ENPM673 - Perception for Autonomous Robots Project 2

Sandeep Kota Sai Pavan
116911603
University of Maryland
College Park, MD - 20740
Email: skotasai@umd.edu

Satyarth Praveen
116752749
University of Maryland
College Park, MD - 20740
Email: satyarth@umd.edu

Revati Naik
116723015
University of Maryland
College Park, MD - 20740
Email: revatin@umd.edu

I. Introduction

This project aims on introducing the concepts of computer vision used in autonomous vehicles like night video enhancement and lane detection. The problem 1 deals with enhancing the details of a video sequence that has been recorded in low lighting conditions to enhance the details and features present in the image frames. The second problem is on lane detection that has to be performed on two different scenes.

II. Problem - 1

This problem aims to improve the quality of the images recorded during the night time. Images recorded in the night tend to capture very little information as the amount of light reaching the sensor is less which is not sufficient to perform image processing operations successfully.

Image parameters like colors, brightness, contrast, edges, etc are mostly used to extract useful information from the images. All of these tend to fade away in the dull lighting conditions and some preprocessing is mandatory before the night time images can be used to extract useful information from it. Below mentioned are some details regarding the techniques tried to improve the night images.

The first thing that was tried was the histogram equalization. The image obtained after this had extreme levels of noise as every tiny aberration within the dark regions was amplified after the equalization. Some smoothing filters such as gaussian, bilateral, guided, etc. were used to pacify this noise and get rid of the artifacts from the output, but this step did not seem very helpful.

The next thing we tried was playing with different color spaces. The image was converted to HSV and the white and yellow colors were identified to highlight the lane pixels. But the Value (V) channel dominates towards the darker version of the colors and there is not enough information to recover the colors from the Hue (H) information of the image frame.

The thing that finally worked and gave a comparatively better output was the manipulation of brightness and contrast values. The following equation was used to manipulate these parameters.

$$I[x, y] = \alpha \times I[x, y] + \beta \tag{1}$$

The values for α and β that are used in the assignment and that give a reasonably good output image are 2.5 and 40 respectively.

The output of this step gives better information about the roads and sign boards within the scene. The downside of using this technique is that the headlights and the tails lights of the vehicles create a huge blob of white pixels around them. This might hinder some of the other processes.

This equation has the tendency to return output values that do not lie within the image intensity range (0 - 255). Hence it needs to be re-scaled to ensure that the correct intensity values are used in further processing.

A comparison of the original and improved image frames is shown in figure 1.

III. Problem - 2

A. Prepare the input

Given the distortion parameters for the camera we correct the image taken from this camera for the distortions. Using the distortion parameters and the camera matrix, a new distortion corrected camera matrix is generated.

Every input coordinate of raw image is multiplied with the new camera matrix to get the new coordinates in the distortion corrected image frame.

For data set 1:

$$CameraMatrix = \begin{bmatrix} 903.76 & 0.0 & 695.75 \\ 0.0 & 901.97 & 224.25 \\ 0.0 & 0.0 & 1.0 \end{bmatrix} \tag{2}$$

$$\text{Distortion Coefficients} = \begin{bmatrix} -0.364 & 0.179 & 0.000603 & -0.000392 & -0.0538 \end{bmatrix} \tag{3}$$

$$NewCameraMatrix = \begin{bmatrix} 629.22 & 0.0 & 692.40 \\ 0.0 & 635.61 & 220.33 \\ 0.0 & 0.0 & 1.0 \end{bmatrix} \tag{4}$$

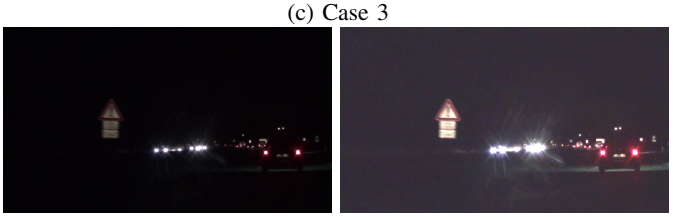
For data set 2:

$$CameraMatrix = \begin{bmatrix} 1154.23 & 0.0 & 671.63 \\ 0.0 & 1148.18 & 386.05 \\ 0.0 & 0.0 & 1.0 \end{bmatrix} \tag{5}$$

(a) Case 1

(b) Case 2

(c) Case 3



(d) Case 4

Fig. 1: Night vision image enhancement - Original Image (left), Improved Image(right)

$$\text{Distortion Coefficients} = \begin{bmatrix} -0.243 & -0.048 & -0.0013 & -0.000088 & 0.0221 \end{bmatrix} \quad (6)$$

$$NewCameraMatrix = \begin{bmatrix} 975.56 & 0.0 & 685.19 \\ 0.0 & 958.94 & 388.21 \\ 0.0 & 0.0 & 1.0 \end{bmatrix} \quad (7)$$

B. Detect Lane Candidates

For detection of lane candidates, we have two approaches. First is by using Hough Line Transform on the edge detection outputs and the second is by thresholding the lane entities and extracting the lane features from the histogram peaks for the lane pixels in the mask.

We apply both the techniques in this project.

1) Hough Line Transform: In this approach, we work on the undistorted and denoised image. Canny edge detection is used to obtain the edges in the image which is used as the input for finding the Hough Line Transform. Refer figure 2. We detect the lane candidates for both the left and the right lane of the car. For computing the hough lines we consider the θ values to be in between 20 and 60 degrees for the left lane candidates and in between 110 and 160 degrees for the right lane candidates. Refer figure 3 to check for the hough lines on the lanes.

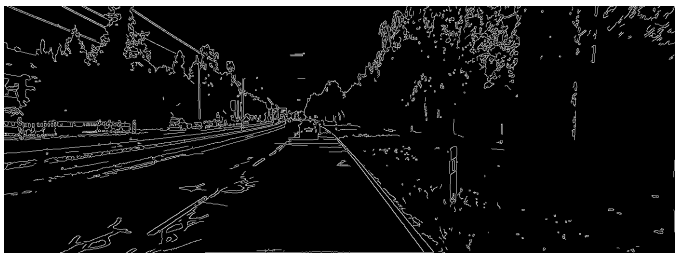


Fig. 2: Image Frame after Canny Filtering

Fig. 3: Hough Line Transform on the image

*2) Lane Detection using Color Threshold - **Dataset 1**:* The white lane markers in the image are extracted based on their intensities. All the pixels above the threshold value of **230** in the grayscale warped image are masked and we proceed with separating them into left and right lane components. We calculate the columnwise frequency of non-zero pixels in the image into left half and right half of the image. The peak value of the pixel count along the columns in the left half of the image defines the left lane pixel candidate and the peak value of the pixel count along the columns in the right half of the image defines the right lane pixel candidate. Refer to figure 4d and 5c. We consider a boundary window of 15 pixels on both sides of the maxima points and remove all pixels in the remaining region. The resultant is an image that shows just the lane entities in the image. These indices of left and right lane point entities are used to find the equations of the best fit curves of both the lanes using the polyfit() method of numpy. The result of the polyfit method gives the coefficients of a second order equation that is the best fit curve for the left and right lanes. Using these equations of the left and right lanes, we plot the polyfit equations over all the points of a column of an image. This will give the lines in the birds eye view. These equations are also used to find the curvature of the lanes, and hence predicting the turn. The equation to find the curvature of a second order curve $f(x) = ax^2 + bx + c$ can be written as equation 8.

$$Curvature = \frac{2a}{(1 + (2ax + b)^2)^{\frac{3}{2}}} \quad (8)$$

Setting a threshold of greater than 8×10^10 for a rightwards leaning curve and lesser than -7×10^10 for the leftwards

leaning curve. Else the lane is predicted to flow in the straight direction.

Once the lanes have been found from the polyfit equations, these lane points have to be mapped onto the camera frame by performing inverse perspective transform on all the points of the polyfit lane entities. The projected points of the lane entities are used to fill a polygon using fillPoly() method of openCV and then superimposed on the camera image using the addWeighted() method of openCV. The complete pipeline of the lane detection is shown in figure 4.

*3) Lane Detection using Color Threshold - **Dataset 2**:* The pipeline for dataset 2 is similar to the pipeline used for dataset 1. The difference being that the lane in the dataset 2 has color properties and cannot be extracted using simple thresholding operation. Color thresholding has to be performed, for which we convert the image into HSV space and give the lower and upper threshold of both yellow and white colors. The lower and upper threshold for yellow was found to be (0,28,146), (40,255,255) and the lower and upper threshold for white color was found to be (0,0,178), (255,255,255).Once the thresholding of yellow and white colors is done, the masks of both the colors are added and histograms are found along the columns. The rest of the pipeline is same as the pipeline used for dataset1 as shown in figure 5.

IV. Results

Lane detection output on dataset 1 video link - https://github.com/sandeep-kota/Advanced-Lane-Detection/blob/master/Data1.mp4

Lane detection output on dataset 2 video link - https://github.com/sandeep-kota/Advanced-Lane-Detection/blob/master/Data2.mp4

Video Enhancement video link - https://github.com/sandeep-kota/Advanced-Lane-Detection/blob/master/Improved_Frames_Prob1.mp4

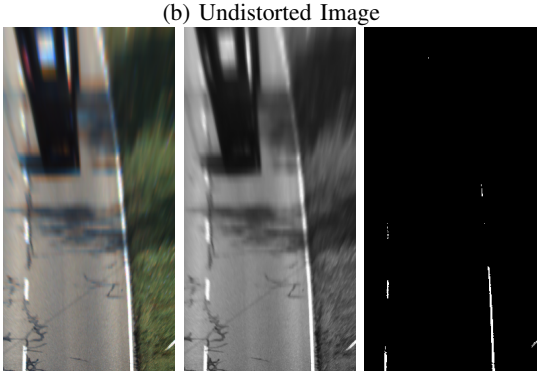
V. Conclusion

For problem 1, we can conclude from the given experiments that enhancing night images purely using software techniques without using artificial intelligence algorithms, introduces significant amount of noise to the image scene. The trade-off between the introduced noise and the usable information depends on the how we process the image further. Figure 1c shows us that if we are using the enhanced image, the vehicle headlight information is absolutely unusable as the patch of white light become too big to contribute towards any useful information. But looking at figure 1b and 1d it is evident that the sign boards are much more usable after the enhancement. Also, the figure 1a shows some extra information about the lanes in the enhanced image.

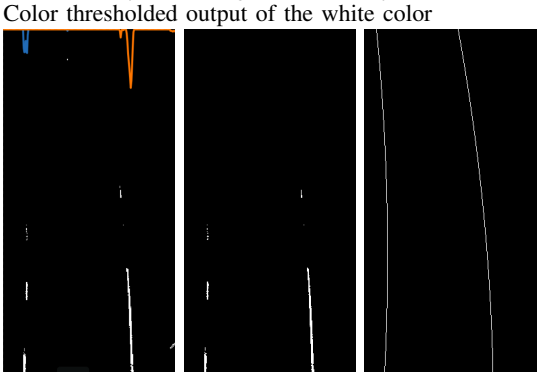
Although conventional computer vision techniques perform relatively well for a given scene, it cannot be generalized for any given scene. A major reason for this is that the threshold of lane entities change with every scene and the thresholds have to be calibrated whenever the lighting conditions change. This phenomenon is evident in the second dataset where the car passes under a bridge and the lanes in the image do not satisfy the defined threshold as the lighting condition changes under the bridge. Hence due to this the defined lane detection algorithms misbehaves and gives incorrect detection. This problem can easily be handled using a tracking algorithm. Another problem with this method of detecting lanes is erroneous detection of the lane color in the lane neighborhood lead to incorrect lane estimation from the histogram peaks. An example of this phenomenon can be seen in the dataset 1 where the algorithm gives incorrect predictions whenever a white object passes nearby the left or right lanes. These errors in lane detection can be mostly solved using data-based techniques by training deep-learning techniques for lane detection.

(a) Original Image

(b) Undistorted Image



(c) *Left:* Bird's eye view after perspective transform. *Middle:* Grayscale image of bird's eye view. *Right:* Color thresholded output of the white color



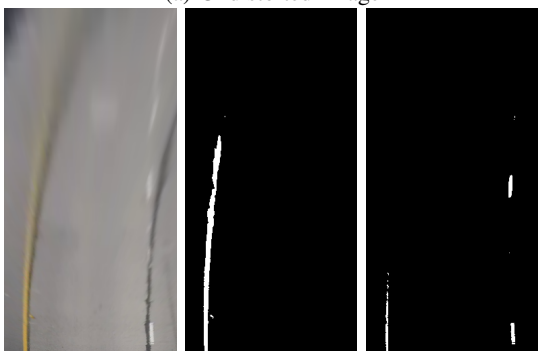
(d) *Left:* The histogram of the columns is shown in blue and orange. *Middle:* Mask after removing noisy threshold along the lane boundaries. *Right:* The polyfit curve of the left and right lanes

(e) Lane output in camera frame

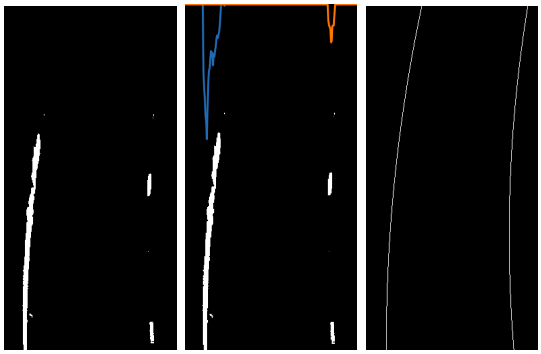
(f) Detected lane outputs with predicted road curvature

Fig. 4: Lane Detection pipeline for Dataset 1

(a) Undistorted Image

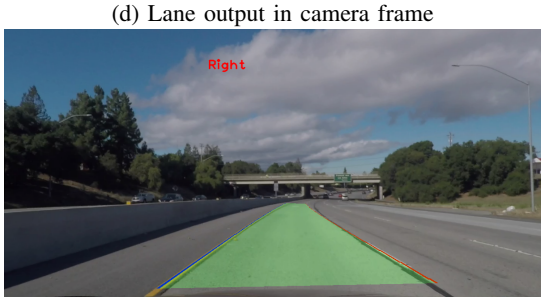


(b) *Left:* Bird's eye view after perspective transform. *Middle:* HSV masked output of the yellow color left lane. *Right:* HSV masked output of the white color right lane.



(c) *Left:* The combined mask of left and right lanes. *Middle:* Histogram of the points represented in blue and orange plots. *Right:* The polyfit curve of the left and right lanes

(d) Lane output in camera frame



(e) Detected lane outputs with predicted road curvature

Fig. 5: Lane Detection pipeline for Dataset 2