COMPUTER VISION

ASSIGNMENT 2

Lane Detection System using Hough Transform

Guidelines:

- Submit all your code and results in a single zip file with name RollNumber_FirstName _02.zip
- Submit single zip file containing
  a) code b) results c) readme d) report e) dataset
- There should be **Readme.txt** explaining how to run your code
- There should be **Report.pdf** detailing your experience and highlighting any interesting result. Kindly don't explain your code in report, just explain the results. **Include all experiments and all their results in the report as well.**
- Your report should include your comments on results of all the steps, with images, for example what happened when you changed thresholds in canny edge detector, impact of HSV ranges on results etc.
- In the root directory, there should be a single Jupyter Notebook code. There should be a folder called **'Results'** where you should place all the results for all images (no separate folders for each image).
- readme.txt, results folder and report.pdf should be in root directory. Root directory should be named as **RollNumber_FirstName_02**
- Your code notebook should be named as **'rollNumber_02.ipynb'**
- Follow all the naming conventions.
- For each convention, there is 3% penalty if you don't follow it.
- Email instructor or TA if there are any questions. You cannot look at others code or use others code, however, you can discuss with each other. Plagiarism will lead to a straight zero with additional consequences as well.
- Deadline for this assignment is Monday 23th November 2020 before 04:59 pm.
- Late submission will result in 0.

Overview

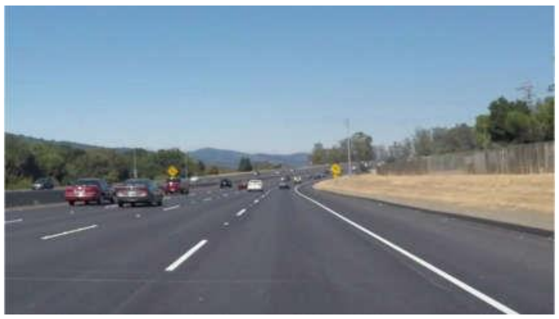
When we drive, we use our eyes to decide where to go. The lines on the road that show us where the lanes are act as our constant reference for where to steer the vehicle. Naturally, one of the first things that is required for developing a self-driving car is to automatically detect lane lines using an algorithm. In this assignment we are going to learn about the implementation of a very important component of self-driving car called lane detection.

Implementation

There are some basics steps for the implementation of lane detection, that are discussed below. But you are not restricted to follow the same steps, we encourage you to put your own thoughts and come up with a better algorithm to solve this problem. *The algorithm steps have been adopted from the Udacity Self-driving car – Project 1.*

1. Convert the input RGB image (I) to the HSV space. You can use cv2 for it function. Create a binary image B of same size (row and column) of RGB image. Set all the pixels in the B to 1.

   HSV, it is basically a different way to describe color than the classic RGB we are used to, you can learn more about it on Wikipedia. We will explain later why this conversion is necessary. Also have a look at this table http://www.rapidtables.com/convert/color/index.htm


(a) RGB

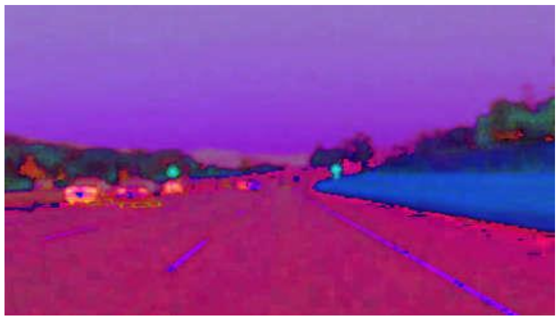
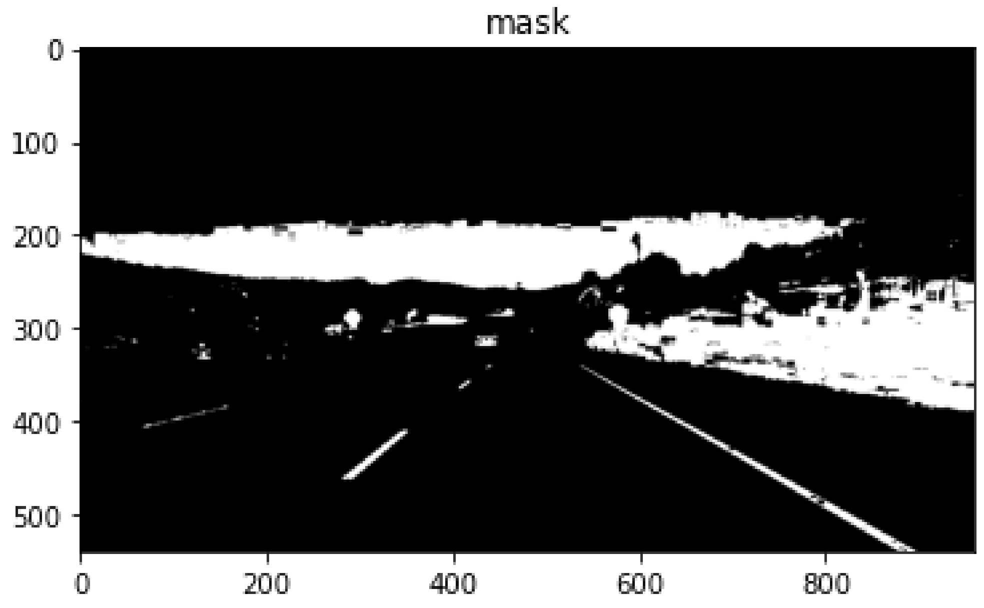

(b) HSV

Figure 1: Images in RGB and HSV color space

2. Remove the noise by applying Gaussian filter. This should also remove small artifacts and small particles.
3. Filter out ***(turn black, set to zero)*** any pixel (in the binary image B) that is not the color of the lanes (in HSV space), in our case white or yellow. Because lines on road are marked with yellow or white color. You should get a result as in Fig. 2.

   First you need to check which values of H, S and V produce a white or yellow color. Use these values of H, S and V as threshold and filter out corresponding pixel locations. Now turn black (set to zero) any pixel that is not in the filtered locations. This way you will be able to filter white or yellow pixel locations.



You will notice that HSV has allowed us to remove unwanted pixels with much more ease. We have to look to both Hue and Saturation of the pixel (which color it is and how intense the color is), giving us a room of adjustment for "value" (how dark it is). This setup allows us to handle shadows and lighting variations. To extract the yellow and white color you will need to study about HSV color space. (http://www.rapidtables.com/convert/color/index.htm)
There only remains the pixels with white or yellowish tint, with some false positives but our two lines are very clearly defined.

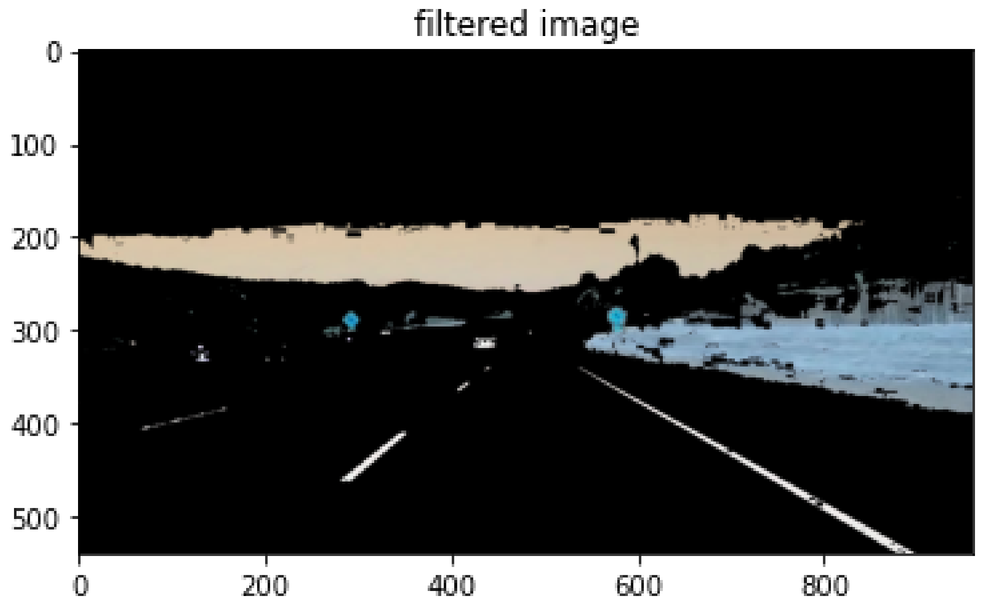


Figure 2: Output of step 3, filter out the pixels that are not color of lanes

4. Convert the input image I to the grey scale image Ig. Use the Canny edge detector to detect the edges. The Canny algorithm detects edges on a picture by looking for quick changes in color between a pixel and its neighbours, in this case between the white or yellow lane line and black pixels. Use the binary image B to remove all the edge pixels (set to zero) which are set to zero in I. You can use Python’s cv2 Canny Edge Detector or the one you designed yourself in Assignment 2.

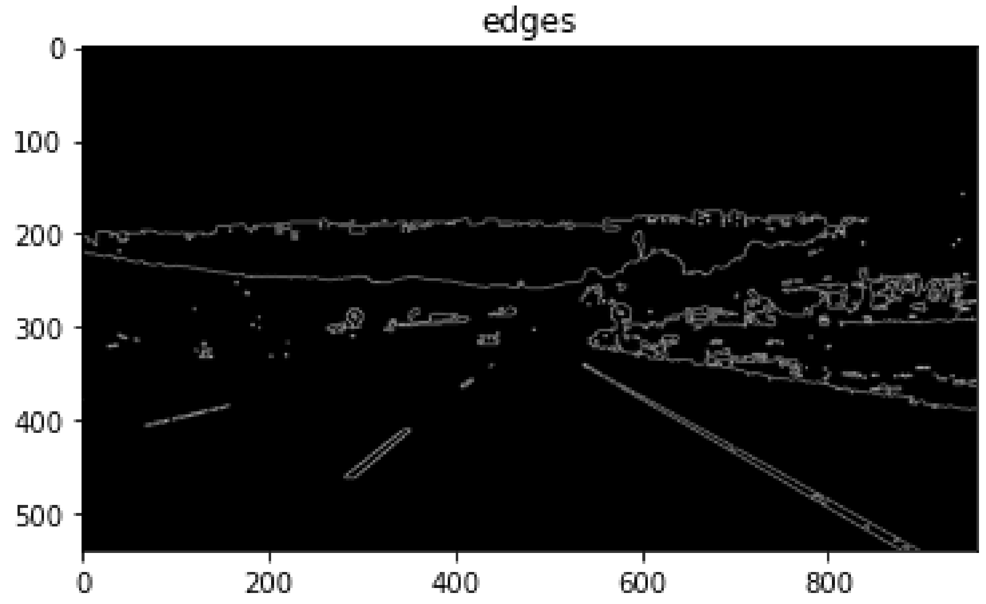


Figure 3: Output of Canny Edge Detection

5. Now we define region of interest. Given the position and orientation of the camera, you know that the lanes will be in the lower half of the image, usually in a trapezoid covering the bottom corners and the centre. You don’t want your region to be too narrow and have the lines out of our region of interest.

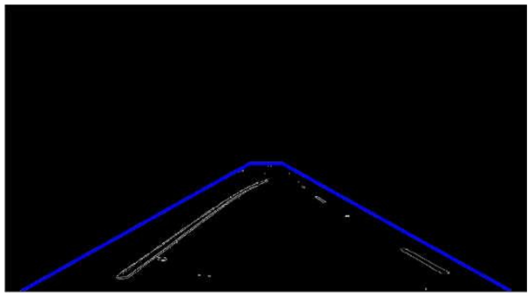


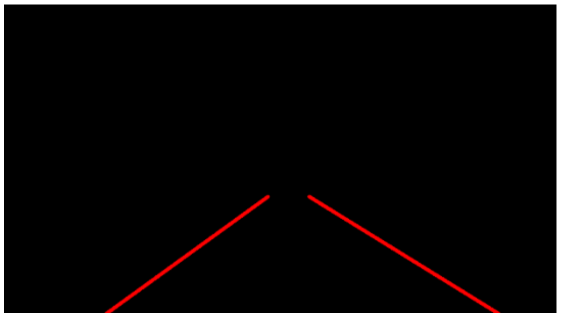
Figure 4: Region of Interest

6. Run Hough transform to detect lines on our image. What does Hough transform do? To summarize quickly, it allows us to easily extract all the lines passing through each of our edge points and group by similarity, thus grouping together the points that are roughly located on a same line.

After the above step we will still get multiple lines, but we only want 2 distinct lines (to form the lane) for our car to drive in between.

7. Filter the lines that only have horizontal slopes. Based on value of slopes distinguish and group the lines into left lines and right lines.

8. Last step is to compute the linear regression of both groups and draw the results on your region of interest. Simply, linear regression is an attempt at finding the relationship between a group of points, so basically finding the line that passes at the closest possible distance from each point. This operation will allow you to fill the blanks in a dashed lane line.


(a) Output of line fitting

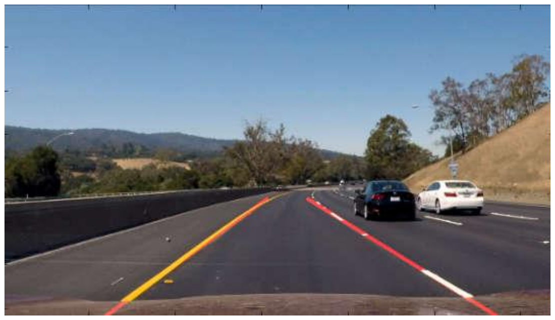

(b) Final output

Figure 5: Output of step 8

Tasks:

For Assignment 2,

- you have to implement lane detection.
    1. Conversion to HSV Color space. (5)
    2. Generate Binary Mask for yellow and white colored pixels (15)
    3. Apply Binary mask on input image to obtain filtered image. Convert the image to GrayScale (15)
    4. Compute Edges and Apply Region of Interest (15)
    5. Run Hough Transformation algorithm and filter unwanted lines (25)
    6. Apply Linear Regression on two group of lines (25)
- Take 5 test images of your choice, you can take images from internet or you can capture images of roads as well.
- For each image you have to submit:
    1. Images showing the outputs of step 2 given above. Name it as **'imageOriginalName_binaryMask.jpg'** **(10)**
    2. Images showing the outputs of step 3 given above. Name it as **'imageOriginalName_filteredImage.jpg' and 'imageOriginalName_filteredImage_gray.jpg'** **(10)**
    3. Images showing your region of interest (output of step 4). Name it as **'imageOriginalName_roi.jpg'** **(10)**
    4. Images showing the final output (step 6). Name it as **'imageOriginalName_output.jpg'** **(10)**

- Include all the results in report as well with proper headings and readable formatting. (20)
    1. Pick one image, and in the start of your report, show and analyze outputs of all the above steps (HSV image, mention HSV ranges that you chose for both yellow and white, ROI design choice and its impact on results, output of Hough transform,
    2. Mention how you filtered wrong lines and based on what criteria, show ‘before-and-after’ images when you filter horizontal lines, result after linear regression and final output). You don’t need to submit the images of this step, just show in report. To support your analysis