673 Project 2

Perception for Autonomous Robots
Faculty: Samer Charifa

Prasanna Thirukudanthai Raghavan
UID: 118287546

Department of Robotics
University of Maryland
United States

Contents

Chapter 1

Question 1:

Here, we aim to improve the quality of the image sequence provided above. This is a video recording of a highway during night. Most of the Computer Vision pipelines for lane detection or other self-driving tasks require good lighting conditions and color information for detecting good features. A lot of pre-processing is required in such scenarios where lighting conditions are poor.

1.1 Solution:

The given images are to be processed so that we get a high contrast image out of it and this is essentially done in the following way.

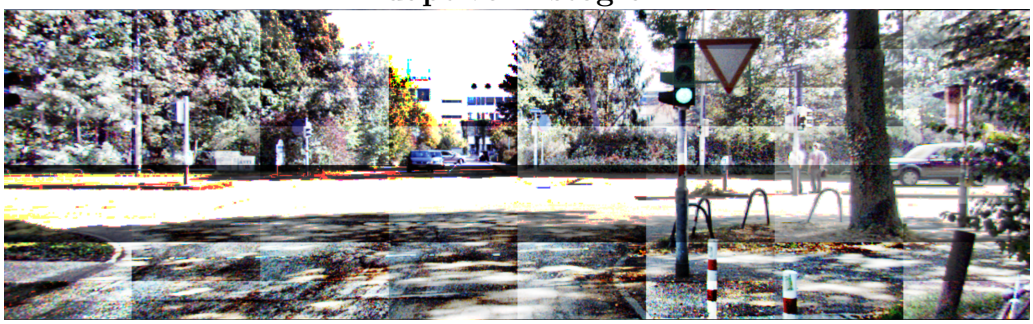
- Initially get the images from the folder and since it is named numerically it gets loaded the same way.
- Separate the r,g,b and then take the histogram for each of them individually.
- Flatten the array and for each of the pixels we increase the intensity values to get a higher contrast.
- We then add each of the previous values to get the cumulative summation so we can normalize the values by subtracting the threshold and dividing by the range.
- For the adaptive histogram we equalize each of the part of the image while only considering that part and this might cause certain parts of the image to be much different from the rest of the image.

- As seen in the above image the different sections of the image are different to each other and the smoothness of the image seems to be lost due to processing of the image with discrete windows.
- Reshape to get each of the processed channels individually

- By combining all of the channels together we get a final histogram value.

Histogram equalization

Adaptive Histogram



Chapter 2

Question 2:

In this problem we aim to do simple Lane Detection to mimic Lane Departure Warning systems used in Self Driving Cars. You are provided with a video sequence, taken from a car. Your task is to design an algorithm to detect lanes on the road, and classify them as dashed and solid lines. For classification of the line type, you have to use different colors. Use green for solid and red for dashed.

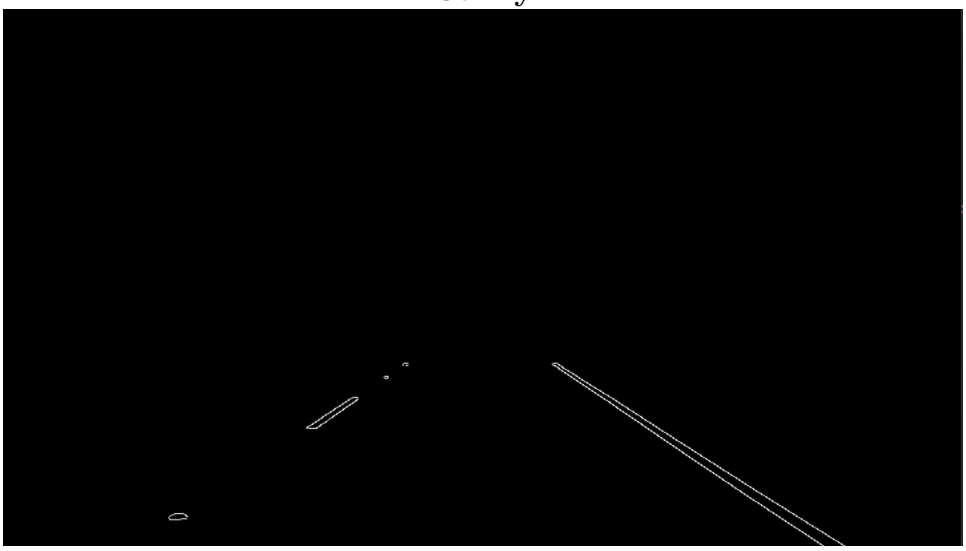
2.1 Solution:

Lane detection and classification is one of the initial steps for automation of vehicles and are are multiple ways to go about it. In this implementation we have.

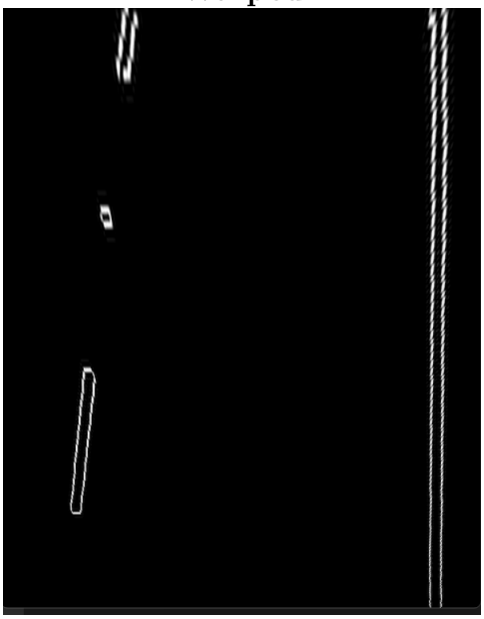
- Open the video-file and extract the frames for processing.(Can flip the frame as well to ensure robustness)
- The frame is then processed by converting to gray and blurring to remove some noise.
- Then we threshold the image to convert into binary and then warp the image using the region of interest(ROI) that is required for the road ways. The region is assumed to be a certain area by trial and error.
- Then we get the lines by using Canny edge detection and change the scene of processing to include only the ROI by warping.
- The line coordinates in the warped image are obtained by Hough lines and this is used in classification of the image(This is different from canny as we use canny to get the lines in the original frame) as we will use these coordinates to get the intensity values of the lines
- Then we take only the intensity of the line information that we got from the warped image for getting the summation of the intensities on both the sides of the image. The solid lines will always have a higher intensity summation value than the dotted lines and we can use this to differentiate between the lines.
- Once we differentiate between the lines we then draw the lines in the original frame that we receive from the video and obtain the same as the output.

Final Output

Canny



Warped



2.1.1 Problems:

- To differentiate between the lines an alternate way to use the higher of the individual value between the lines was initially tried but this logic fails when the solid lines though higher in number of intensities might just not have the highest intensity value and hence we might get a wrong classification.
- Similarly there was also an alternate way of mapping the lines was to the gradient/slope of the line values that we get from Hough lines, by drawing lines from one point to another and in this way, the points with greater distance can be noted to be a part of dotted lines and vice versa.
- Due to imperfections in either the image data or the edge detector the Hough transform groups of edge points into into a singular points through which multiple lines pass through in object space. This grouping allows for compensation for missing points or pixels on a curve.

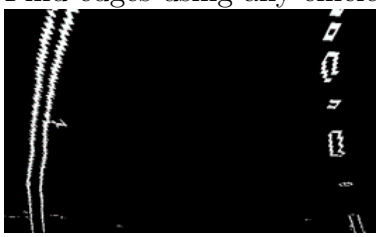
Chapter 3

Question 3:

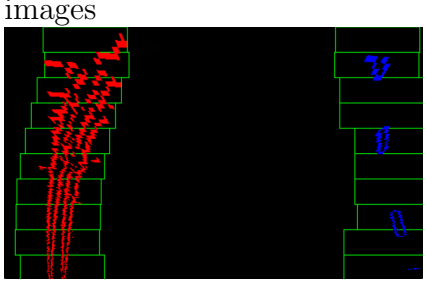
In this problem, we aim to detect the curved lanes and predict the turn depending on the curvature: either left or right turn. The dataset provided has a yellow line and a while line. Your task is to design an algorithm to detect these lanes on the road, and predict the turn. Please note that for the output of the lane detection, you have to superimpose the detected lane as shown in fig.4. Also compute the radius is curvature for the lane using the obtained polynomial equation. Refer to this.

3.1 Solution:

- Convert the RGB channel to HSV channel as that can give up better edges while processing.
- Find edges using any efficient operator, Canny has been used here



- Get the perspective Region of interest by trial and error
- After taking the maximum value along the columns of the image we then proceed to the sliding window technique.
- Using the S channel we are able to get the pixel coordinates for each of the images



- Using polyline fit the curve to the required turn so that we can predict which side the turn is taking place.
- Once we get the line fitting we are able to calculate the curvatures after knowing the predictions.