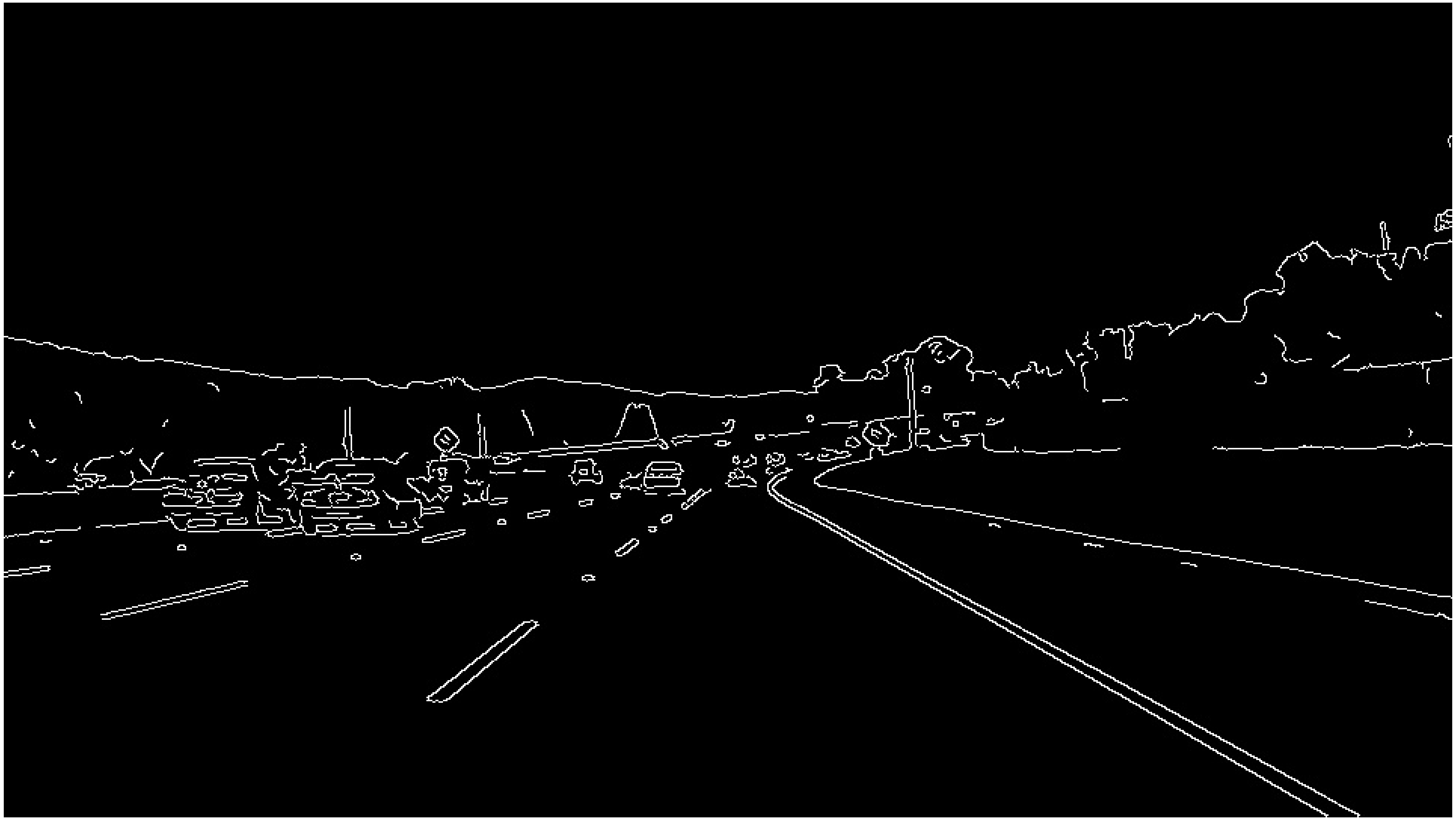
Now I would like to give the examples of its application on the test images in the test_images folder.

The fist step is to convert the image to greyscale image as the following

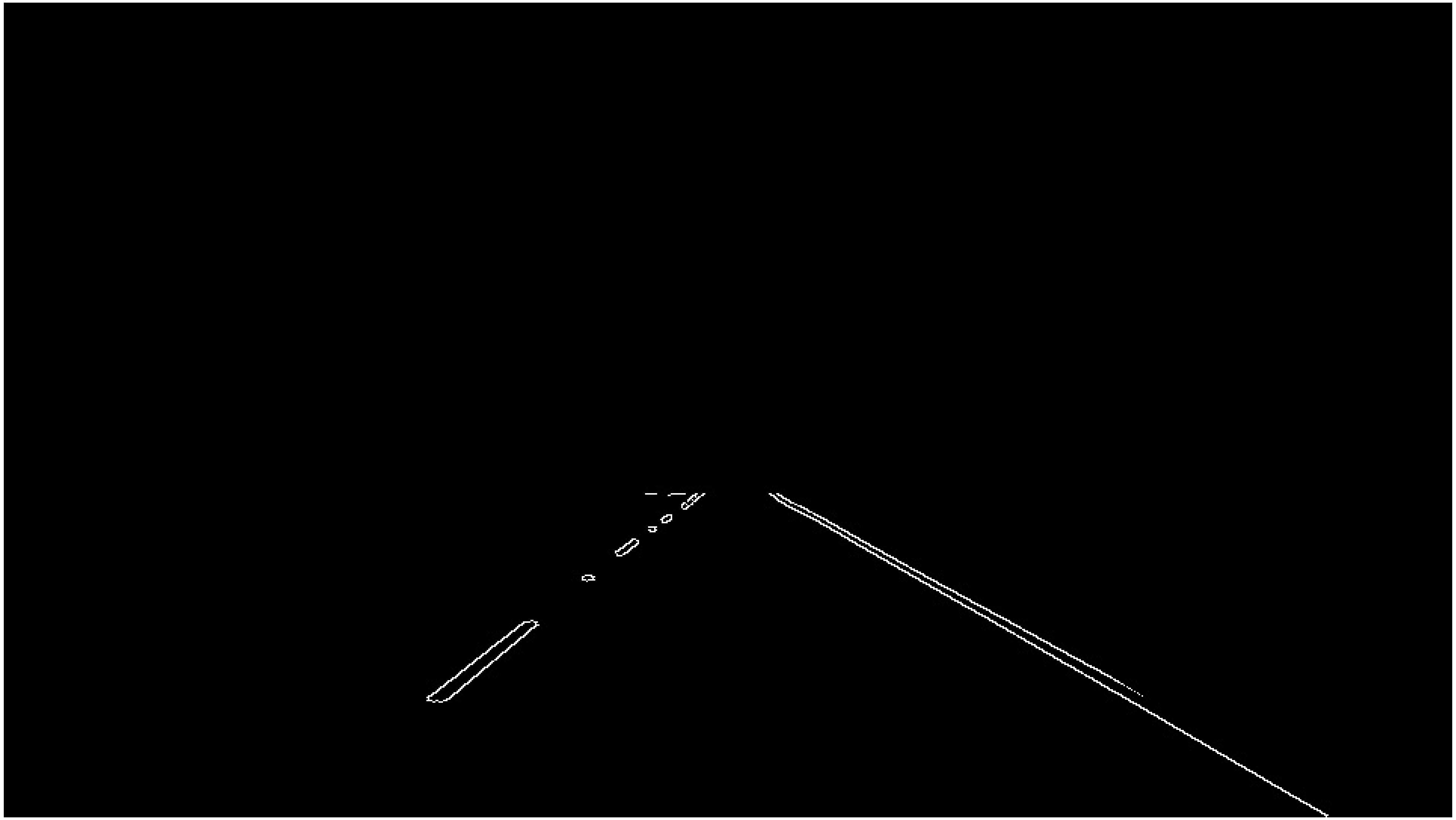
The second step is to use the Gaussian smoothing to get the blurring image with gauss kernel. The blurred image presents as



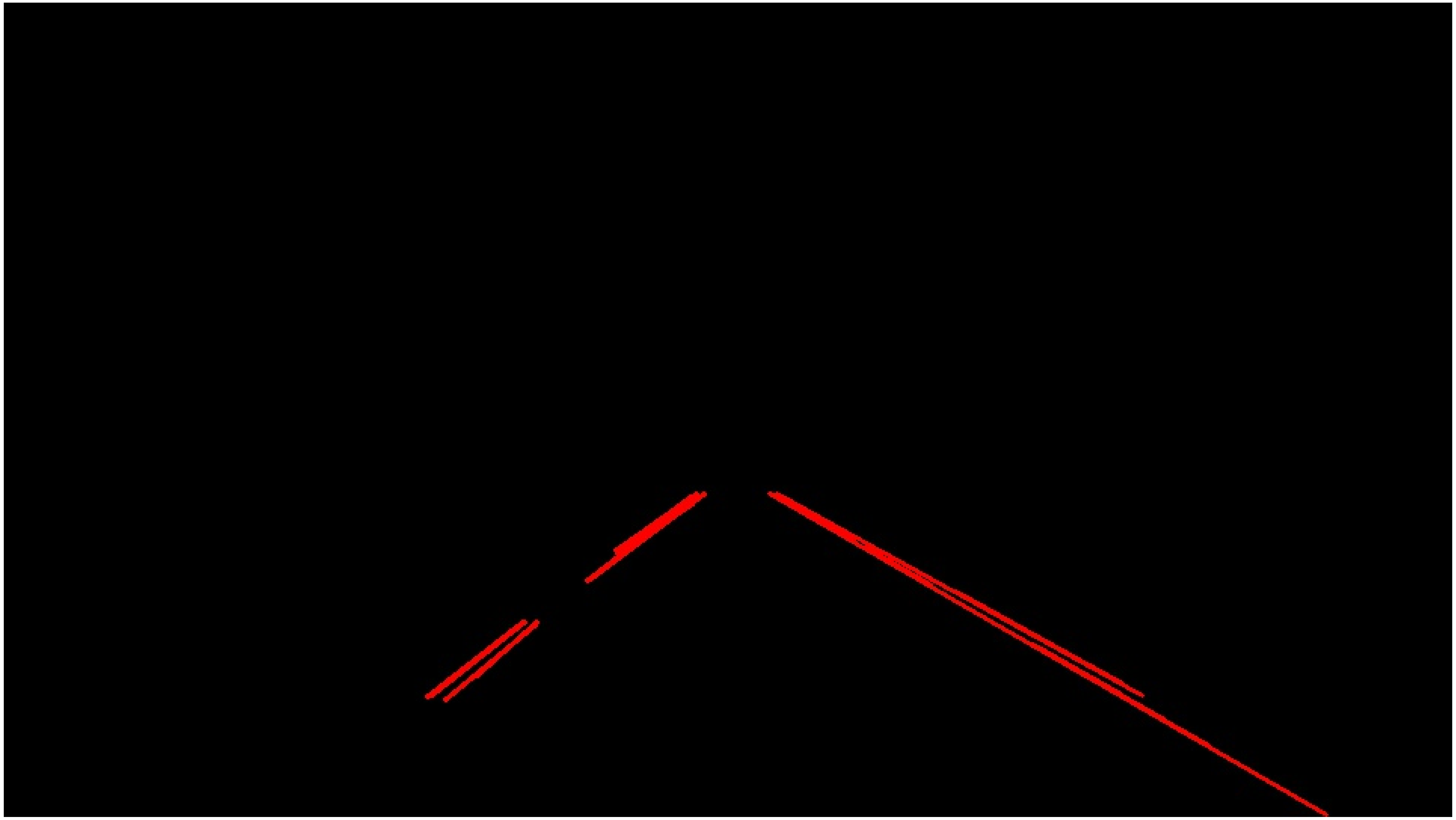
The third step is to make use of the canny function to detect the edges with the gradients algorithm, such that the outline of the possible object can be desribed in the image below



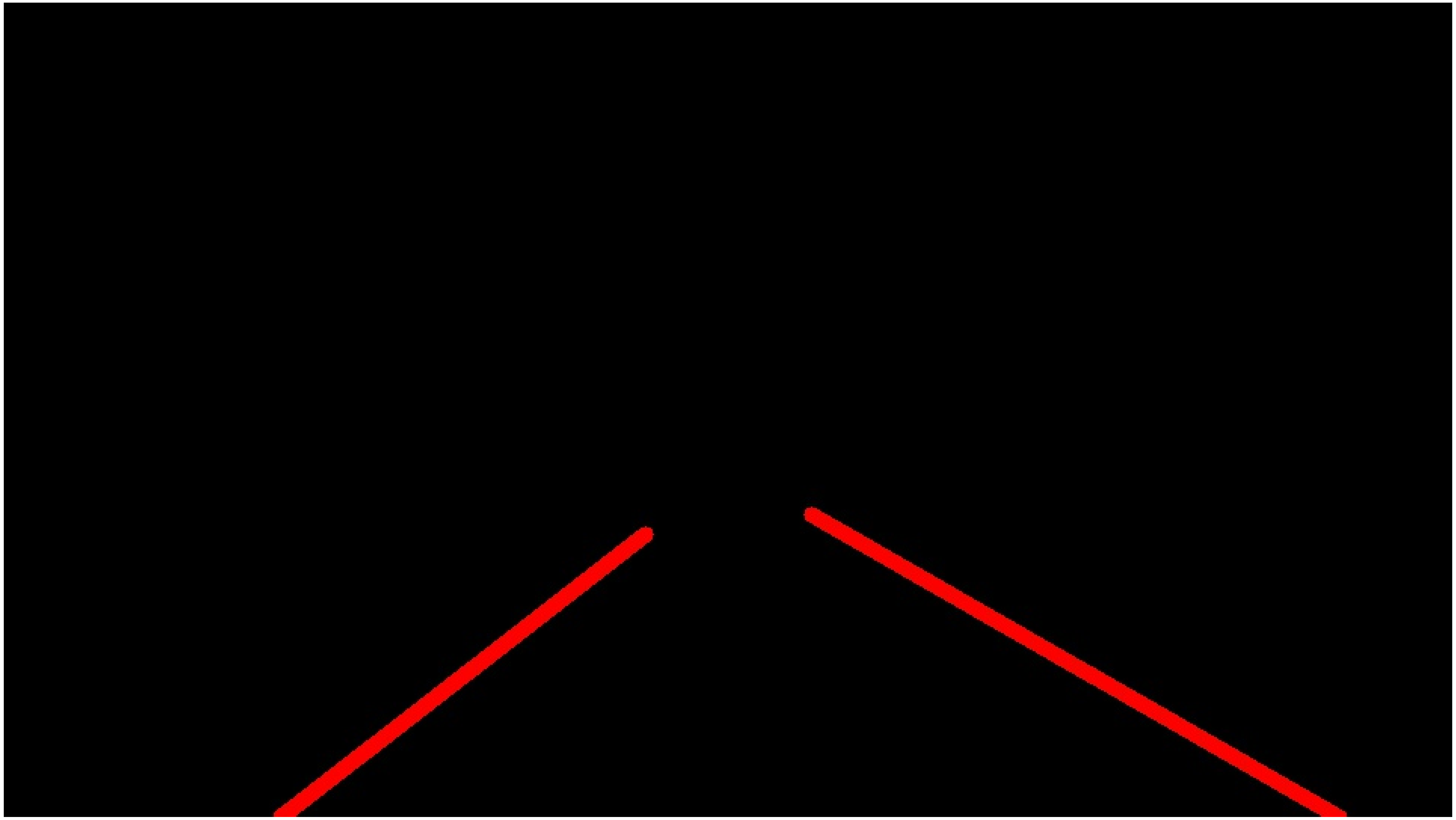
The fourth step is to give the mask region so that the lane in the most possible area can be picked out by the image filter. We use the quadratic region to mask the edges we want for the lanes.



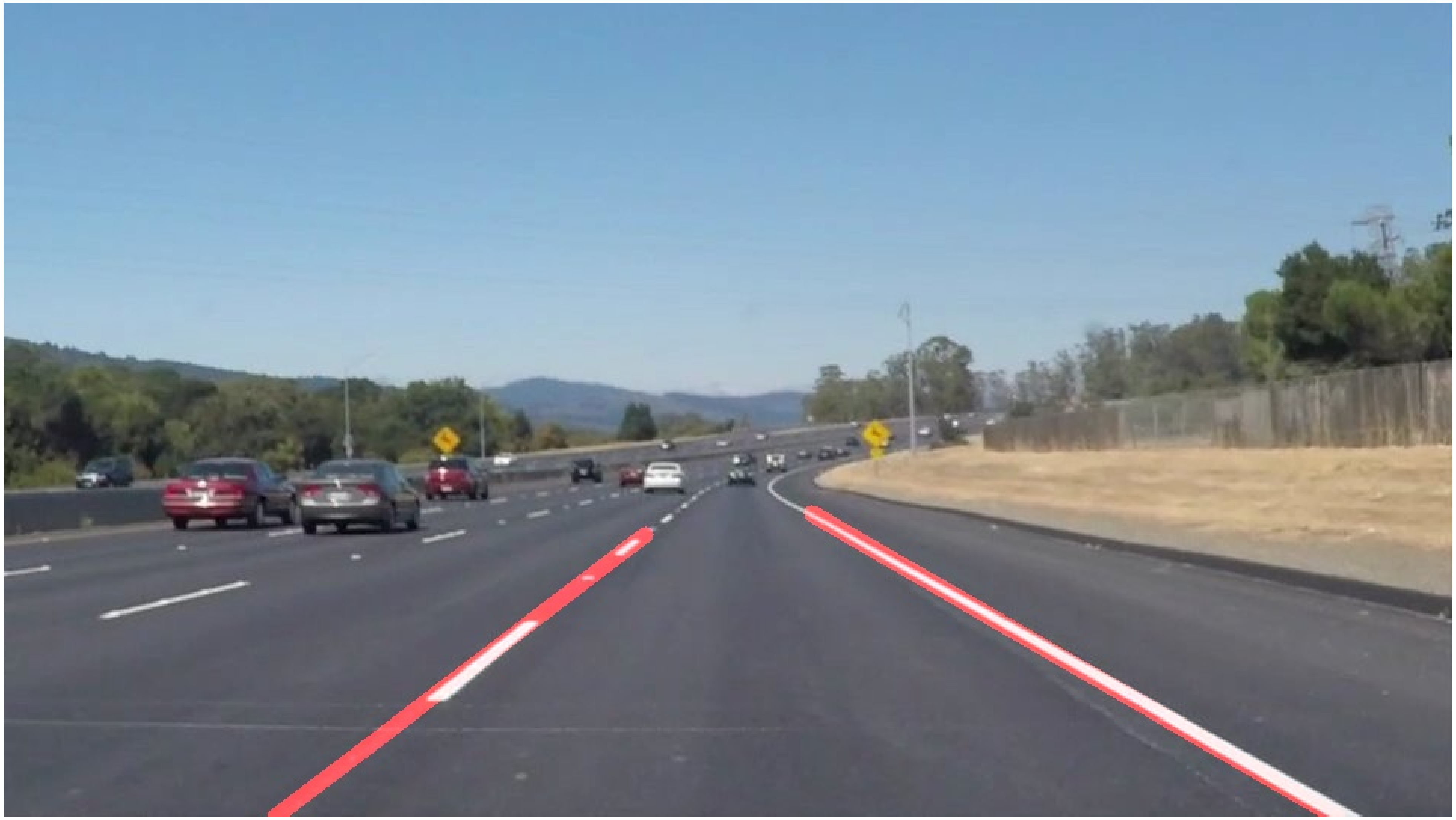
For the fifth step, we apply the Hough transform to draw the hough lines for the masked edges as this:



The sixth and the last step are ornamentary steps, such that we can change the thickness and the transparency of the lines.



By adding the hough lines on the orignal pictures, we can see that the lane can be correctly found by our program.



---

2. Identify potential shortcomings with your current pipeline

One potential shortcoming would be what would happen when my draw_lane_line() function is not so robust for the movie pictures, and the online presentation would fail. As you can see from the HTML presentation of the white_output using the process_image result, the video is actually broken.

Another shortcoming could be brought about in the function challenge_transform_lines() is that the added distance from the center of the "car dashboard" as a variable and introduced several constraints on how far the hough lines can be from a few key points from the image. All are only dependant on the image size. There is a potential to malfunction for other camera angles.

---

3. Suggest possible improvements to your pipeline

A possible improvement would be to make the size of the image as a variable in order to adapt various cameras angles

Another potential improvement could be to renew our constraints on defining the lines and the angles. We need to split the dataframe according to several new constraints so that the changing angles from different pictures can be manipulated. For defining the canny lines, we add into our line_update function some clipping parameters for lower points of X while no clipping for upper points. And the video presentation of the white_output is stable.