ENPM 673 – P2

Name: Hrishikesh Tawade
UID: 116078092

Name: Krishna Bhatu
UID: 116304764

Name: Kapil Rawal
UID:116133357

1) Understanding of Homography

- Homography is the transformation of an image from one plane to another. The image is manipulated using the 8 degrees of freedom. We have used homography to calculate the bird's eye view of a specific region of interest where the lane is always detected. We do this so that we can isolate the lane from the image and visualize the lane in its true form, bringing back its parallelism instead of a projection variant and can utilize these properties.
- We have calculated the homography matrix using the getPerspective function in OpenCV between the region of interest points in the image and a set image points decided by us to make the image upright. Then we pass this matrix to the warpPerspective function of OpenCV and then use it to transform the image. Below images show this operation.

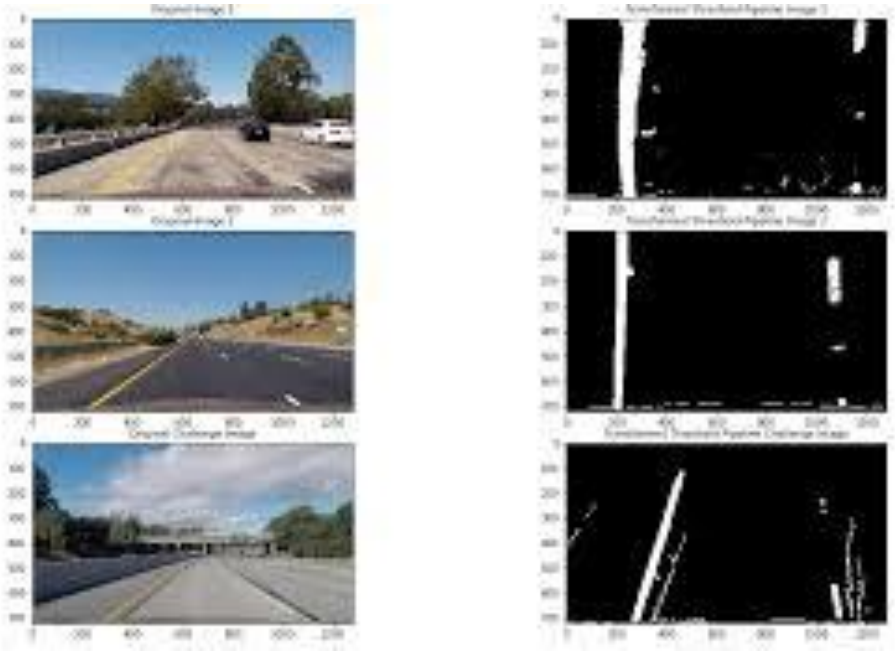

Figure 1. Warp Perspective of Lane

2) Understanding of Hough Lines

- The Hough transform is a feature extraction technique. It is used mostly for detecting lines but can be extended to find circles and ellipses.
- In general, the straight-line y = mx + b can be represented as a point (b, m) in the parameter space. However, vertical lines pose a problem. They would give rise to unbounded values of the slope parameter m.
- Hence in hough transform we consider 'r' which is the distance from the origin to the closest point on the straight line, and theta is the angle between the x-axis and the line connecting the origin with that closest point. This is shown below.

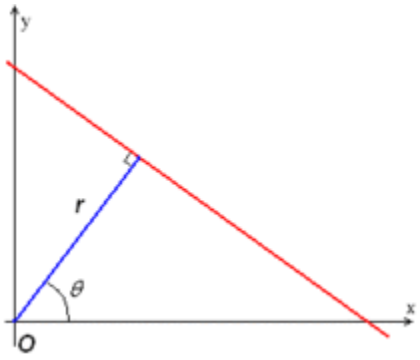

Figure 2. Hesse normal form

Name: Hrishikesh Tawade UID: 116078092
Name: Krishna Bhatu UID: 116304764
Name: Kapil Rawal UID:116133357

- Given a single point in the plane, then the set of all straight lines going through that point corresponds to a sinusoidal curve in the (r,θ) plane, which is unique to that point. A set of two or more points that form a straight line will produce sinusoids which cross at the (r,θ) for that line. Thus, the problem of detecting collinear points can be converted to the problem of finding concurrent curves. Sinusoids corresponding to co-linear points intersect at a unique point. It can be seen in image below.

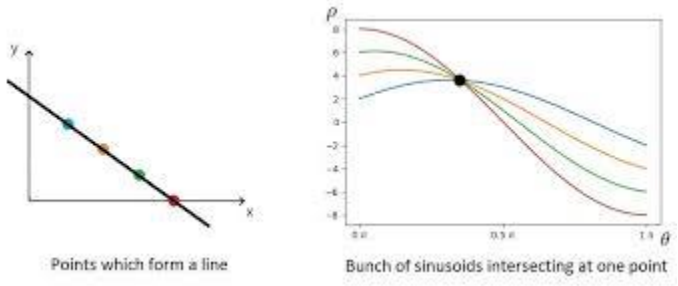

Figure 3. Hough transforms of collinear points

3) Pipeline for Lane Line Detection

The image processing pipeline for the lane line detection algorithm is as follows.

3.1) Warp image and Applying color transform

- To identify the lane properly I experimented with different color spaces. We plotted the image for different channels, HSV color spaces channels, LAB channels and HLS channels.
- Ultimately, we chose to use just the L channel of the HLS color space to isolate white lines and the B channel of the LAB colorspace to isolate yellow lines. We did, however finely tune the threshold for each channel to discard the shadow in the challenge video. Following is there outputs. (Credits: Jeremy Shanon) LINK

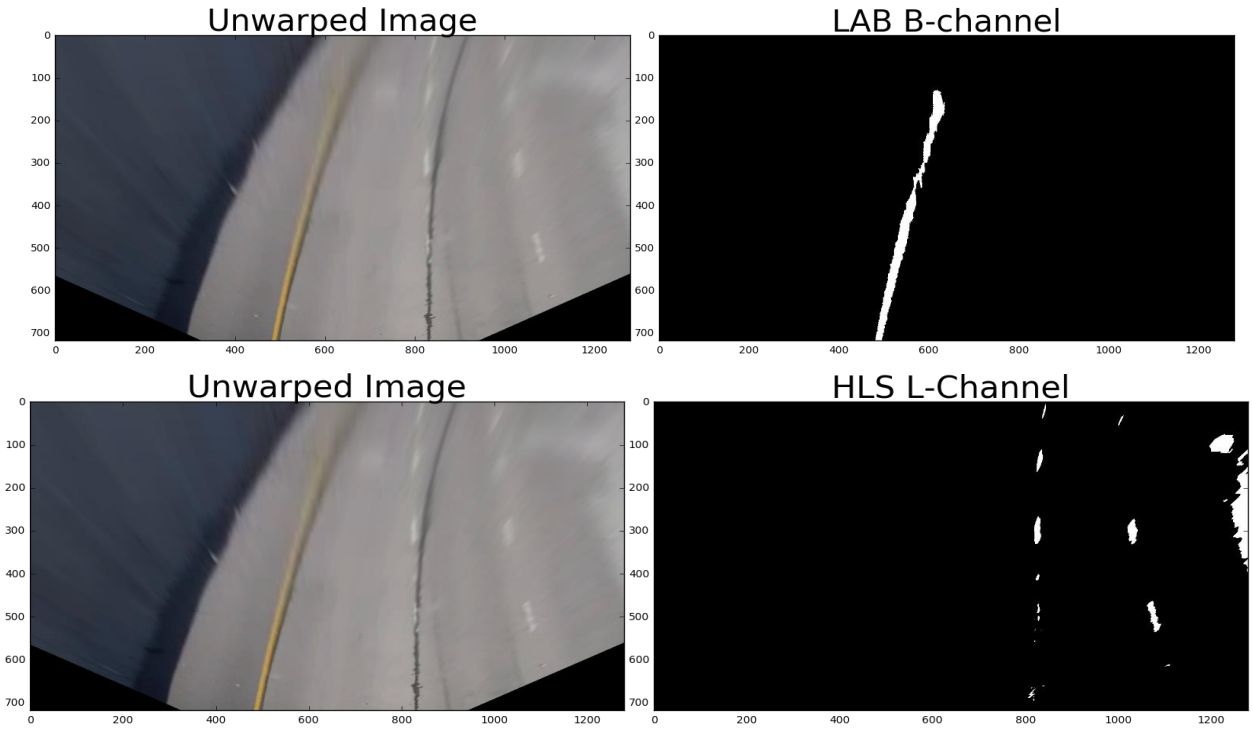

Figure 4. Thresholding of the warped image

Name: Hrishikesh Tawade
UID: 116078092

Name: Krishna Bhatu
UID: 116304764

Name: Kapil Rawal
UID:116133357

- The warp perspective is done for the following points.

```
src = np.float32([(575,464),
                  (707,464),
                  (258,682),
                 (1049,682)])
dst = np.float32([(450,0),
                  (w-450,0),
                  (450,h),
                  (w-450,h)])
```

- We then combine the above obtained images and pass it to the sliding window function

3.2) Using Histogram and Kalman Filter for prediction

- We first calculate the histogram of the lower half of the thresholded image obtained above since we use to calculate the base points of the lane lines.
- We then find then find the midpoint and quarter point of the image and use it to slice the histogram and find the peaks in that using the argmax function of numpy.
- These peaks are passed to the Kalman filter so that we can remove any noisy detection of lane and prevent lane from expanding or contracting suddenly.

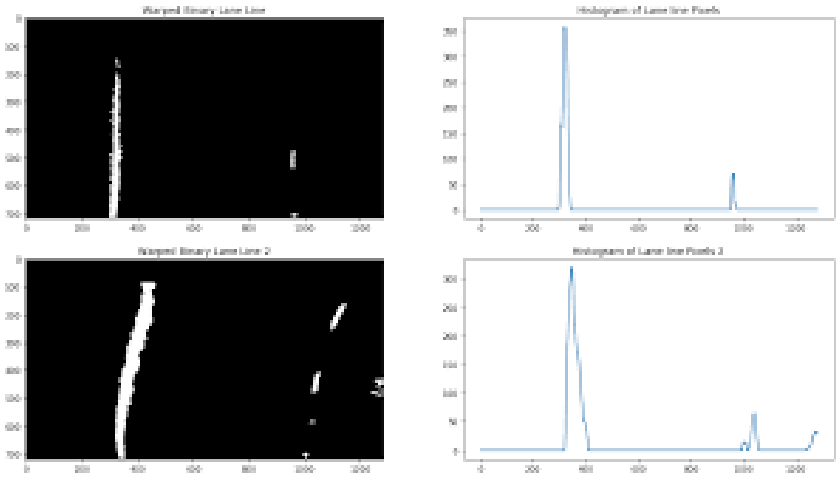

Figure 5. Histograms of the thresholded image

3.3) Using Sliding window polyfit to find all lane pixels

- After we get the two base points from above we use a sliding window of a preset size to appear at that point.
- This window checks the number of nonzero pixels in that window which are basically the number of pixels corresponding to the lane.
- If the number of pixels is greater then 40 then we consider it as a lane window and find the mean of all the point location which will be the x of the next window.
- Since we are using 10 sliding windows, we have divided the height of the image into 10 and those gives us the y-coordinate.
- Thus, the sliding window tracks the lane and gives following output. Finally we collect these pixels and pass it to the next function to calculate the equation of the curve.

Name: Hrishikesh Tawade
UID: 116078092

Name: Krishna Bhatu
UID: 116304764

Name: Kapil Rawal
UID:116133357

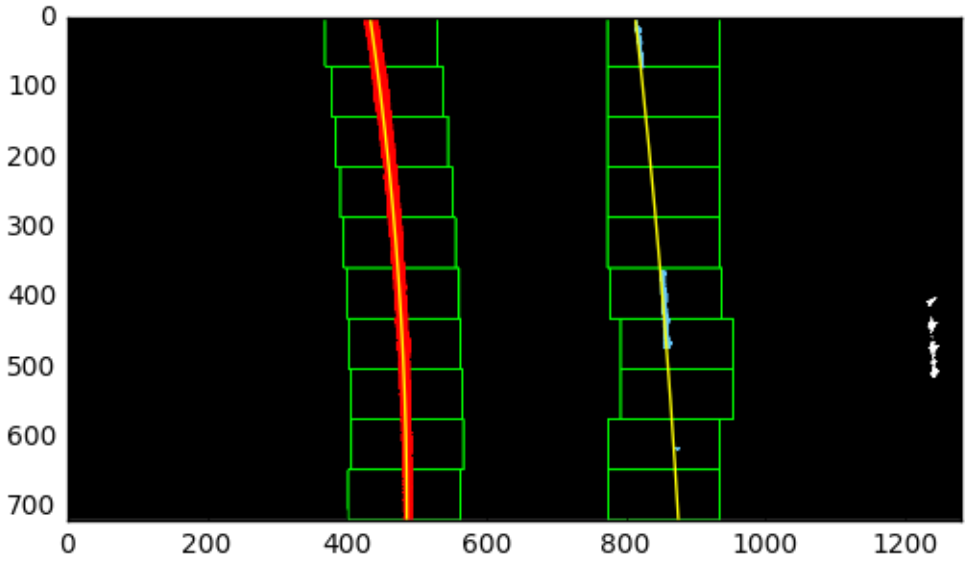

Figure 6. Sliding windows

3.4) Fitting the curve of the lane and Moving Average.

- The points found in the previous function are separated into x and y coordinates separately.
- Then we use the np.polyfit function to fit these points in a second degree polynomial and the function gives us a, b and c values. The below image shows more clarity.

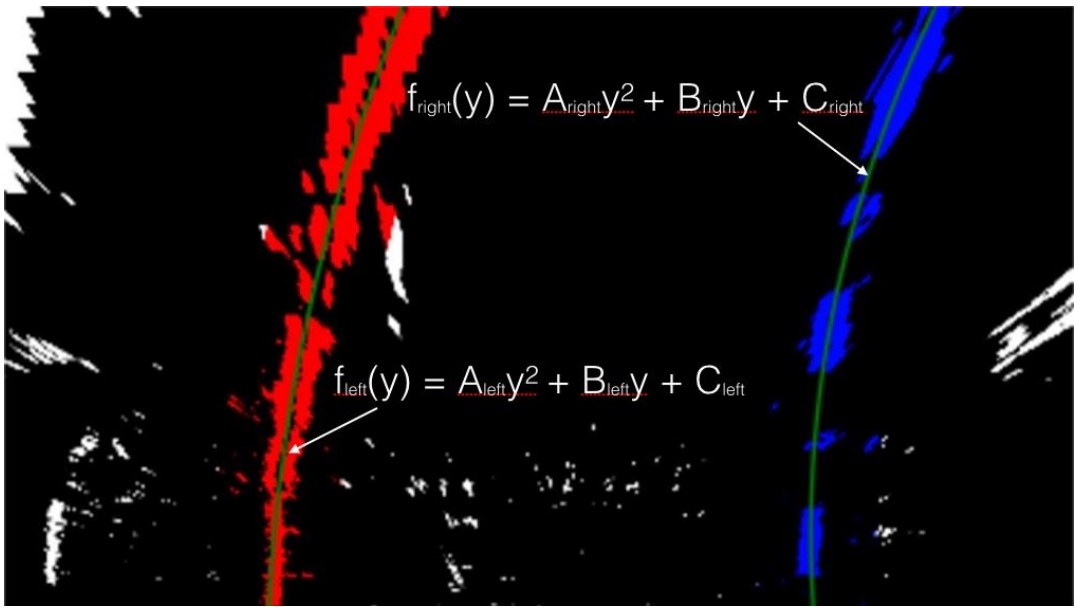

Figure 7. Fitted curves of lane

- We then accumulate the last 5 a, b and c values and calculate the moving average with every new incoming value and also remove the oldest values.
- This gave us much robust results.

3.5) Radius of curvature and Lane offset calculation

- The radius of curvature is based upon this website and in the below line of code.

$$Rcurve = |dy2d2x|[1+(dydx)2]3/2$$

- In the case of the second order polynomial above, the first and second derivatives are:

Name: Hrishikesh Tawade UID: 116078092
Name: Krishna Bhatu UID: 116304764
Name: Kapil Rawal UID:116133357

$$f'(y)=dydx=2Ay+B$$

$$f''(y)=dy2d2x=2A$$

- So, our equation for radius of curvature becomes:

$$Rcurve= ((1+(2Ay+B)^2)^3/2)/|2A|$$

- Which in code looks like below.

```
curve_radius = ((1 + (2*fit[0]*y_0*y_meters_per_pixel + fit[1])**2)**1.5) /
np.absolute(2*fit[0])
```

- In this example, fit[0] is the first coefficient (the y-squared coefficient) of the second order polynomial fit, and fit[1] is the second (y) coefficient. y_0 is the y position within the image upon which the curvature calculation is based (the bottom-most y - the position of the car in the image - was chosen). y_meters_per_pixel is the factor used for converting from pixels to meters. This conversion was also used to generate a new fit with coefficients in terms of meters.
- The position of the vehicle with respect to the center of the lane is calculated with the following lines of code:

```
lane_center_position = (r_fit_x_int + l_fit_x_int) /2
center_dist = (car_position - lane_center_position) * x_meters_per_pix
```

- r_fit_x_int and l_fit_x_int are the x-intercepts of the right and left fits, respectively. This requires evaluating the fit at the maximum y value (719, in this case - the bottom of the image) because the minimum y value is at the top (otherwise, the constant coefficient of each fit would have sufficed). The car position is the difference between these intercept points and the image midpoint (if the camera is mounted at the center of the vehicle). (Credits: Jeremy Shanon. LINK)
- Thus, we get the radius of curvature of the lane closer to the car.

3.6) Turn Direction Prediction

- Initially we used the derivate of the polynomial of left curve closer to the car to calculate the turn direction, but since that part doesn't change much or doesn't produce strong gradients we can't predict the road curvature direction accurately.
- Thus, we used the farthest point on the left lane which we used to draw the lane, to find the slope between it and a point in middle of the lane.
- When the turn is left the gradient is positive due to the opposite convention of the image coordinate system and is negative when the turn is right.
- We then found that the left turn is strong if the gradient is between 1 to 25 and for right from -25 to -1 and for the rest values of gradients say that the lane is straight.

Name: Hrishikesh Tawade
UID: 116078092

Name: Krishna Bhatu
UID: 116304764

Name: Kapil Rawal
UID:116133357

Figure 8. Fitted curves of lane