University of Maryland College Park

Professional Master of Engineering

Robotics Engineering

ENPM673 - Perception for Autonomous Robots - Project 2

Students:
Akash Agarwal (116904636)
Nupur Nimbekar (116894876)
Anubhava Paras (116905909)

Lecturer:
Dr. Mohammed Samer Charifa

March 11, 2020

Problem 1

Here, we aim to improve the quality of the video 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.
Now, using the techniques taught in class your aim is to enhance the contrast and improve the visual appearance of the video sequence. You can use any in-built functions for the same.

Given: Dataset for Problem 1

Solution:

To improve the quality of video sequence, we have tried to implement a series of methods to work on contrast adjustment. The list of methods used is as below: -

1. Histogram Equalization: -

   Histogram equalization is a method in image processing of contrast adjustment using the image's histogram.This method usually increases the global contrast of many images, especially when the usable data of the image is represented by close contrast values. Through this adjustment, the intensities can be better distributed on the histogram. This allows for areas of lower local contrast to gain a higher contrast. Histogram equalization accomplishes this by effectively spreading out the most frequent intensity values.

2. CLAHE (Contrast Limited Adaptive Histogram Equalization): -

   The first histogram equalization we just saw, considers the global contrast of the image. In many cases, it is not a good idea. So to solve this problem, adaptive histogram equalization is used. In this, image is divided into small blocks called "tiles". Then each of these blocks are histogram equalized as usual. So in a small area, histogram would confine to a small region (unless there is noise). If noise is there, it will be amplified. To avoid this, contrast limiting is applied. If any histogram bin is above the specified contrast limit, those pixels are clipped and distributed uniformly to other bins before applying histogram equalization. After equalization, to remove artifacts in tile borders, bilinear interpolation is applied.

3. Gamma Correction

   Gamma correction establishes translation between the sensitivity of our eyes and sensors of camera. Gamma correction is also known as the Power Law Transform. First, our image pixel intensities must be scaled from the range [0, 255] to [0, 1.0]. From there, we obtain our output gamma corrected image by applying the following equation:

$$O = I\hat{}(1/G)$$

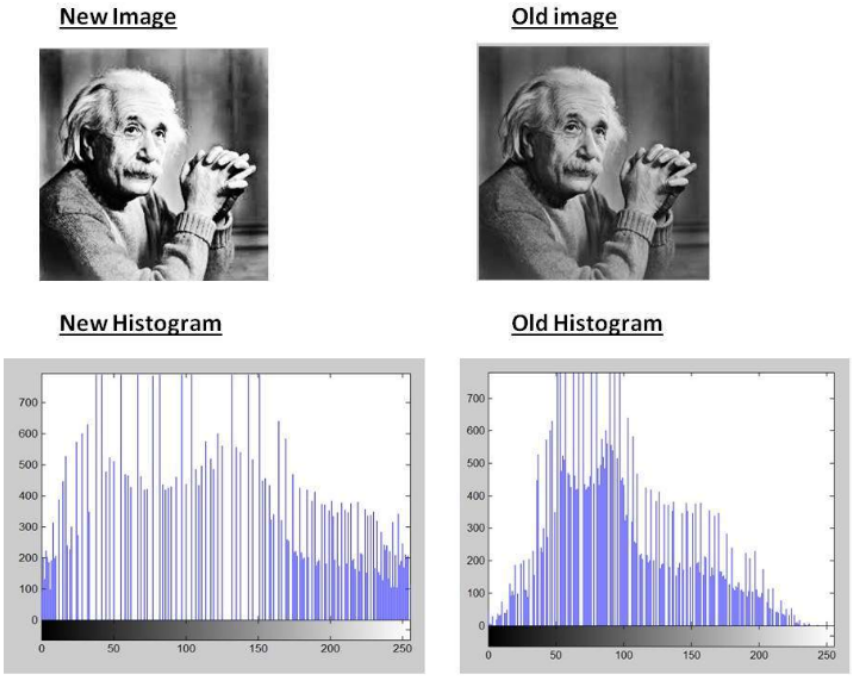
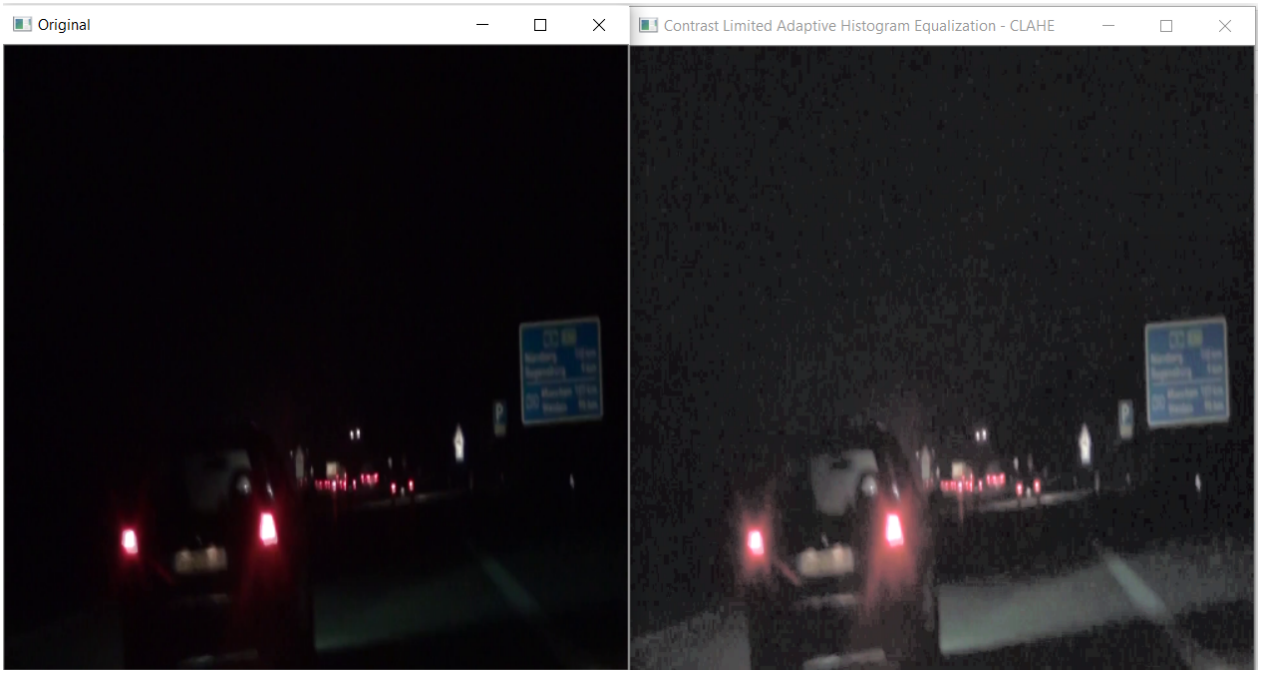


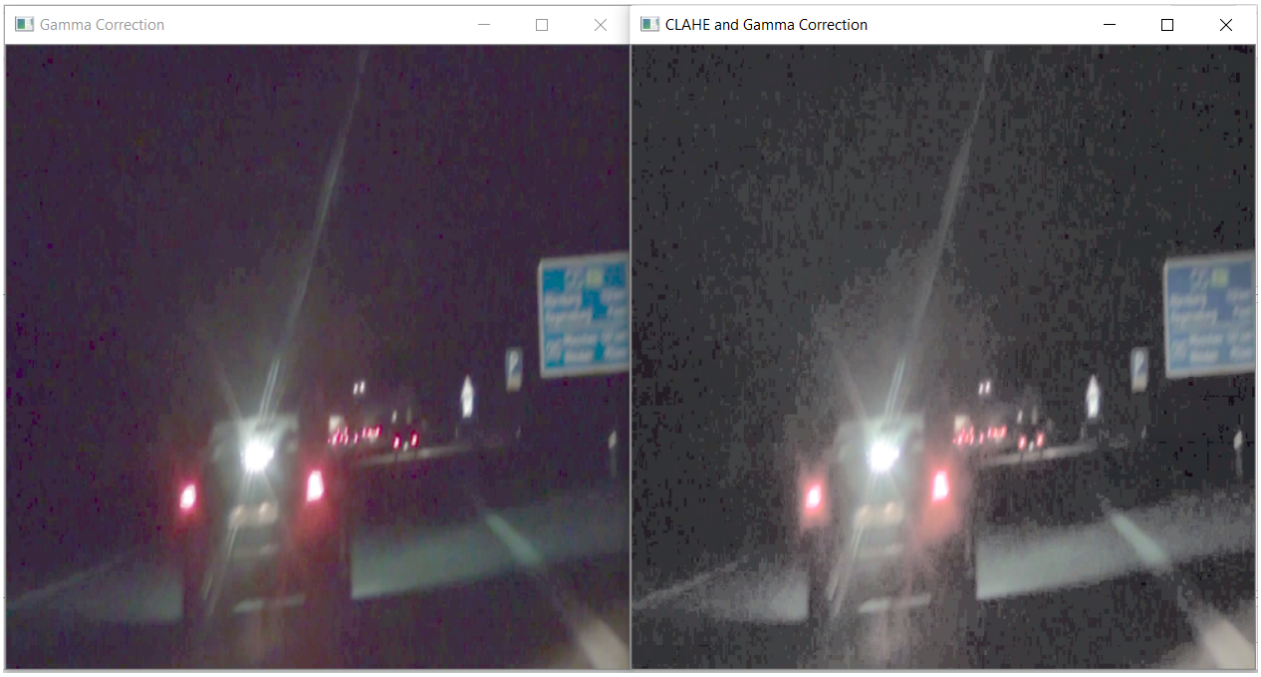
Figure 1: Example of how Histogram Equalization modifies the contrast level of image data

Gamma values < 1 will shift the image towards the darker end of the spectrum while gamma values > 1 will make the image appear lighter. A gamma value of G=1 will have no affect on the input image.

Figure 2: Our original image (left); Gamma correction with G < 1 (center); Gamma correction with G > 1 (right)



((a)) Original frame from the video (Top Left); CLAHE (Top Right)



((b)) Gamma Correction (Bottom Left); CLAHE and Gamma Correction (Bottom Right)

Figure 3: Output for Problem statement 1

Problem 2

Lane Detection to mimic Lane Departure Warning systems used in Self Driving Cars.

Given: **Dataset for Problem 2**

Solution:

In this problem our aim was to reconstruct a Lane Departure Warning system which is used more frequently in the field of Self Driving Cars. General Pipeline structure provided in the Problem statement was used while developing the algorithm for implementation.

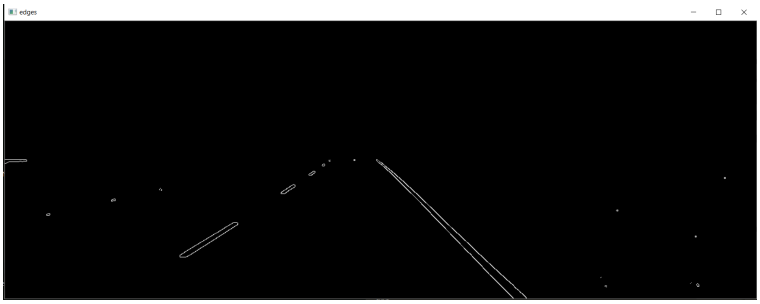
Step 1: Preparing the Input

We started with getting the video input and extracting a frame from the video. Then the method of cv2.undistort(), and cv2.medianBlur() were used to remove distortion and noise.

Output video after removing distortion and noise

The image frame is first converted to grayscale and then the function of cv2.Canny() is used for edge detection. Masking is performed on the top part of the frame t concentrate more on the lane part of the frame and increase the precision of detection. The part of frame without the mask defines our region of interest(ROI). To mask we created a function which uses cv2.fillPoly() to create the mask on the image.

((a)) Depicts how the frame will look if we show the mask that we have applied



((b)) Implementation of canny edges on backend

Figure 4: Fetching Lane information from frame

Step 2, 3 4: Detect Lane Candidates, Refine the Lane Detection and Turn Prediction

Hough Lines: -

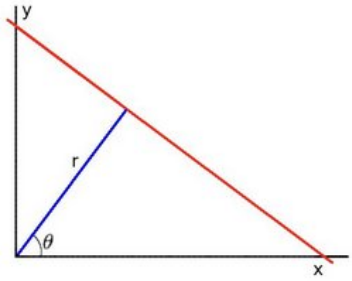
The main idea of using Hough Lines is to identify straight lines. Now the straight line equation is given by: -

$$y = ax + b$$

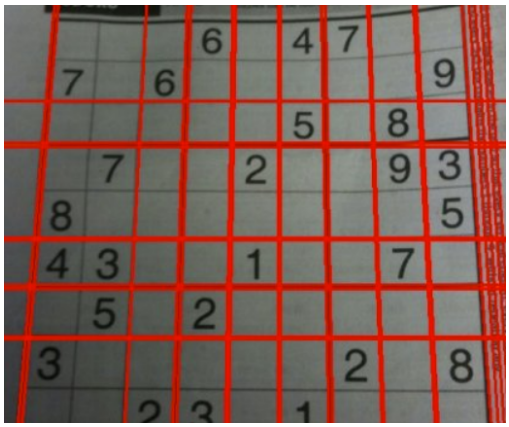
Where, a is the slope and b is the intercept for that particular line. Now when we don't have these parameters, it is very difficult to find any line. But we know that lines can also be represented in terms of (ρ, θ) in polar system, where ρ is the shortest distance of the line from origin and θ is the angle formed in between the x-axis and the distance line.One of the benefits of such representation is that we can describe vertical lines by ρ and θ which is impossible by using only (a, b) parameters in cartesian system.

for any line if we have ρ and θ, Then, the following equation is satisfied for each $\boldsymbol{x}_i, \boldsymbol{y}_i$ point belonging to this line:

$$\rho = x\cos(\theta) + y\sin(\theta)$$



((a)) Representation of line in Cartesian and Polar system



((b)) Representation of Hough Lines

Figure 5: Concepts for Houghlines

Probabilistic Hough Transform: -

Probabilistic Hough Transform is an optimization of Hough Transform. It doesn't take all the points into consideration, instead take only a random subset of points and that is sufficient for line detection. Just we have to decrease the threshold.

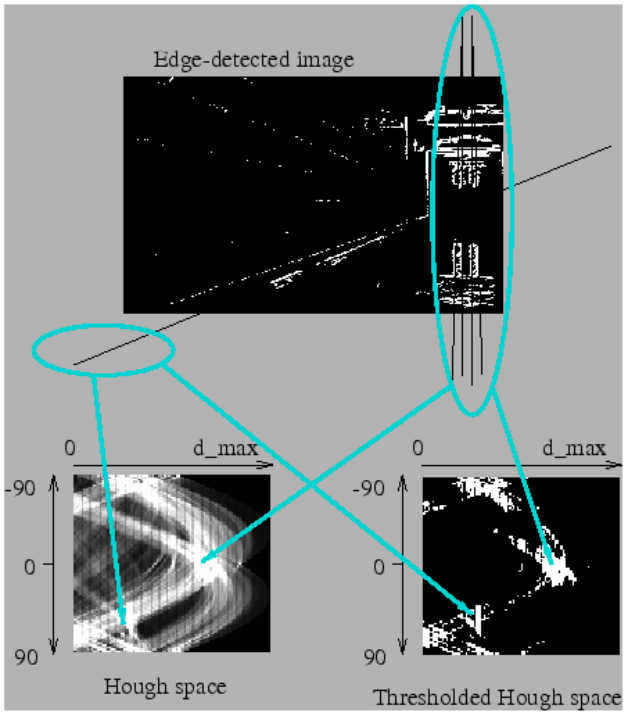
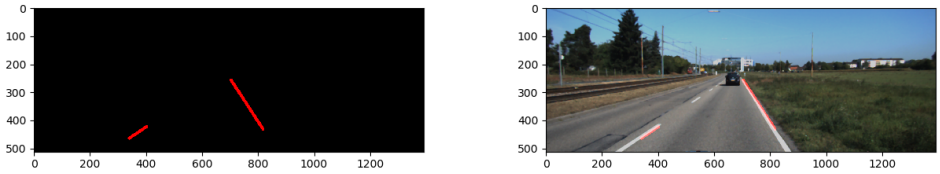


Figure 6: Example of the Hough space of an image. The figure shows the result of an edge detection and its corresponding Hough space. And we can see the difference between space with and without threshold

Overview of how we implemented concepts of Hough Lines: -

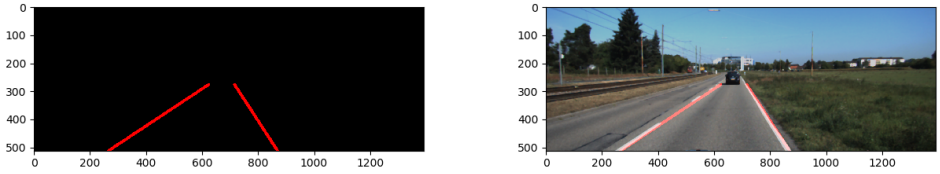
1. To find Slope: -

   We used method of cv2.HoughLinesP() to find the slope of the lines in the warped image. The idea where negative slope indicates line to the left of lane and positive slope indicates the line to the right of the target lane.



(a) Initial Houghlines

(b) Initial Houghlines on the frame



(c) Extrapolated Houghlines

(d) Extrapolated Houghlines on the main-frame

Figure 7: Initial frame with the Houghlines and the extrapolated Houghlines

When we detect lines from the initial image we extrapolate them to maintain the connectivity. The initial and extrapolated images look as above.

To find four Force Points: -

We find Four force point for image so that we can warp it accordingly. Initially we tried using cv2.HoughLinesP() as that would have ensured that no hard-coding is required for four point detection. But as we progressed in the project, we observed a lot of noise in the data at evey point. With more filtering that can be eliminated.

Logic for Turn Prediction: -

We do the turn prediction based on the logic of slope for lines detetcted with Houghlines method.

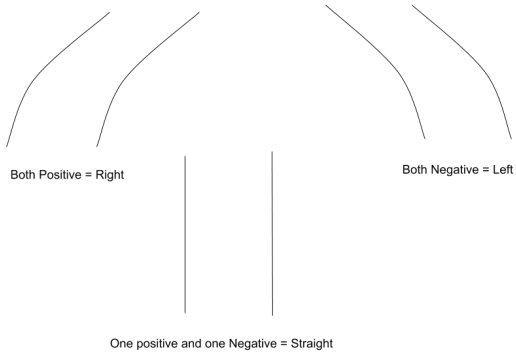
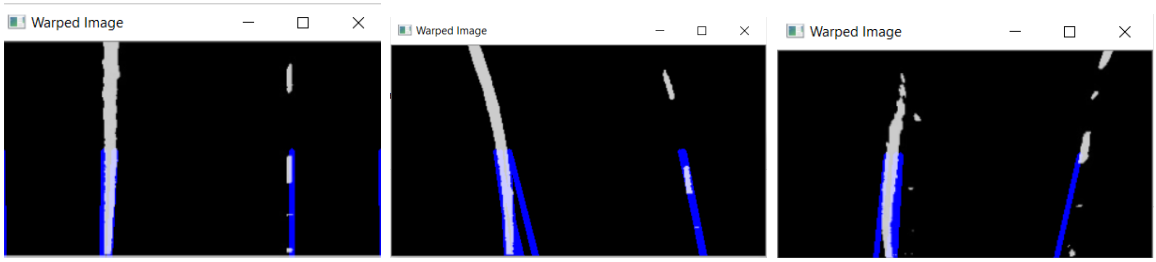


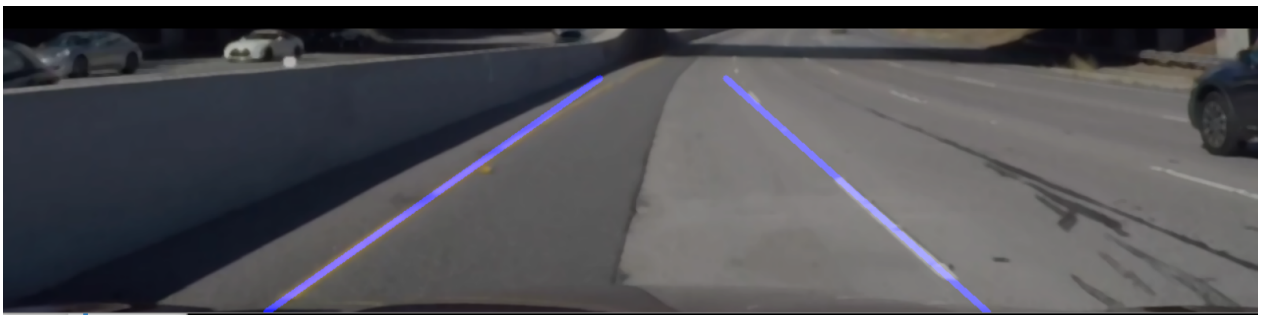
Figure 8: Turn Prediction Logic



(a) Straight Path (b) Turn Left for Path (c) Turn Right for Path

Figure 9: Turn Prediction on warped image

((a))



((b))

Figure 10: Lane detection in Main Video Frame