ENPM673: Perception for Autonomous Robots

Project 2: Lane Detection and Tracking

Pruthvi Sanghavi UID: 116951005
Naman Gupta UID: 116949330
Amoghavarsha Prasanna UID: 116952910

Submitted **Date:**11 March 2020

Contents

1. Problem 1:

Given a video recording of a video at night, there is a need to apply pre-processing techniques on the video in order to improve the quality of the video sequence before it can be used for any computer vision pipeline.

There were multiple approaches employed and used to improving the quality of the video by increasing the contrast. Initially, the method of adjusting the brightness and contrast of the image by varying the α(contrast) and β(brightness of the image). This approach was definitely increased contrast for the entirety of the image.

Figure 1: Processed Output:Adjusting α and β

The second approach was to improve the contrast of the image through histogram equalization. The builtin function cv2.equalizehist() yielded result with very grainy and noise output. The next approach was to apply histogram equalization through contrast limited adaptive histogram equalization(CLAHE). CLAHE is a more focused and adaptive approach, where several histograms are computed based on different sections of the image and redistributing the values to improve contrast.In addition, to achieve better results, the video that was read frame by frame was converted into HSV and CLAHE was applied on each channel and then converted back into BGR to get the result as seen below. Please refer code for full implementation. [1]

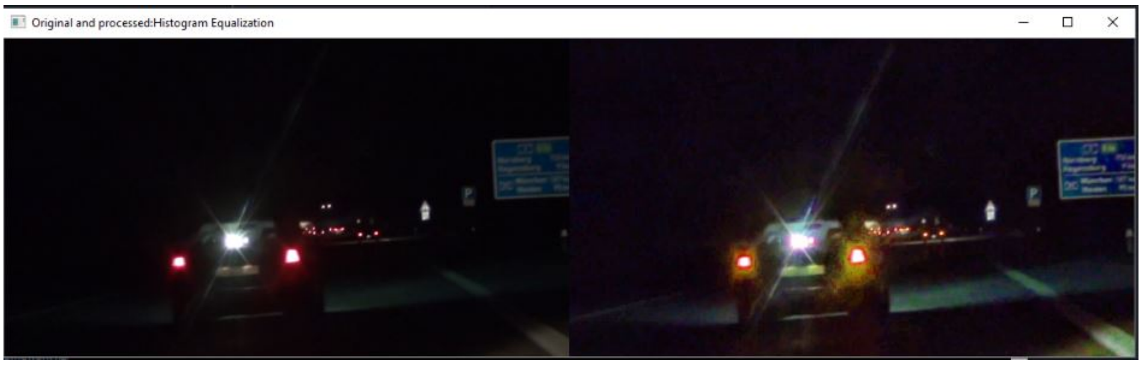

Figure 2: Processed Output: Histogram Equalization

2. Problem 2:

The aim of the problem is develop a pipeline for Lane Detection and Tracking for Autonomous road vehicles.

2.1 Input Preparation:

First, pre-processing operations must be carried out on the input, in order to ensure that the lane detection algorithm yields productive results/output. The image needs to be undistorted, using the K(camera matrix) and distortion parameters provided, the camera parameters are further refined and the Region of Interest(ROI) are extracted. This is done using a custom function in the code, where the built in function cv2.getOptimalNewCameraMatrix is used to refine parameters.

2.2 Detection of lanes:

This implementation of the lane detection uses histogram of lane pixels. A custom function is defined to for warping and to find Homography. The basic idea behind the function is that a rectangular contour region is being defined and then the function cv2.getperspective() is used in between the defined contour region and the ROI, this basically finds the homography.Next the function cv2.warpperspective() for warping.[1]

2.3 Homography:

Homography is basically the transformation between two frames. In this implementation opencv function cv2.getperspective() is used to find homography between the image frame and the points interest.

$$\begin{bmatrix} x_1^{(w)} & y_1^{(w)} & 1 & 0 & 0 & 0 & -x_1^{(c)}x_1^{(w)} & -x_1^{(c)}y_1^{(w)} & -x_1^{(c)} \\ 0 & 0 & 0 & x_1^{(w)} & y_1^{(w)} & 1 & -y_1^{(c)}x_1^{(w)} & -y_1^{(c)}y_1^{(w)} & -y_1^{(c)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\ x_4^{(w)} & y_4^{(w)} & 1 & 0 & 0 & 0 & -x_4^{(c)}x_4^{(w)} & -x_4^{(c)}y_4^{(w)} & -x_4^{(c)} \\ 0 & 0 & 0 & x_4^{(w)} & y_4^{(w)} & 1 & -y_4^{(c)}x_4^{(w)} & -y_4^{(c)}y_4^{(w)} & -y_4^{(c)} \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Figure 3: general form of homography

here, $x_i^{(w)}$,$y_i^{(w)}$ and $x_i^{(c)}$,$y_i^{(c)}$ are the points in the world co-ordinate system and the image co-ordinate system respectively, between which homography needs to be applied.

Since the A matrix is not symmetric, the solution needs to be found using least squares estimation

Sequence of execution: The order of sequence of operation of algorithm is as follows: The input video/image is read, its then undistorted using the custom function that both undisorts and returns the ROI. Then, the input is denoised and cropped, finally the custom warped function is called.

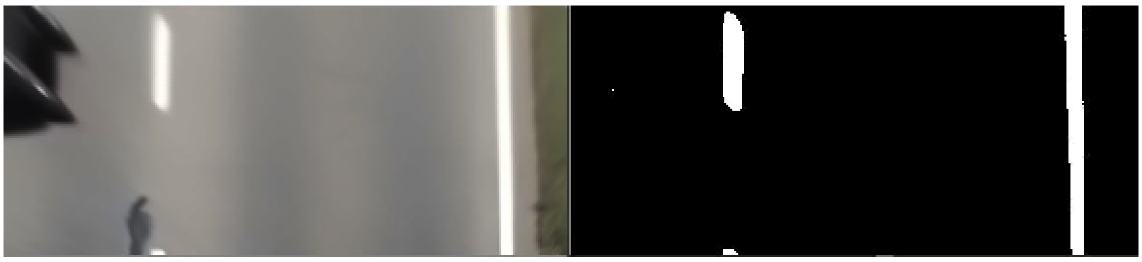

Figure 4: Input Image and output with lane detection

Finally, once the warping is done, the histogram is found through color separation and the pixel sum is determined, the plotted results can be seen below.

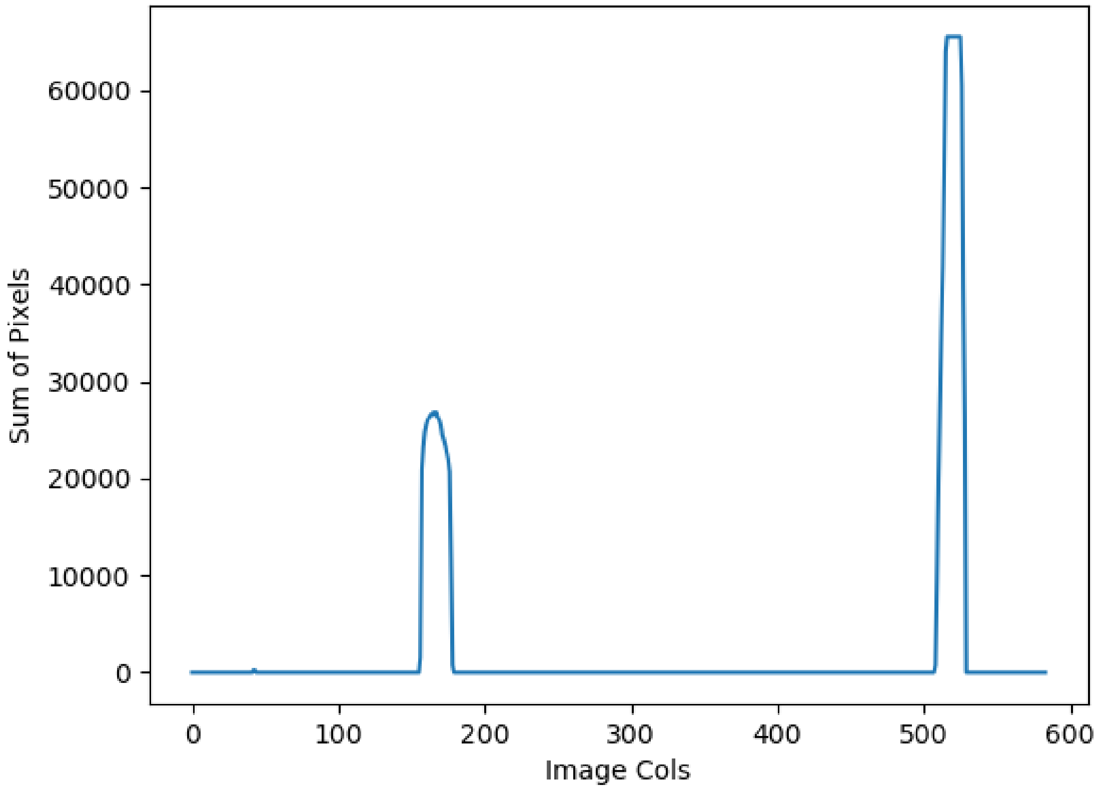

Figure 5: Image columns vs sum of pixels

2.4 Hough transforms

Hough Transforms is an extraction technique used to detect features in images. Hough Transforms can be used to detect lines, circles and ellipses. Consider a line y=mx+c, Hough transform represents the slope(m) and the intercept(b) in the parameter space. However, this representation for vertical lines can result slope(m) to tend to infinity. As a result, polar coordinates are used and a line can be represented as: $r = x\cos(\theta) + y\sin(\theta)$

where r is the distance from the origin to the closest point on the straight line, and θ is the angle between

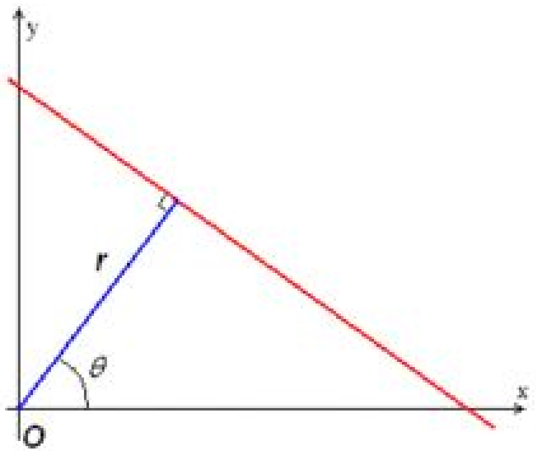

Figure 6:

the x axis and the line connecting the origin with that closest point. Hough Transform is a voting scheme, where, points vote for a set of parameters describing a line or curve.
However, it has not been used for this implementation of lane detection, since its difficult when working with curved roads.

2.5 Refine Lane Detection and Prediction:

For this part of the pipeline, the pixel sum and the pixel locations for both the white and yellow lanes from the 2 videos are extracted. The pixel values for both lanes are stored in lists. Next, the curve fitting is done using the pixel locations using the builtin function cv2.polylines().[1] Then, using inverse homography from a custom function, the curves are then superimposed back in the video.

Prediction is achieved by mapping the difference between the top most pixel of the current frame and the top most pixel of the previous frame.

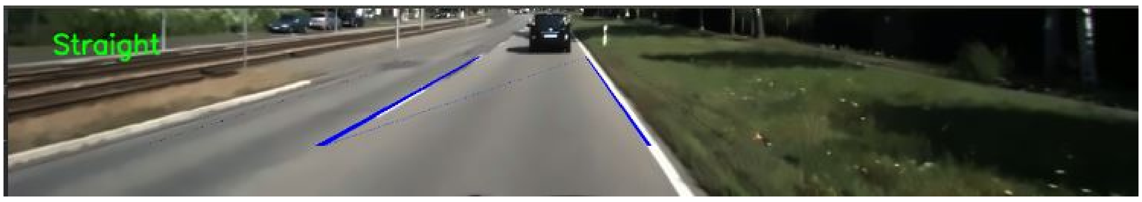

Figure 7: Prediction for data 1

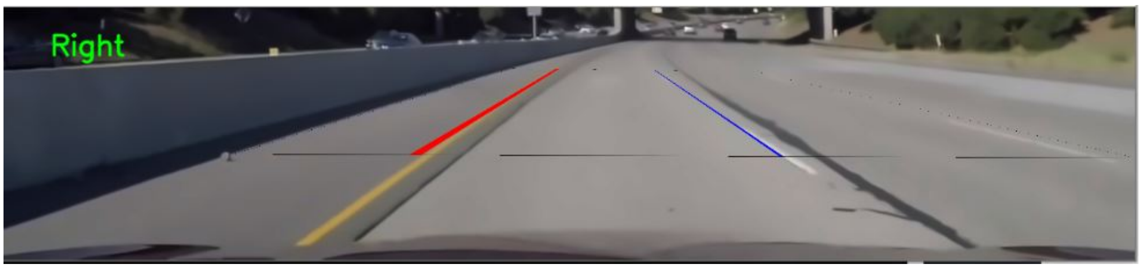

Figure 8: Prediction for data 2

3. Bibliography

References

[1] G. Bradski, “The OpenCV Library,” *Dr. Dobb’s Journal of Software Tools*, 2000.