ENPM673 Perception for Autonomous Robots

PROJECT 2

Lane Line Detection

Aditya Khopkar (116911627)
Nalin Das (116698290)
Nidhi Bhojak (116787529)

Contents

List of Figures

1 Introduction

The first problem is concerned about applying image enhancement techniques on the input video file. It can be observed that the given video has very poor lighting conditions. Thus, various methods were studied and applied in-order to improve the video quality. The second problem of this project aims to mimic the Lane Departure warning System used in autonomous cars.Lane Detection is one of the most important attribute of self driving cars. This helps to check whether the car is on the desired lane and correct the position if it is not. The project overall covers the major concepts of Advanced Image Processing.

The output video files can be accessed from this link:
https://drive.google.com/drive/folders/1HOUaz0S4cs5UAjKaBKgdB87HFfdcVYHr

2 Problem 1

The goal here was to improve the Visual appearance of the given Video. The video file shows a scene from a highway during the night. The input video sequence is dark and not much details can be gathered from it. Hence, first we tried to apply histogram equalization which is generally used to enhance the contrast and brightness in an image. The non-uniform pixel values are stretched over whole image uniformly to correct the contrast. The video was converted to grayscale and then the equalization was applied but the output was observed to be noisy and was showing aberrant equalization behaviour. All the channels for individual BGR channels were also separated and tried for the equalization process. But the results were similar and no subsequent improvement was observed in the video quality. Therefore, we then moved to "Gamma Correction".

Each pixel in an image has a brightness level, called luminance. Now, every image capturing device cannot capture the luminance value perfectly. Hence the images are encoded for their gamma values. This method is also know as power law transformation. It manipulates the brightness values of the pixels as well as their red to blue to green ratio. Firstly, we scale our image pixel values from [0,255] to [0,1], where 1 is black and zero is white. From that, we get gamma corrected output image using the formula $O = I^{1/G}$. Gamma values ¡ 1 shifts the image to the darker spectrum and values¿1 shifts the image to the brighter side of the spectrum. Here, we used gamma =2 for the corrected output image. The pipeline of solving this problem is described as follows:

1. Initialize a gamma value
2. Convert each pixel of the image from the range of (0-255) to (0-1).
3. Apply gamma correction formula as described above.
4. Convert the output image back to the original pixel value range of 0 to 255.
5. To achieve this manipulation, implement a lookup table which matches the pixel intensity values of the input image to that of the corrected output lookup table values obtained by performing tasks from 1-4.

3 Problem 2

Below is the pipeline we followed to solve this problem:

1. Undistorting the image to get rid of the barrel effect/fish- eye effect due to the camera lens.
2. Removing the noise present in the image
3. Getting the Bird's eye view of the Region of interest
4. Pre-processing the image to detect only the lanes by changing the color space from BGR to HLS
5. Detecting Lanes using histogram peak values.
6. Fitting a curve over the lanes using the Polyfit() function and using the sliding window operation
7. Dewarping and Projecting the lanes detected on the original frame.
8. Detecting the Curvature to predict the turn.

3.1 Getting the Input Ready

Because of the physical properties of the camera lens, the images captured are not perfect. Hence, the first step is to undistort the image because distortion can change the size and shape of the object. There are various distortions like the spherical aberrations, radial distortion that affect the final image. The image distortion can be corrected by knowing the camera intrinsic parameters and the distortion coefficient of the lens. OpenCV allows quick and simple implementation of this by allowing us to use the function undistort(). Refer the figure below.

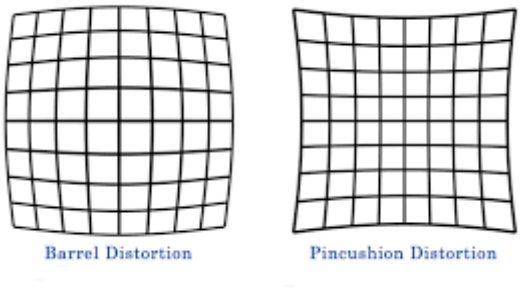

Figure 1: Distortion effects

Now, we are only interested in our region of interest and hence we define four points on the input image and get the bird's eye view of the lane image. For this we use the getPerspectiveTransform() then use warpPerspective() to get the warped image.

Figure 2: Undistorted Input Image

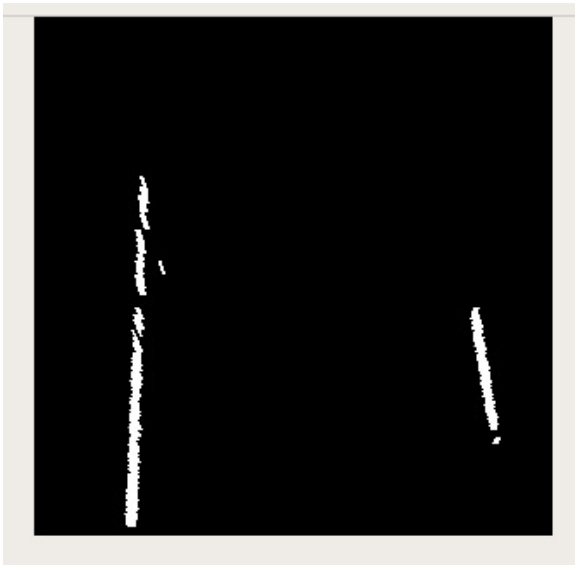

Figure 3: Warped image

3.2 Color Space Transformation

To estimate the lanes on the road, we do not require any other details apart from lanes itself. The road lanes from the image are white and yellow and hence we can mask out everything else in the image except for the white and yellow lanes. The color format we chose to work with is HSL which takes into consideration the Hue, Saturation and the Lightness values of the pixels. Hue is the representation of the perceived color in the combinations of red, green and blue format. Saturation means how colorful or dull the image is based on the Chromatic Intensity. Lightness is the measure of how close the color is with white color.

We create two masks to detect the yellow and white details. These are then combined to get a final binary image by performing thresholding with yellow and white mask in it. After this conversion the detection of lanes lines is done by merely determining the pixels in the image whose values are not "0"(black).

3.3 Detecting Lanes and Curve Fitting

After getting the binary image with only lane lines in it, we need to extract the details about the respective lane pixels. To differentiate between the left lane and right lane, we use the histogram."Histogram" is the graphical representation of the tonal distribution in the digital image. It tells us how the data is distributed within an image.

The left lane and right lane will be identified using the two the peak values of the histogram, which is calculated by summing all the pixel values of the image ranging through 0 to 255. Then, we define two separate regions in the image for left and right lane and perform individual operations on both. For this, we need to traverse along the height of image in small steps.

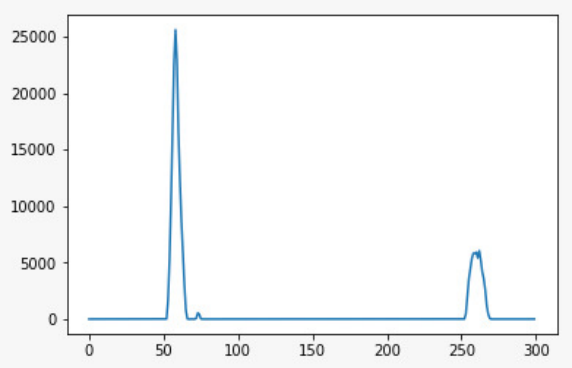

Figure 4: Histogram of the detected lane

3.3.1 APPROACH 1: Slicing the 'lane' image on the basis of white pixel occurences

In this approach, we planned to take only the desired portion of the image to be cropped out from the main image acting as a sort of window. And then perform further operations on it. But the difficulty with this was that after cropping the image, the pixel co-ordinates we were getting for the white lane lines were on the basis of the cropped image and not on the basis of the main image. Thus, the co-ordinates of white pixels needed to be mapped separately from the main image. Hence, we moved on to another approach.

3.3.2 APPROACH 2: Implementing a sliding window to track the lane line candidates

In this approach, we created a separate window on both the lanes keeping a certain margin from the lane center. These windows can be visualised to be smaller frame sections with only one lane line it. The number of windows are kept to be fixed i.e 10. Once we have two separate regions, the white pixels can be extracted out from individual windows. The window is traversed through the whole image height and the indices of the white pixels are appended into a separate list using the nonzero function. The windows are shifted to left or right side depending on the mean of the pixel value indices found in the previous window.This will help us to take into account the steep turns that vehicle might encounter and better trace the lane line candidates.

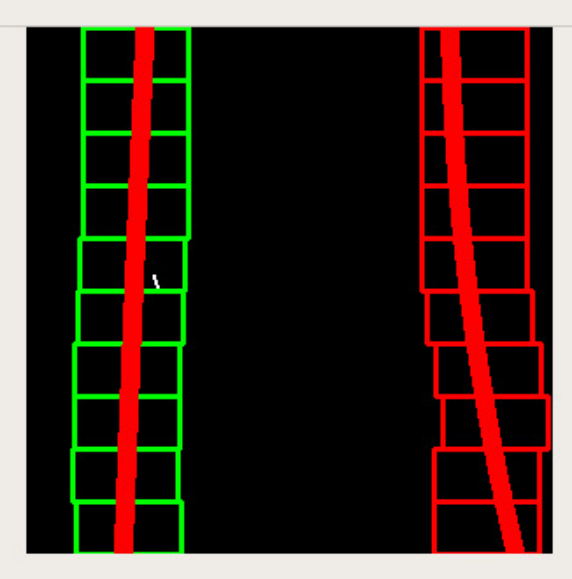

Figure 5: Implementing Sliding Windows

3.4 Fitting Curves on the lane

After getting the white pixel indices for both the lanes in separate lists, we need to get the polynomial function that fits the pixel locations. The polynomial can be found using the PolyFit() function available in numpy-python that returns the coefficient values of the equation i.e (a,b,c) or in other words, the model parameters. The degree of the polynomial chosen is 2 so as to take into consideration the curves during turning. Then after getting the curves of the lane, we fill that detected region with a particular color. This is done using the fillPoly() function. The detected lane co-ordinates are given to this function in correct order so that it can be filled with the chosen color.

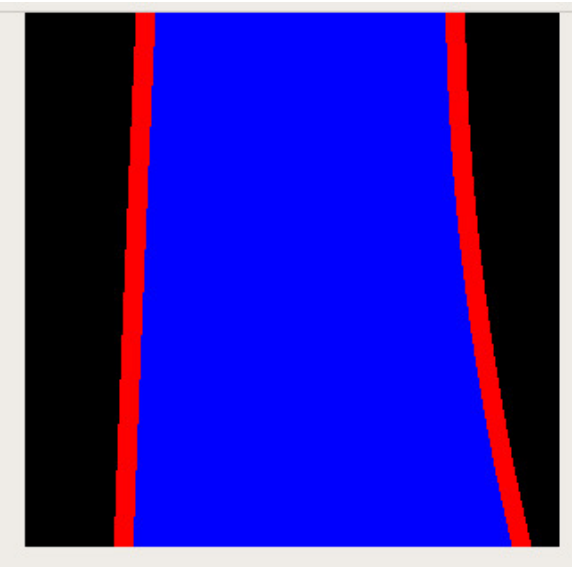

Figure 6: Coloring the Detected region

3.5 Dewarping and Projection on the main Video

After getting the lane region colored in the warped image, we apply inverse warp to transfer the detected lane region onto the camera view. Further to project this on the main image, we use the addWeighted() function available in Python. The function takes two images and assigns weight value to both. Based on these weights the two images are merged by manipulating their opacity. The result we get is the projection of the detected blue region on the main highway lane.

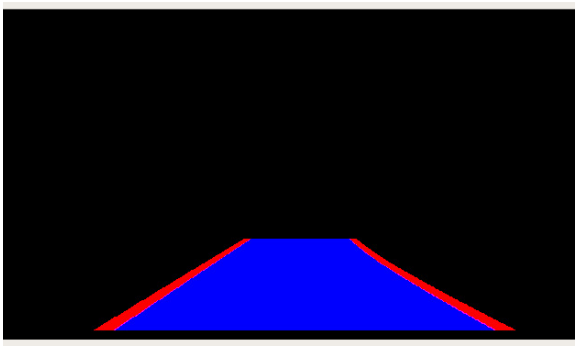

Figure 7: Dewarping the image

3.6 Turn Prediction

The next step was to predict the turns. This can be done by computing the radius of curvature of the road. The radius of curvature was computed by implementing the formula:

$$R_{curve} = \frac{[1+(\frac{dx}{dy})^2]^{3/2}}{|\frac{d^2x}{dy^2}|}$$

Figure 8: Radius of Curvature

The polynomial fits for the left and right lane were used. Correspondingly mean was calculated and then we compared the slopes. If the value is greater than zero, we display the text "Turn Right" and if the value of the slope is less than zero, we display the text "Turn Left".

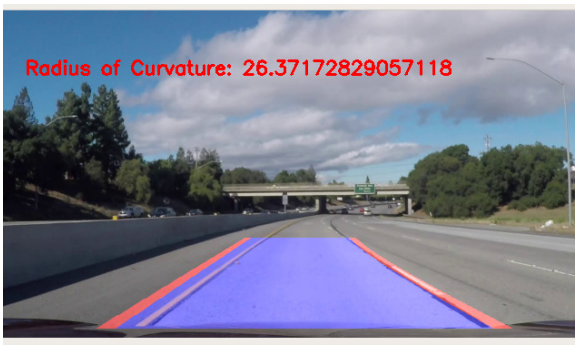

Figure 9: Projecting the detected region

4 Understanding of Homography

Any two images of the same planar surface in space are related by homography. Homography captures the transformation between the two camera images in a matrix form called "Homography matrix". First, we take four points from the first image and another set of four points from the second image. The number of points can be more. Now, homography matrix maps the points from first image to the corresponding points in the second image. This process gives us the relation of rotations and translation between the two camera frames.

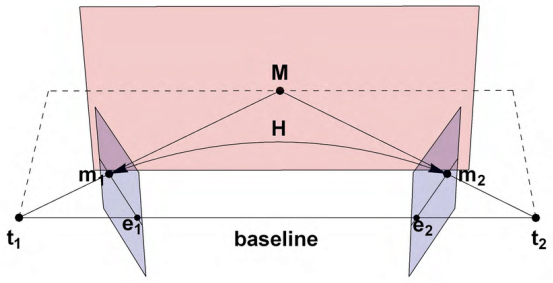

Figure 10: Concept of homography

5 Understanding of Hough Lines

The Hough transform is a feature extraction technique used in computer vision and image processing. It can be used when we want to separate out some specific features of a particular shape in an image. It requires desired parameters to be specified in a parametric form and identify the curve parameters that best fit the points.

All lines can be represented in the polar form of the equation i.e $r = x * cos\theta + ysin\theta$ where the value of θ varies from 0 to 180. The Hough space for lines has therefore these two dimensions r and θ and a line is represented by a single point, corresponding to a unique set of parameters.
An important concept of Hough transform is the mapping of single points. A point is mapped to all lines, that can pass through that point. This gives a sinusoidal like line in the hough space.

The Algorithm for detecting straight lines using Hough Lines can be broken down to the following steps: Detection of the edges using any edge detector(Canny, Sobel), Mapping of the edge points to the Hough Space and storing them, Imposing constraints so as to yield line of infinite length and finally conversion of infinite lines to finite lines. The finite lines can superimposed back on the original image.

6 Conclusion

During the implementation of this project following observations were encountered:

- The road lanes are not of constant brightness level and hence distortions in lane projection was observed.
- The concept of traversing a window over the image height was important to encounter the curves of the lane.
- When the car goes under the bridge in the challenge video the projections become aberrant because of the sudden change in the brightness level of pixels.
- There were some issues with the video writer used that changed the lane projection to some other color than desired.
- A sufficient understanding about the colorspaces to be used is necessary as it forms the basis of any image processing application. While we used HSL colorspace in this project, other colorspaces such LAB,LUV etc., could also be explored.
- The accuracy of the lane line candidate detection depends on the colorspace taken into consideration.
- In the larger blank spaces, when the lane lines would not be captured, We implemented an algorithm to keep track of the previous polyfits and be rather used than the empty polyfits as a result of empty candidate detection.

7 References

- https://www.youtube.com/watch?v=VyLihutdsPk
- https://medium.com/activating-robotic-minds/finding-lane-lines-on-the-road-30cf016a1165