"[ENPM 673] Project-2"

Perception for Autonomous Robots

PROJECT REPORT

Nevil Patel-116897068

Shesh Mali-116831055

Akshay kurhade-116914529

LANE DETECTION

Contents

Abstract:

The project focuses on improving the lighting conditions, detection and tracking of Lanes. The pipeline of the projects describes the Lane Detection after applying filters like Blur, HSV space, Thresholding and many more to it. We used the principles of Homography and Hough Transforms. Lane Deciding was done using Histograms peaks. After collecting points of lanes, we form lanes lines using the fitting technique. Prediction of turns were done by the shift of lane co-efficient with respect to the window. Finally, we overlaid the detected lines on the original frame.

Problem 1:

A better approach was converting the image colour space from RGB to HSV where H is a representation of Hue, S for saturation and V for Brightness (as mentioned in Wikipedia). Then, Histogram Equalisation Technique was applied to the image after removing the V part of the HSV image. Now gamma correction is done on the image which improved the lighting of the image without distorting the colour scheme but also aggravated noise. Lastly, we add back the V component of the image and convert the improved frame to the original RGB colour space. Finally, we get the frames with improved lighting conditions and the road could be seen better. Images below shows the change in frame before and after processing it.

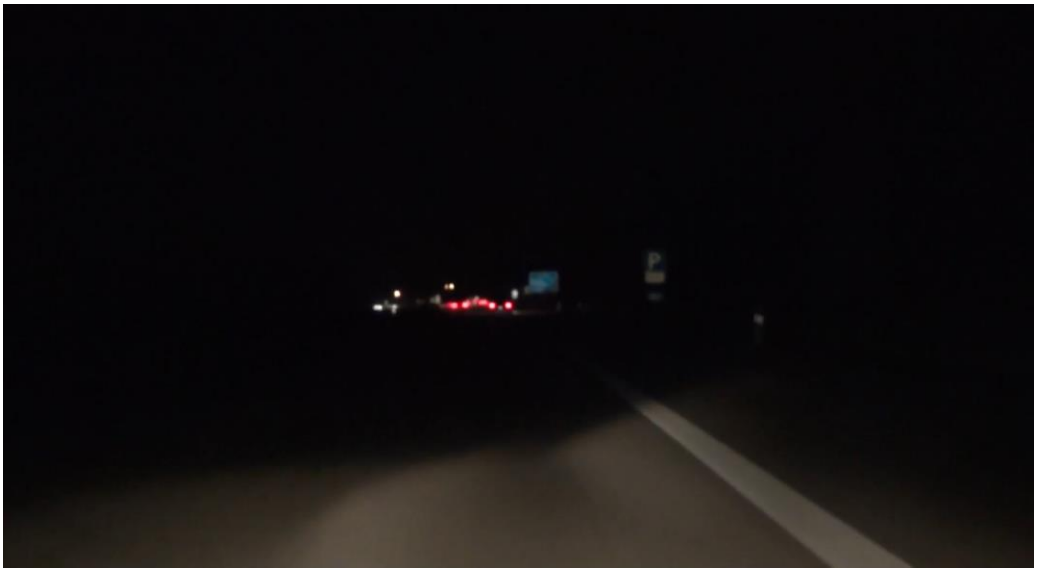


Fig: Poorly lit Video.

Fig: Improved Video with Better Lightning Conditions.

Problem 2:

1. Here we first preprocess the frames. Distortions such as radial distortion, spherical aberration etc. are present because the video is taken from camera. In an distorted image, straight lines seem to curve slightly as we move away from the center of the image. This can cause unwanted results. Hence, we use the provided camera parameters to obtain an undistorted frame. The undistorted frames are then filtered using the Gaussian filter. However, the difference isn't that drastic and noticeable in the frames but further processing is significantly affected by the smoothing.

2. After that the smoothed image is cropped to remove the unwanted region from the frame(i.e sky).

3. Source points have been selected from an image frame where the car appears to move staraight and used for all the frame to get the top-down view of the image with reference to the selected points.

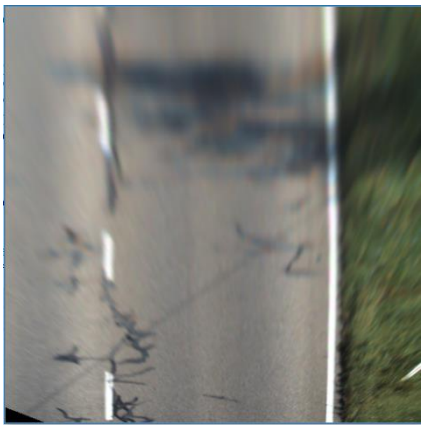


Fig. Top-down view for data1

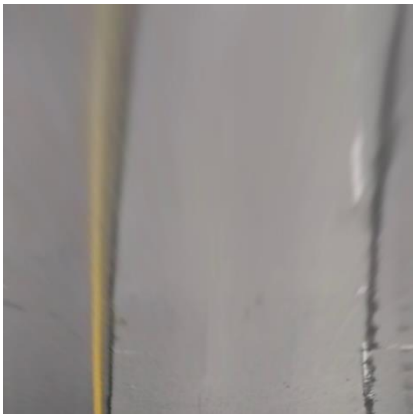
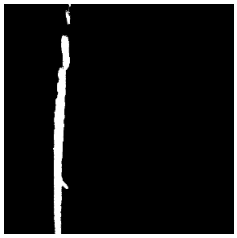
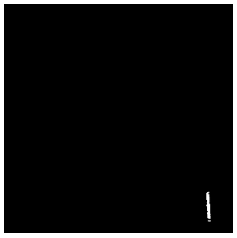
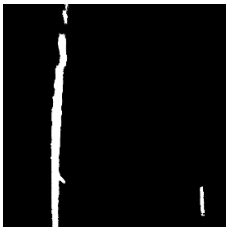


Fig. Top-down view for data2

4. Then we convert the image to HSV color space for better detection of lanes. The main issue we faced was choosing the right color space for the operations. We decided to settle on the HSV color space after trying a lot with HLS. Hence, by thresholding the HSV frame, we filtered out the lanes in the first dataset. For the second dataset, we define two thresholds for the yellow and white lane respectively. We first obtain both the lanes individually and then perform a bitwise OR operation. Sampled images can be seen below.


Yellow Threshold


White Threshold


BitwiseOR'ed output

5. Next, we obtain the histogram of non-zero pixels positions in the binary image. The histogram is taken along the x axis. In the histogram we see two peaks. The left peak indicates presence of non-zero pixels representing the left lane line and the right peak indicates presence of non zero pixel indicating right lane line.

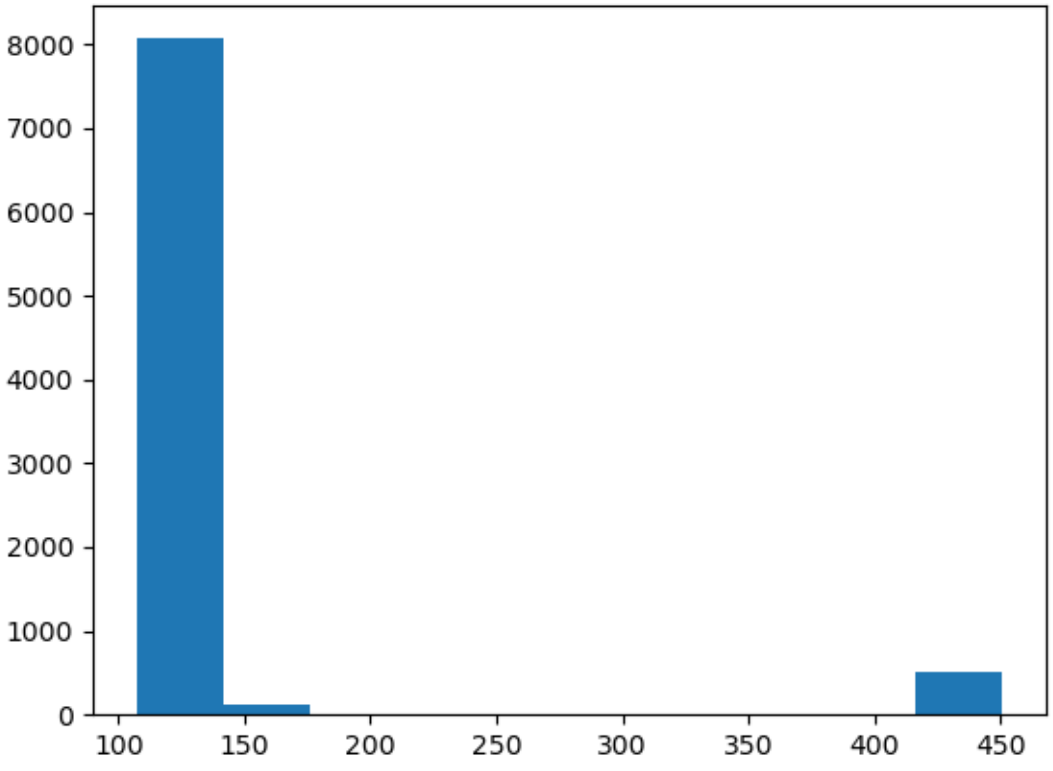
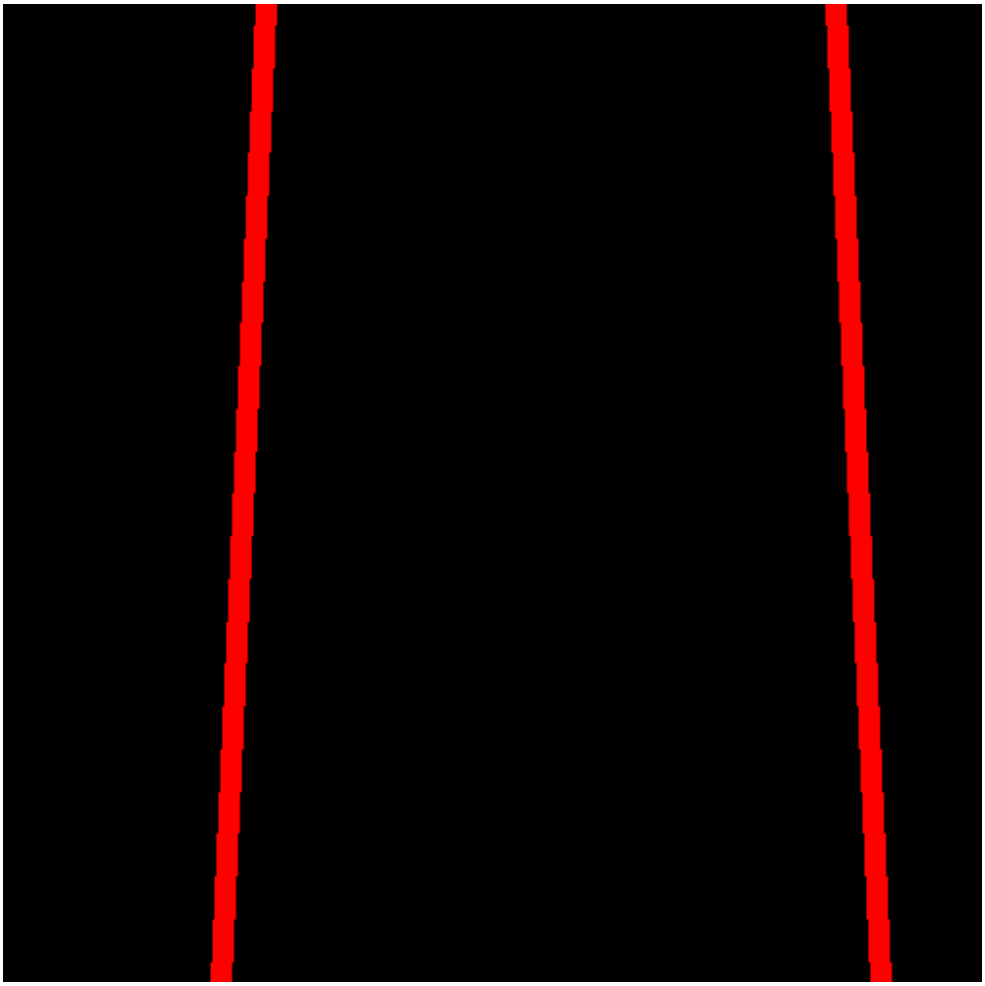


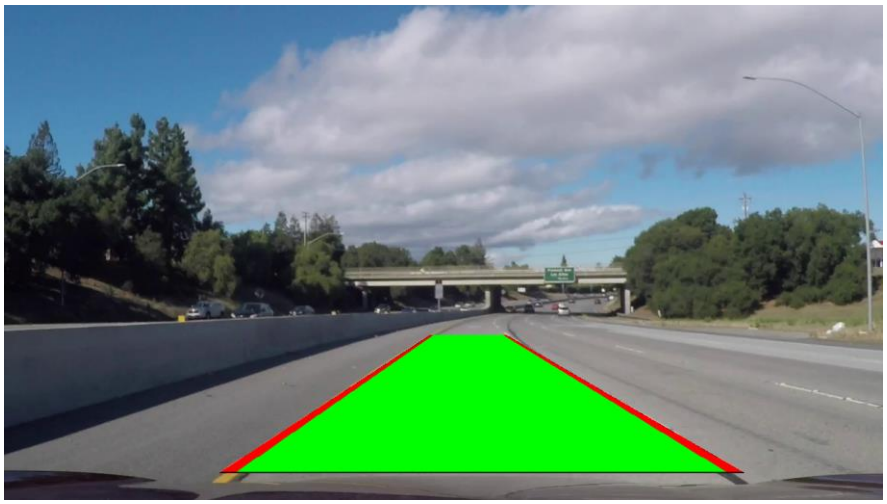
Fig. Histogram of pixels

6. The next step is to decide the lane. If there are no peaks in the histograms then there will be no lane separator markers. Upon checking which side of the median the peak points are, we get three different conditions as lanes on left side, lanes on right side or on both the sides.

7. As we find the lanes, we collect all the points and using the fitting techniques i.e np.Polyfit we can finally form lane lines and draw them.



8. Then we overlay the detected lines on the original lane and fill the lane area with a colour.



9. To predict turns, we find the slope of line in the center of the two lanes using the lane points.

$$\frac{y_{right} - y_{left}}{x_{right} - x_{left}}$$

Problems Faced:

1. The major challenge we faced was in selection of the colour space.
2. The second, most difficulty was choosing the approach for further processing. Whether to go with Hough transform, or to use histogram method and polynomial fitting. After experimenting with both the approaches, we preferred the histogram of lane pixels approach.
3. In the Challenge video, we faced problem in detecting the lane at some points due to change in lighting conditions and signs in the centre of the lane.
4. In either of the videos we faced difficulties in thresholding for desired masks.
5. Due to the noise in lane pixel detection and curve fitting, the slope fluctuates sometimes.

Homography:

What is homography and why it is important in Computer Vision:

When we are dealing with the perception of objects in real world through the computer/camera, we are basically mapping the points from one coordinate system to another coordinate system. Any point/object in 3D world, when projected onto the camera plane, gets mapped to the 2D image plane. So, we can consider this as a basic example of projective geometry. Unlike the Euclidean space, the projective space has ability to project more points (even points at infinity), hence it is widely used in computer vision field.

Homography or homographic transformation is simply the kind of transformation between two image planes in the projective geometry. For this homography transformation, we need to know about the basic concept of homogeneous coordinates. In homogeneous coordinates, we simply add an extra coordinate in the vector representation of the point under consideration. For example, if a world point has coordinates [X^w,Y^w, Z^w], then the representation of its homogeneous coordinates is [X^w,Y^w, Z^w, S^w], where S is the scale factor. In other words, we can write it as: [X^w/S^w, Y^w/S^w, Z^w/S^w, 1].

Suppose we have a point of an object in world coordinate system, denoted as x^w. Now, when this point gets projected onto the image plane, we get the corresponding point in camera coordinate system, denoted as x^c. Then, we can write the following relation between these coordinates as follows:

$x^c = \lambda$ * H * x^w, where H is the homography matrix.

Using these homography, we have transformed the point from one plane(world) to the other(image). Similarly, we can have homography between two image planes as well. In this project for lane detection, we have one plane as the original frame as extracted from the video. In order to have the top-view(birds-eye-view) of that lanes, we need to compute homography between that plane and the desired plane (the dimensions of the new image plane in which we want the view). For this, we simply use the inbuilt function from OpenCV.

i) cv2.getPerspectiveTransform

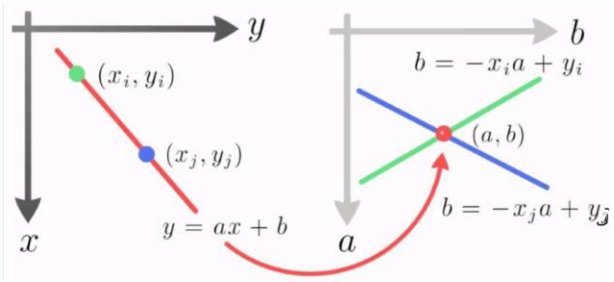
ii) cv2.findHomography

Both these functions yield desired homography transformation between two planes.

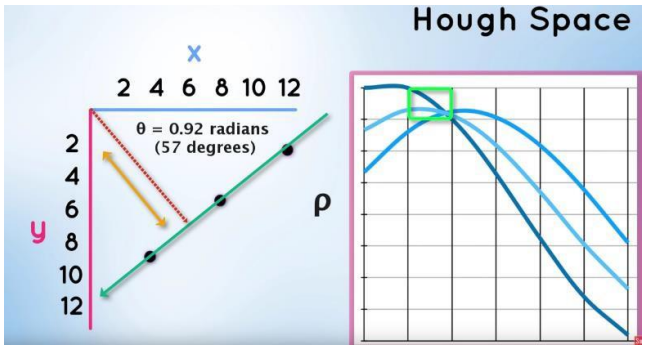
Hough Transform:

Hough transform is a technique which can be used to isolate features of a particular shape within an image. Because it requires that the desired features be specified in some parametric form, the Hough transform is most commonly used for the detection of regular curves such as lines, circles, ellipses, etc. The main advantage of the Hough transform is that the technique is tolerant to gaps in feature boundary descriptions and is relatively unaffected by image noise.

In order to use Hough transform we first apply an edge detector e.g. canny, sobel etc. on an image. Right now, the obtained edges are just a sequence of pixels. The generic idea is to loop through all pixels, and somehow figure out the slope and intercept. Hough transform is basically a mechanism that gives more weightage to pixels that are already in a line. It gives vote to each point on the image and because of the mathematical properties of the transform, this voting allows us to figure out prominent lines in the image.



Line in cartesian coordinate space = to a point in Hough space



Intersection of sinusoid curves

The three plots intersect in one single point, these coordinates are the parameters (θ,r) or the line on which (x0, y0), (x1, y1) and (x2, y2) lay. To detect a line, we find the number of intersections between curves. More curves intersecting means line have more points. Therefore, we find the intersection between curves of every point in the image. Moreover, we define a threshold of the minimum number

of intersections needed to detect a line. If the number of intersections is above the set threshold, then we get a line. For a higher threshold, fewer lines will be detected.

References:

1. https://benchpartner.com/hough-transform-in-image-processing/
2. https://aishack.in/tutorials/hough-transform-basics/
3. https://docs.opencv.org/3.4/d5/daf/tutorial_py_histogram_equalization.html
4. https://docs.opencv.org/3.4/de/d25/imgproc_color_conversions.html#color_convert_rgb_hsv