ENPM673: Perception for Autonomous Robots

Project 2

Umang Rastogi UID: 116773232

Sayani Roy UID: 116818766

Prateek Bhargava UID: 116947992

Date: 11 March 2020

1. Problem 1

In this problem we are pre-processing the low light video given to us trying to improve the contrast and video quality. There are various methods to perform pre-processing in order to improve the contrast and given below are some of the methods compared.

The first image displays the original image without video enhancement techniques applied.

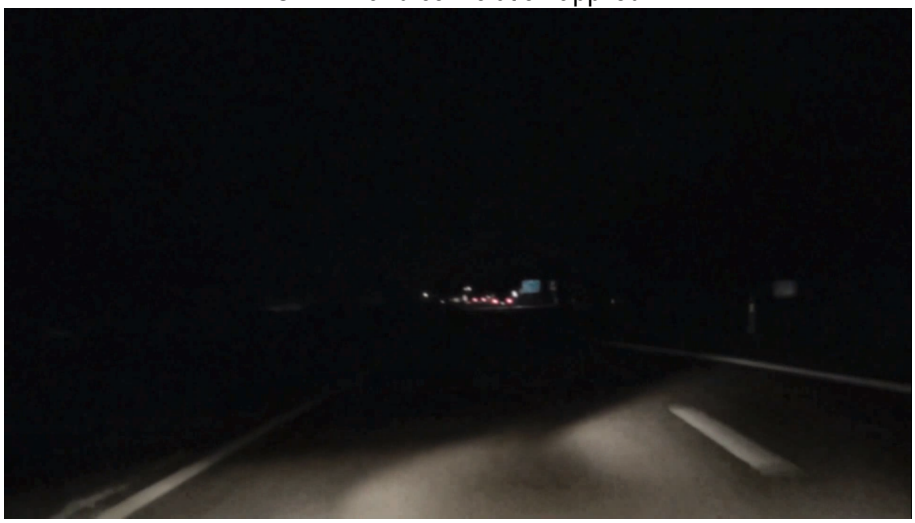
Original Image without enhancement

Now, applying the CLAHE (Contrast Limited Adaptive Histogram Equalization) we obtain the image below. In this method we divide the image into blocks of 8x8. we then plot the histogram to check whether to clip the blocks or not. Then we compute the CDF and transformation function for each block in the center pixels. Then the remaining pixels are transformed with respect to the center pixel.

CLAHE Applied 

Now we use convolution method to apply filter and then use CLAHE method. In this we place our filter in one corner of the image, such that it overlaps with 9 pixels present in the corner. Then we multiply the corresponding values of the filter and the pixels it overlaps and then we finally sum them up. We then replace the center of the pixels of the image which is overlapped by the filter with the calculated vale of sum. We repeat this process by moving our filter to the right by one pixel. Here we use the function cv2.filter2D() present in Opencv to convolve a kernel with an image.

CLAHE and convolution applied



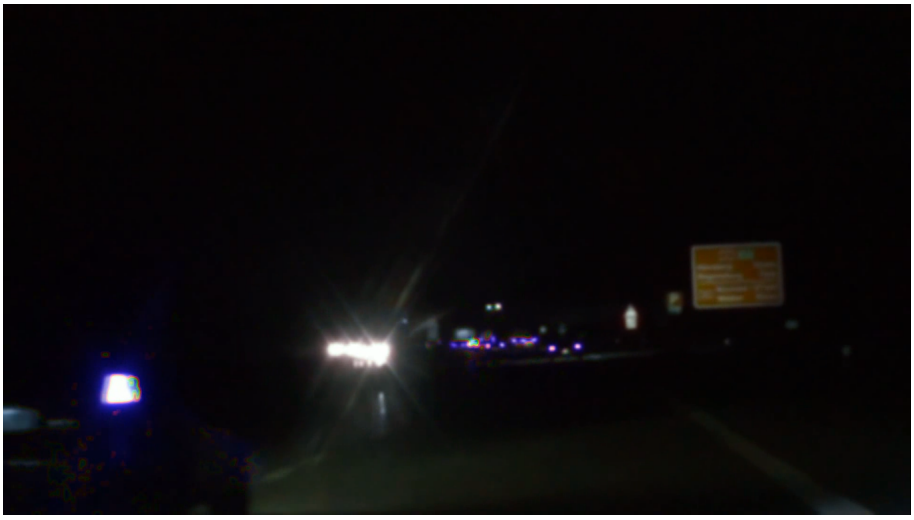
Alpha Beta method applied

Gamma Correction is a method which controld the overall brightness of an image. In this method we first scale our pixels from the range [0,255] to [0,1]. For gamma corrections in darker toned imageswe use gamma > 1 so that more colors are observed which having a darker tone. For images which are too bright we use gamma < 1, so that more colors can be observed which have a bright tone.

Gamma Correction applied

In OpenCV we use the function cv2.blur(). This is a method of smoothing images by reducing the amount of intensity in the image between the neighbouring pixels.

Mean Filter applied



In Opencv we use the function cv2.medianBlur(). This function replaces the center pixel with the median of all the values present in the kernel. We perform this method to remove salt and pepper noise as can be seen in the image below. The center element is replaced which was previously present in the image.

Median Filter applied 

2. Problem 2

2.1 Introduction

In this problem, the following pipeline has been followed to do simple Lane Detection to mimic Lane Departure Warning systems used in Self Driving Cars.

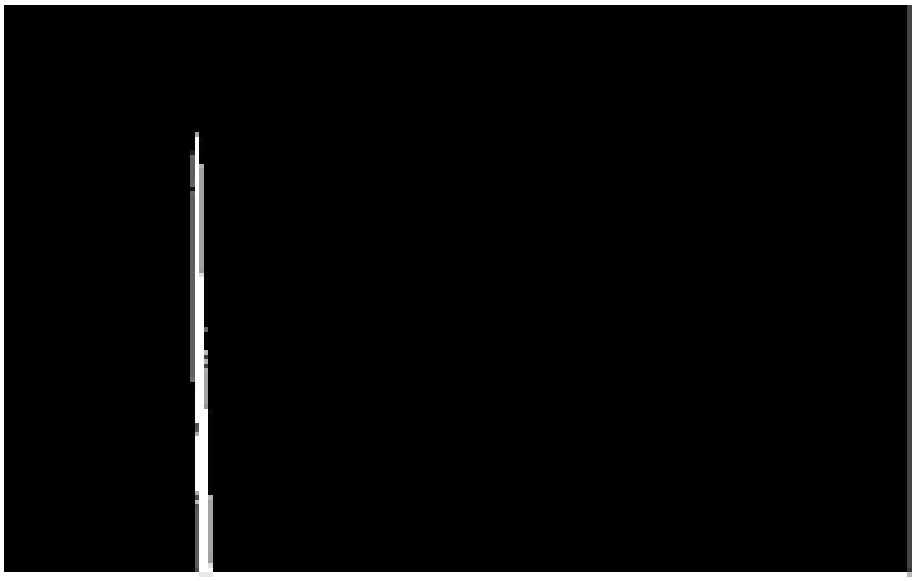
Correction of distortion and removal of noise: The given camera calibration matrix and distortion matrix are used to remove distortion from each frame of the video. The openCV function undistort() has been used for this step. The function takes the frame, camera calibration matrix and distortion matrix as input and returns an undistorted frame as output. To remove the noise from the undistorted frames, openCV function fastNlMeansDenoisingColored is used. The function suppresses the noise from each frame.

Extraction of ROI: For this problem, the region of interest is bottom half the image where the lane of the road is visible.

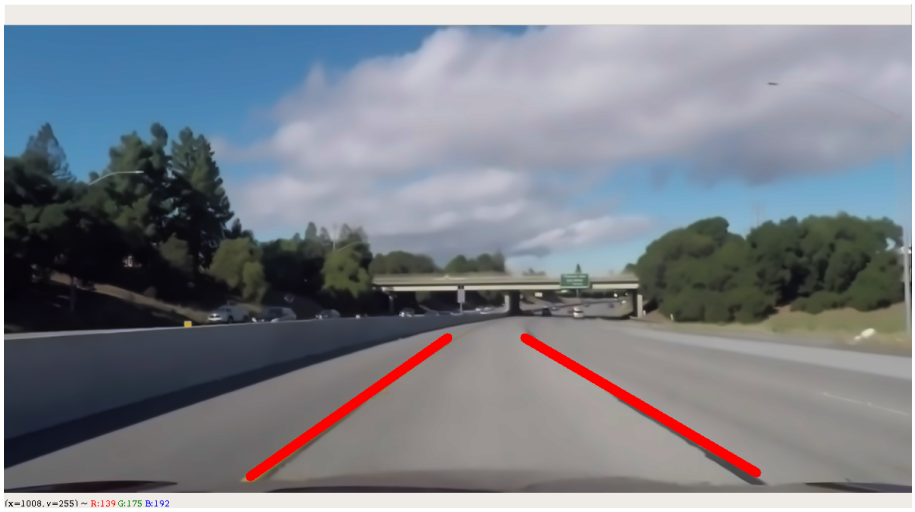
Detection of lane candidates: After extracting the ROI, we warped the image of the road. This gave us the view as seen from above the road. OpenCV function goodFeaturesToTrack was then used to detect corners in the image. The image is divided into a leftmost section and a middle section. The corners near the leftmost section are used for the yellow lane lines while the corners near the middle section are used for the white lane lines. OpenCV function Lines in then used to plot the lines on the image.

Warped image of the road

Mask for white lane lines

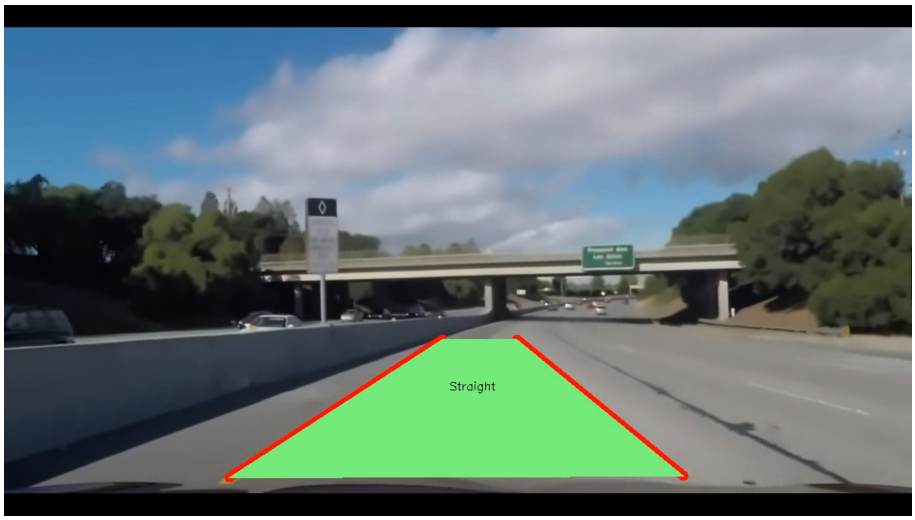


Mask for yellow lane lines



Detected lane lines

Prediction of turns: To predict the turns, the coordinates of the lanes were used. A point which is the mean of the coordinates of left and right lane is derived and checked if it is the right or left section of the image. An average of the image frames were taken to avoid errors in turn prediction. The turn is then predicted according to the position of the mean point. That is, if the mean point belongs to the right section, the turn is predicted as 'right' and if the mean point belongs to the left section, the turn is predicted as 'left'. In case, no turning is predicted, the image will show 'straight'.



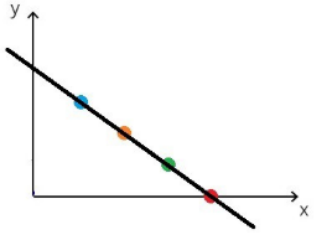
Prediction of turn

2.2 Homography

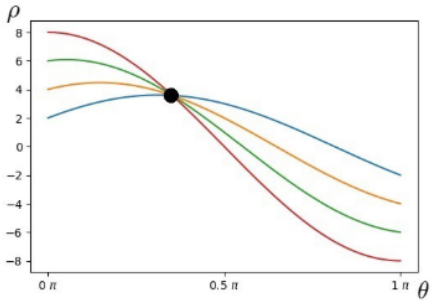
Homography can be explained as simply as a transformation that maps the points in one image to the corresponding points in the other image. The warped image of the road is obtained using homography concept. We used openCV function findHomography and warpPerspective for that purpose and then applied goodFeaturesToTrack function to get corners in the image frame.

2.3 Hough Lines

Hough Lines transform in its simplest form is used to detect straight lines. This transformation can be applied on images after edge detection techniques like Canny.
A straight line can be represented in terms of 'ρ' and 'θ', where the parameters 'ρ' is the distance from the line to the origin and 'θ' is the angle of the line.
For any point detected in the image, a group of lines can be drawn that goes through that point, i.e, for all values of (ρ,θ) there will be a line passing through the point. This group of lines for all the points in the image plane when plotted in Hough space, different sinusoidal curves are generated. The points where most of the curves intersect in the graph represent straight lines in the image plane. Now, those corresponding values of (ρ,θ) are selected to draw lines on the image. This is how Hough Line transform results in drawing of lines over the edges detected in the image.



Points which form a line



Bunch of sinusoids intersecting at one point

We tried probabilistic Hough Line transform after Canny edge detection step to identify the lanes on the road. However, there was error in the extrapolation of the coordinates of the best fit lines. This resulted in intersection of the lines which was undesired. Hence, the Hough Line transform was not used to identify the lanes.

2.4 Pipeline generalization for similar datasets

2.5 Problems encountered

Edge Detection: The next step after frame rectification was to detect the edges present in the frames. For this reason, Canny Edge Detector was used. Probabilistic Hough Line transformation was applied on the resultant image from Canny Edge detection. However, due to error in extrapolation of the good fit lines, Canny Edge detection and Hough Line transformation was not finally applied in the code.

3. Links

This section includes links to the google drive where the video outputs are stored and the GitHub repository.

- Link to folder containing all the output videos: `https://drive.google.com/drive/folders/1JGIGTMWVbhL-LIZfgKRpVNANmlEMGXFP?usp=sharing`

- GitHub repository link : `https://github.com/urastogi885/advanced-lane-detection`

4. References

- Understanding the logic behind Hough Line transformation