“[ENPM 673] PROJECT - 2”

ENPM 673 – PERCEPTION FOR AUTONOMOUS ROBOTS

PROJECT 2 REPORT

ADITYA VAISHAMPAYAN -116077354

ISHAN PATEL – 116169103

NAKUL PATEL – 116334877

Instructor: Dr. Cornelia Fermuller

INDEX:

PROJECT PIPELINE:

1. The first step of the entire process is undistorting the image. Since the image is taken by the camera it has various distortions in it. Such as radial distortion, spherical aberration etc. hence we need a calibration matrix to obtain a proper video frame. In an undistorted image, straight lines seem to curve slightly as we move away from the center of the image. This can cause unwanted results in the process further. Hence it is of utmost necessity to undistort the image.

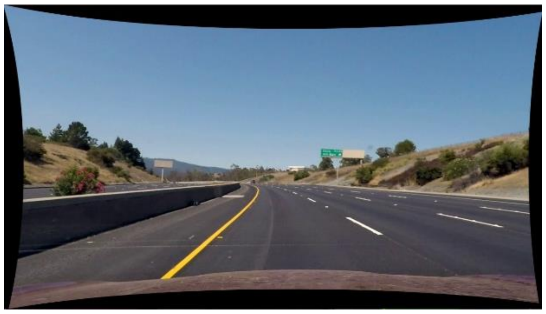

Fig 1: Image after distortion with black regions

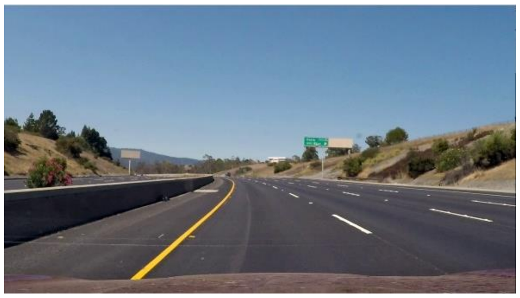

Fig 2: Video frame after removing black unwanted regions

2. The second most important step in the process pipeline is denoising the image. We have used median filter to remove salt and pepper noise and have used gaussian filter as well. However, the difference isn't that drastic and noticeable.

3. Preprocessing is the most important step in the entire pipeline. Also, this is where, we faced most of the challenges. The main issue was choosing the right color space for the operations. Hence, we decided to settle on the HLS color space. We separated the L channel and the S channel. Further we applied Sobel X filter on the L channel. The reason behind using the HLS color space, is the S channel. The lanes have to be brighter and have higher saturation as compared to other pixels in the images and hence, by thresholding the saturation channel, we have filtered out the lanes easily.
4. Taking the perspective transform of the frame in order to obtain the birds eye view of the lane lines. It is essential to obtain the birds eye view in order to obtain accurate results from polynomial fitting through the sliding window technique. The image given below is the binary thresholder image of the perspective transform of the lanes.

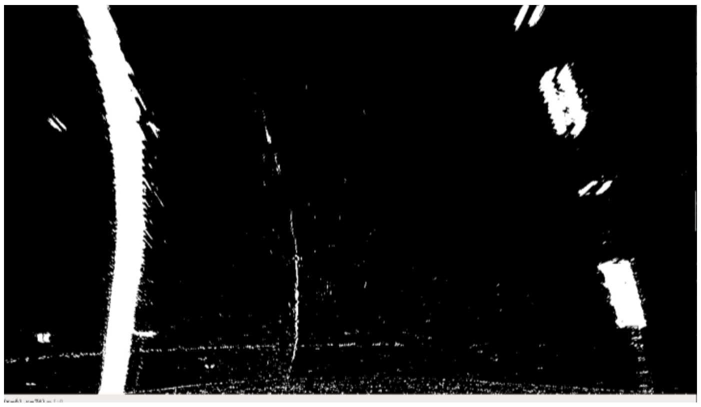

Fig: Warp Perspective of the binary thresholded frame

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.

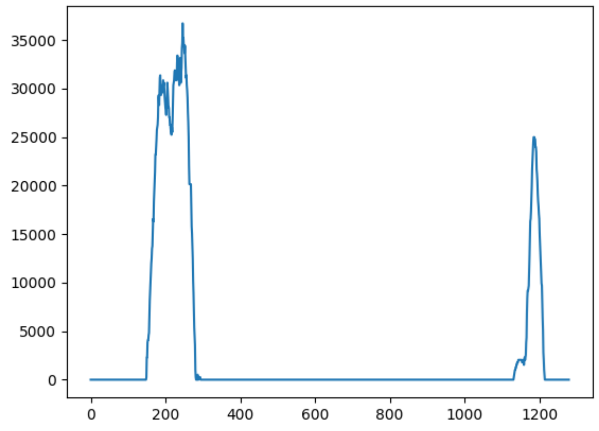

Fig: Histogram of pixel values along X-axis in the image

6. Next, we apply the sliding window algorithm in order to obtain the x, y coordinates of the lane line pixels and to adjust a window around it dynamically. Since the lane curves, the position of the sliding window also changes.

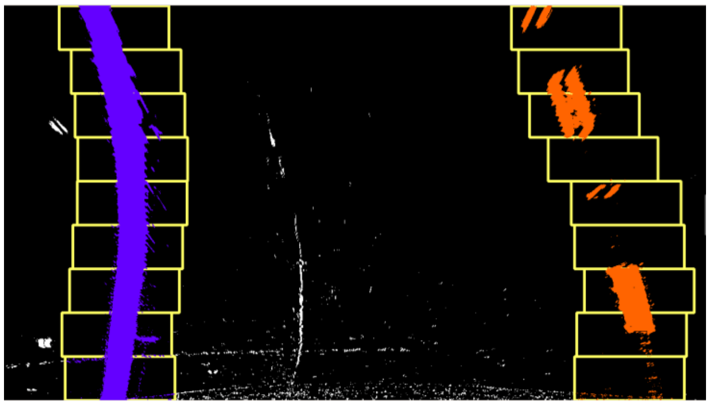
Fig: Sliding window technique applied on the binary image

7. Next, we calculate a polynomial from the obtained left and right lane points. By obtaining this polynomial we obtain the points require to show the green background on the lane. Once the polynomial is obtained, we again apply homography to place, the green polygon back onto the original image frame.
8. The next important step is calculating the radius of curvature and turn prediction. The basic idea behind these is accessing the polynomial values. For turn prediction, we took two points and found the slope of the line between them. If the slope is positive, then we can say that the car will be turning towards the right direction. However, if the slope is negative, the car will be turning in the left direction. The images shown below show turn predictions and lane detection.

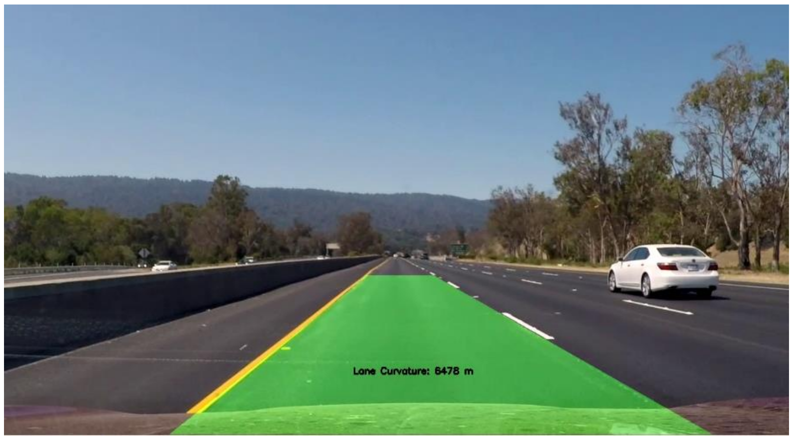

Fig: Detected lane. No turn prediction is shown as car is moving **Straight**

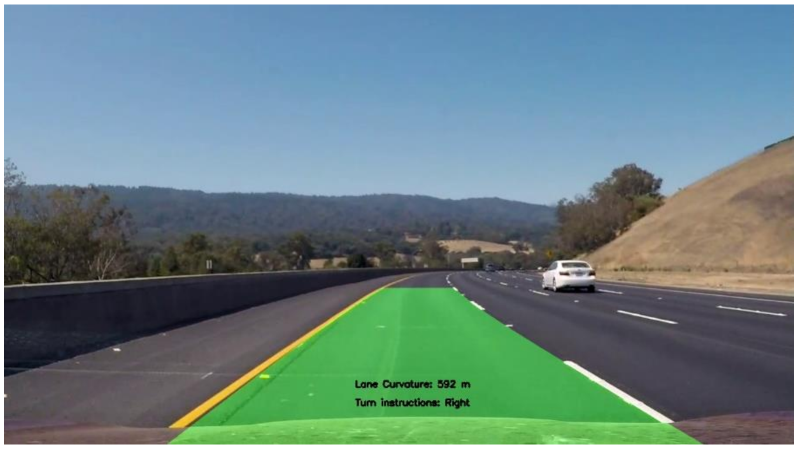

Fig: Detected lane, with turn prediction and radius of curvature. Detected turn **RIGHT**

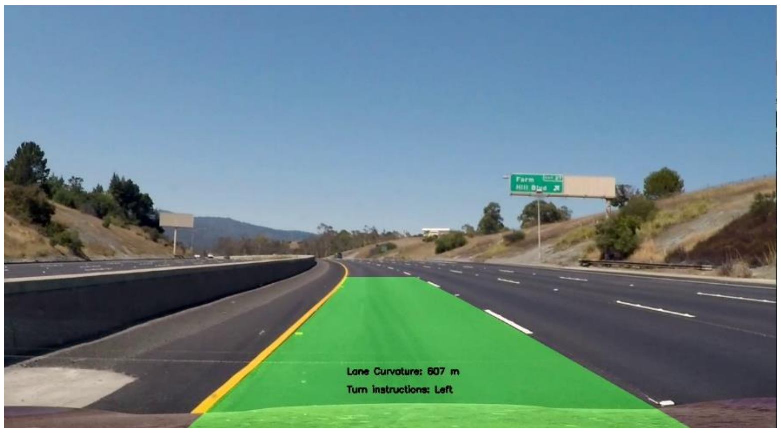

Fig: Detected lane, with turn prediction and radius of curvature. Detected turn **LEFT**

CHALLENGES FACED:

1. The major challenge we faced was in selection of the color space. It was very essential to detect the lane lines properly.
2. The second, most difficult challenge we faced, was choosing the approach for further processing. Whether to go with Hough transform, or to go with histogram method and polynomial fitting. After experimenting with both the approaches, we concluded that histogram of lane pixels approach.
3. Some other challenges we faced were in the sliding window approach, and in the polynomial fitting. The sliding window kept coming back to the center of the image because the histogram couldn’t have the right peak in it. This was corrected by re-choosing the four points taken to obtain the birds eye view.

Above given points are the major challenges that we faced apart from minor syntax errors and numpy array mismatching.

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.

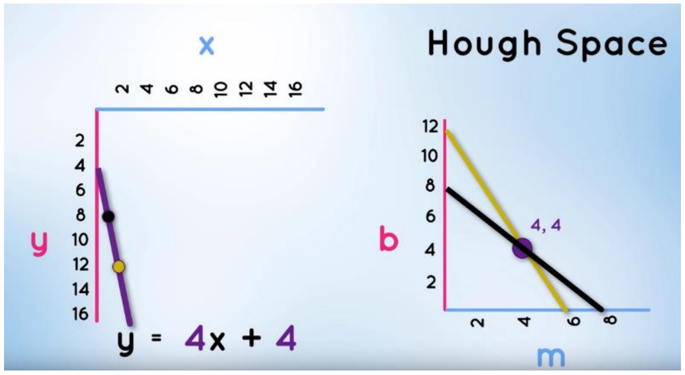

Fig: Line in cartesian coordinate space is equivalent to a point in Hough space

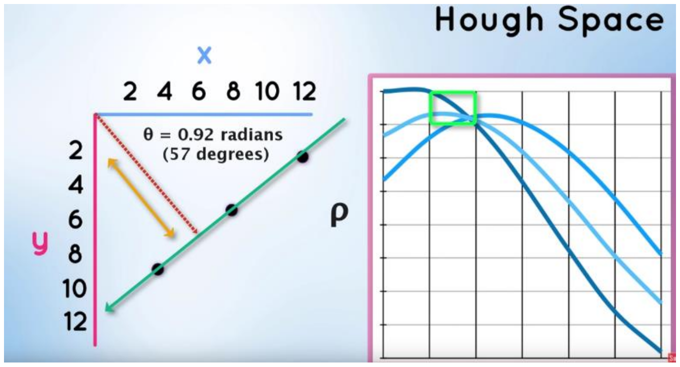

Fig: Intersection of sinusoid curves in a particular bin

Describing an equation of line: $x\cos\theta + y\sin\theta = r$. Here **r** is the length of a normal from the origin to this line and θ is the orientation of **r** with respect to the X-axis.

In an image analysis context, the coordinates of the point(s) of edge segments (x_i, y_i) in the image are known and therefore serve as constants in the parametric line equations, while r and θ are the unknown variables we seek. If we plot the possible (r, θ) values defined by each (x_i, y_i), points in cartesian image space map to curves (*i.e.* sinusoids) in the polar Hough parameter space. This *point-to-curve* transformation is the Hough transformation for straight lines. When viewed in Hough parameter space, points which are collinear in the cartesian image space become readily apparent as they yield curves which **intersect** at a common (r, θ) point.

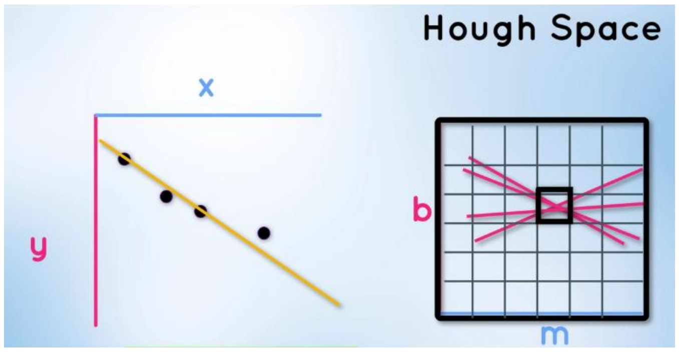

Fig: Intersection of Hough lines in Hough space

The transform is implemented by quantizing the Hough parameter space into finite intervals or bins. As the algorithm runs, each (x_i, y_i) is transformed into a discretized (r, θ) curve and the bins which lie along this curve are incremented. Resulting peaks in the accumulator array represent strong evidence that a corresponding straight line exists in the image.

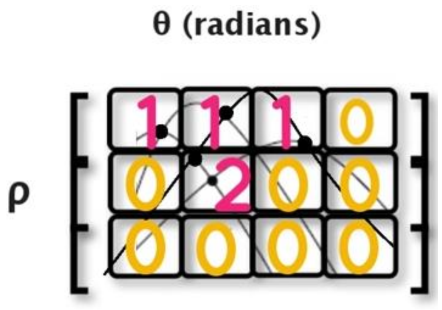

Fig: Assigned weights to bins

REFERENCES:

1. https://www.youtube.com/watch?v=eLTLtUVuuy4
2. http://aishack.in/tutorials/hough-transform-basics/
3. https://en.wikipedia.org/wiki/Canny_edge_detector
4. https://github.com/pkern90/CarND-advancedLaneLines