UNIVERSITY OF
MARYLAND

ENPM-673 PROJECT 2

SIMPLE LANE DETECTION FOR SELF DRIVING CARS

Submitted by:

Nikhil Mehra - 116189941

Sayan Brahma - 116309165

Contents

1. Pipeline Explanation

2. Understanding the concepts

    a. Homography
    b. Hough lines

3. Generalization

4. Screenshots

5. Some interesting problems encountered during this project

6. Conclusion and Result

Pipeline explanation: -

1. First, the image is undistorted using the calibration matrix and distortion vector provided as a supplement for the project.
2. Extract four points around the lanes (the current lane) for performing the homography operation. Perform the homography operation using the inbuilt “homography” function or the “getPerspectiveTransform” function. The acquired homography region is previewed on the screen using the inbuilt function “warpPerpective”.
3. In the homography image operations like brightness and contrast are increased using add and multiply functions respectively on the frame. Brightness and contrast are increased uniformly across the frame. Only brightness is increased for the white lane. Brightness is increased twice and contrast is increased once for the yellow lane. Parameters like “alpha” and “beta” variables are used for these changes. This is done to avoid mismatch, in different conditions like bright daylight, gloomy day, rainy day or shadow region as in tunnels or due to side plantations.
4. After getting the homography image,all the color calibrations are done. Masking of yellow color is done using the HSV color space and masking of white color is done using the BGR color space. For masking the variables like “lower_white”, “upper_white”, “lower_yellow” and “upper_yellow” are used to specify the upper and lower bounds of the colors in their respective color space(HSV for yellow and BGR for white).
5. After the masking, the Shi-Tomasi method of detecting corners is used, the function “goodFeaturesToTrack” is used on the two different masking images. Shi-Tomasi method is used to determine the corners in the yellow and white lanes respectively.
6. The corners, if detected in between the lanes are rejected. So as to choose the best-required points/regions (here the yellow and white regions in our current lane).
7. Once the required yellow and the white pixels are found using Shi-Tomasi we use the function named “polyfit” to fit a line through those pixels(corners).
8. Now using the inverse homography to find points in which the line has been to be drawn in the video, we project the lane lines on the actual frame of the video. After observing the stability of the lines fitted on the main video frame we shade the region enclosed within the lanes using the function “fillPoly”. If the stability is not up to mark we tune the parameters mentioned above to see if the stability can be increased, which we have successfully accomplished.
9. For turn detection, we compared the distance between the middle of the yellow line and white line with the middle of the frame in the homography image i.e. 100. If the mid-point is before 100 then the car is turning left,

similarly, if the mid-point is beyond 100 then the car is moving right. If the car moves straight then the mid-point is near 100 theoretically.

10. As our homography is not perfect, we have decided to give a range in which the car moves straight. If the midpoint is before 95 then the car is turning left, if mid-point is beyond 107 then the car is turning right. If the mid-point is in between 95 and 107 then the car is moving straight.
11. After this we test this code in different scenarios especially in the 2 videos provided in the assignment. There is a minor shift of the lane region in the left side under the bridge which is kind of acceptable because if we analyze each frame under the bridge in first 3-4 frames the white lines are not at all visible. And there is minor gitterning after the car comes out of the bridge as the change in intensity of light is very high. We believe this issue can be solved with superior algorithms and hardware.

Understanding of Concepts:-

Homography: In the field of computer vision, any two lines belonging to the same planner surface in the space are related by a homography. This is used for many applications like image rectification, image registration or camera motion computation.
According to the definition:

$$\begin{pmatrix} x'_1 \\ x'_2 \\ x'_3 \end{pmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

$X' = Hx$, H matrix has 8 DOF.
Homography = projective transformation = projectivity = collineation

$$\begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix} \cong \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

$$x_i' = \frac{h_{00}x_i + h_{01}y_i + h_{02}}{h_{20}x_i + h_{21}y_i + h_{22}}$$

$$y_i' = \frac{h_{10}x_i + h_{11}y_i + h_{12}}{h_{20}x_i + h_{21}y_i + h_{22}}$$

$$x_i'(h_{20}x_i + h_{21}y_i + h_{22}) = h_{00}x_i + h_{01}y_i + h_{02}$$
$$y_i'(h_{20}x_i + h_{21}y_i + h_{22}) = h_{10}x_i + h_{11}y_i + h_{12}$$

$$\begin{bmatrix} x_i & y_i & 1 & 0 & 0 & 0 & -x_i'x_i & -x_i'y_i & -x_i' \\ 0 & 0 & 0 & x_i & y_i & 1 & -y_i'x_i & -y_i'y_i & -y_i' \end{bmatrix} \begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1'x_1 & -x_1'y_1 & -x_1' \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y_1'x_1 & -y_1'y_1 & -y_1' \\ & & & & & & \vdots & & \\ x_n & y_n & 1 & 0 & 0 & 0 & -x_n'x_n & -x_n'y_n & -x_n' \\ 0 & 0 & 0 & x_n & y_n & 1 & -y_n'x_n & -y_n'y_n & -y_n' \end{bmatrix} \begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

A (**2n × 9**) **h** (**9**) **0** (**2n**)

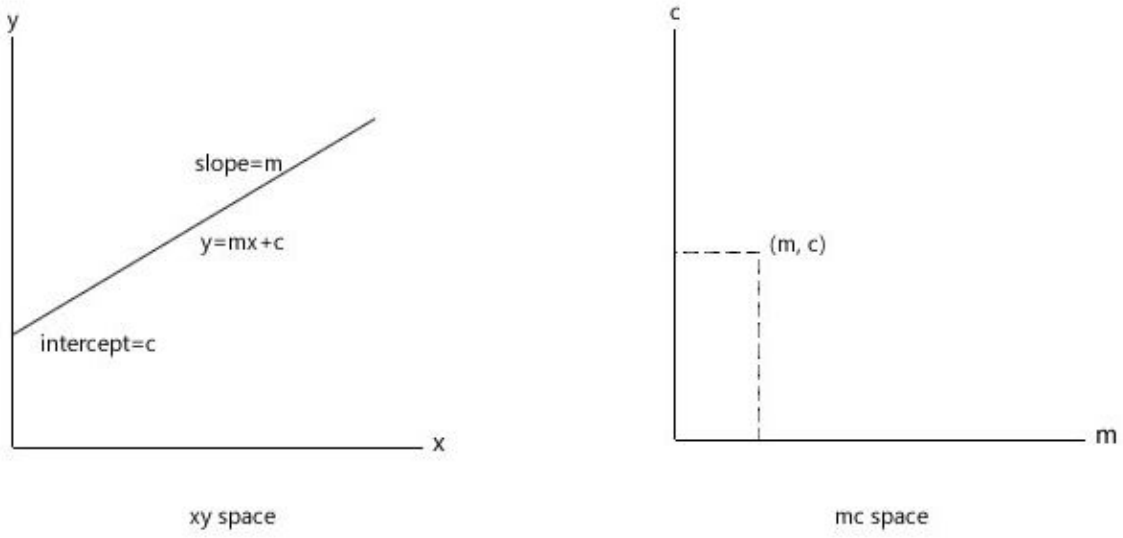
Finally the solution for h = eigenvector of A^TA with the smallest eigenvalues.
Once we have got the homography matrix we now apply this matrix to get the region of interest without any perspective i.e., bird's eye view.

For performing tasks like projecting a certain part image we use homography. In case of projective something back to the image inverse of the homography matrix is used.

Hough lines – Hough lines is a tool which lets us identify lines in this project. Hough method not only identifies lines but also other shapes as well. In a practical point of view, once we have detected the edges and we want to fit a line through those points. We need to do fit all possible lines from 0 to pi on all the necessary points. We keep a count of all possible pairs of (slope and intercept). Hough lines lets each point of the image “vote” and because of the mathematical properties of the transform, this “voting” allows us to figure out the prominent lines in the image.

From lines to points:

A line is a collection of points, we represent a line as a single point, without losing any information about it. It is done through the m-c space. Every line can be described completely with 2 parameters i.e., intercept and slope. With these 2 entities, we can equivalently represent each line in mc-space.



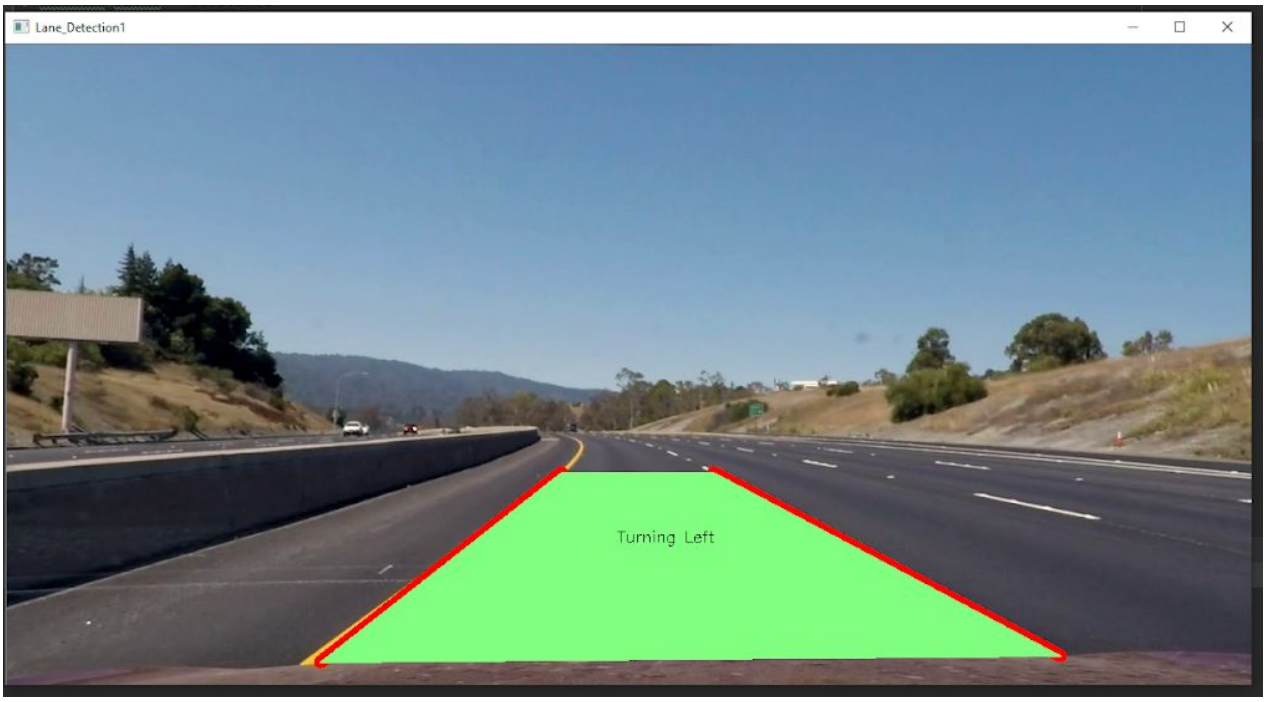
The m-c space

Generalisation

1. The resulting code is quite generic and will work in most of the scenarios containing a yellow lane or a white lane.
2. In the challenge video the lanes shift a little due to lack of visibility of the white light, hence the program might fail in darkness.
3. The program will fail if the color of lanes is not yellow or white.
4. The program will fail if there is a sharp 90 degrees turn.
5. Once the car gets out of the bridge shadow region there is a sudden increase of light intensity because of which all the colors get blown out.

Screenshots:-

Project_video




PRANALI DESAI - 116182935
Lane_Detection1
Straight


Lane_Detection1
Straight

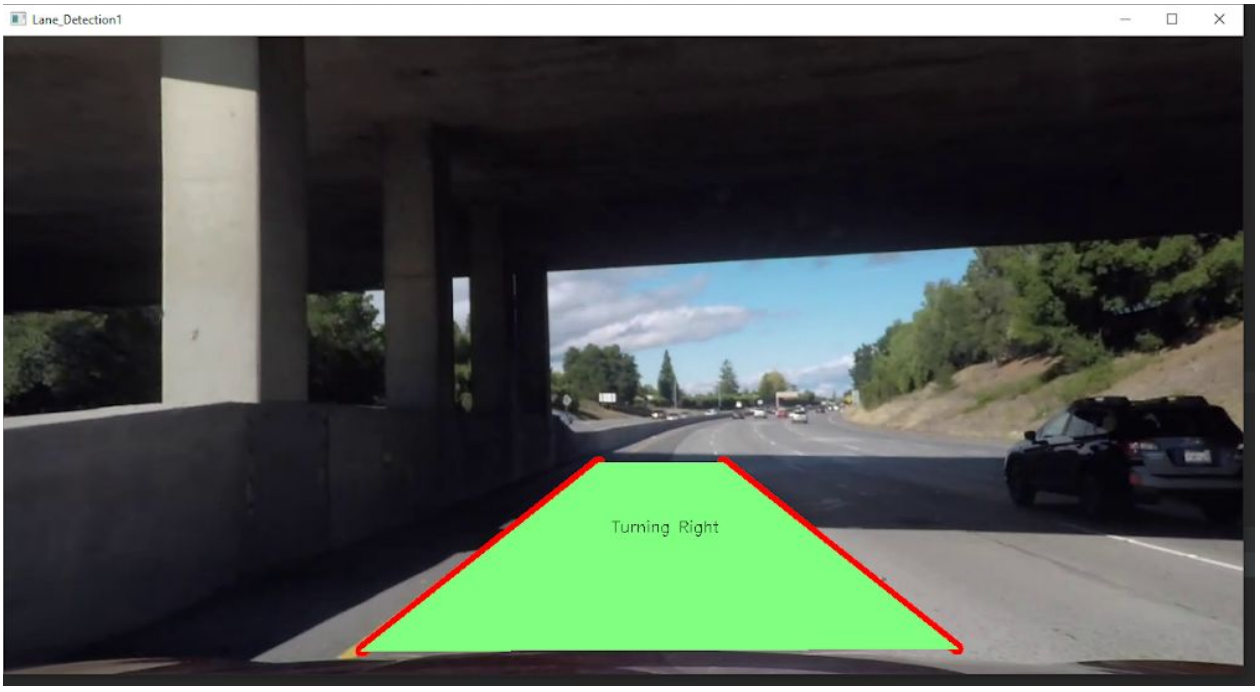

Lane_Detection1
Straight

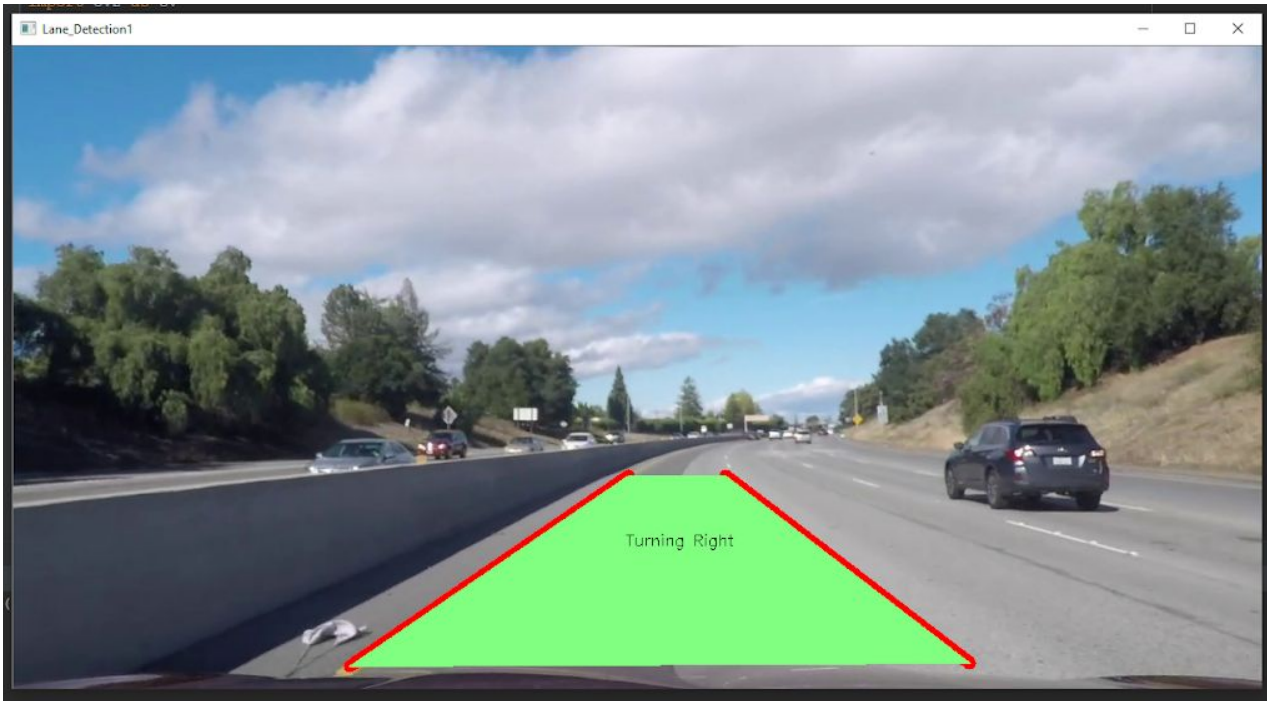

Lane_Detection1
Turning Right

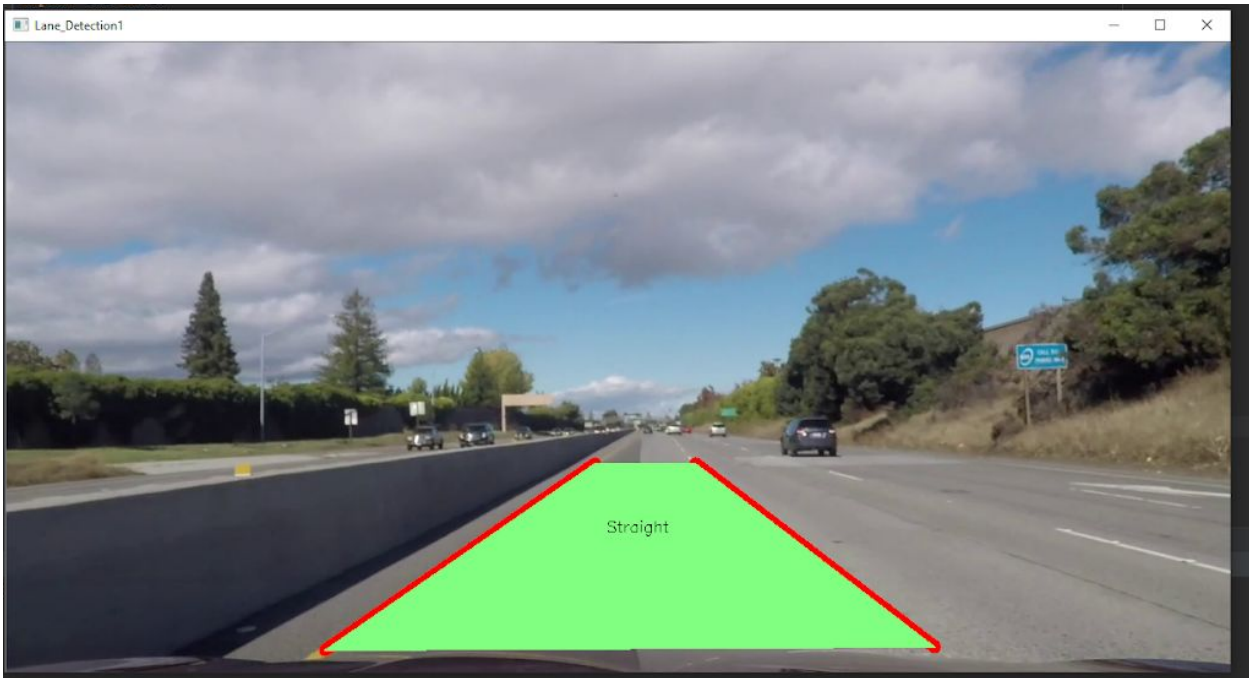
For project_video the lane was very stable and there was no distortion in the shadow or the gray area.

Challenge video


Lane_Detection1
Turning Right


Lane_Detection1
Turning Right





For the challenge video, we faced some problems. The lanes shift a little under the bridge due to lack of visibility of yellow lanes, and while coming out of the bridge since the intensity increases all of a sudden so the colors get blown out.

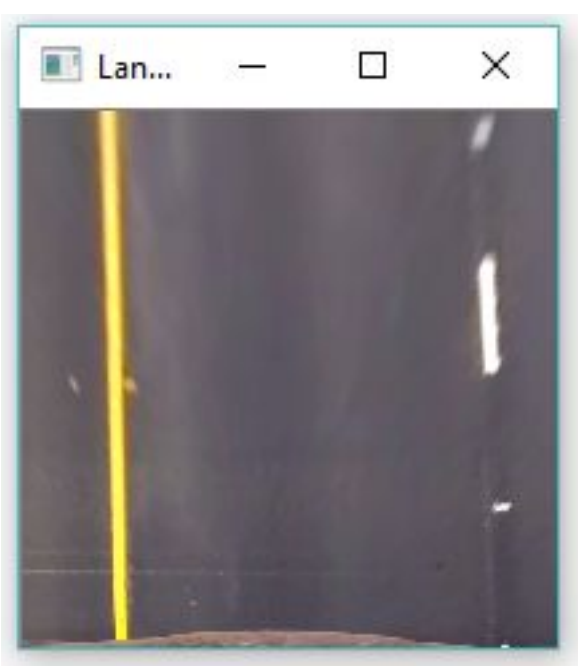
Some interesting problems encountered during this project:

1. We were using the normal gaussian blur but we were not getting good results with it, so we came across a new type of filter known as Bilateral filter which

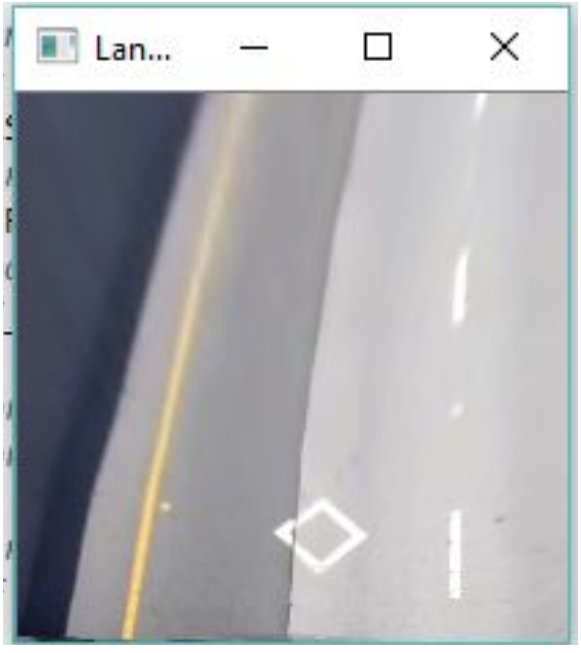
preserves the edges while blurring, though after that we decided not to use the blur operation at all since that fetched better results.

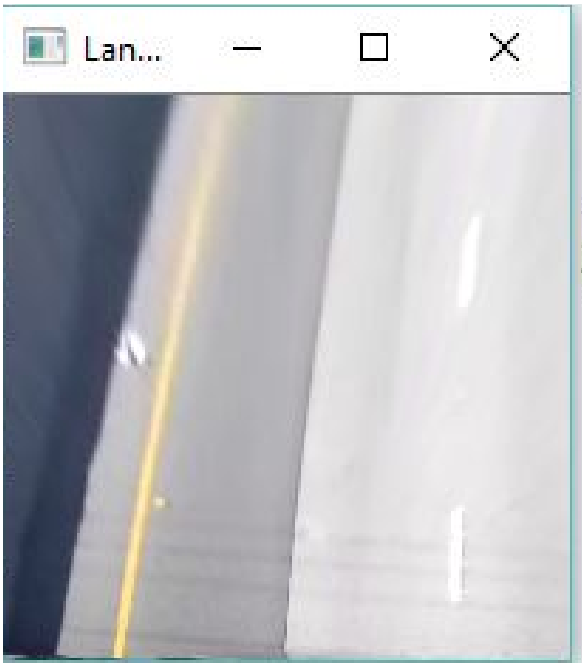
2. We also tried to deploy RANSAC for rejecting the outliers and form lane lines, we achieved very good results with this but the frame rates were compromised i.e., the video became slow.

Homograph window from project_video



Homograph window from challenge_video





Conclusion and Result:

For project_video the lane was very stable and there was no distortion in the shadow or the gray area.

For the challenge video, we faced some problems. The lanes shift a little under the bridge due to lack of visibility of yellow lanes, and while coming out of the bridge since the intensity increases all of a sudden so the colors get blown out.