Cruise Automation Computer Vision Challenge
(Naman Kumar)

<u>Problem</u>: The problem was to detect Lane Markings in a Real world scenario using Computer Vision Techniques.

<u>Approach</u>:
Step - 0 - I have saved the name of all the images in a text file “imNames.txt” and I am using this file to get the name of the image and then I read that image.

Step - 1 - I read the original image and I ignore the top half of the image. Then, I convert it into a Edge Map as shown below using a filter to check change in the vertical direction. It is useful in this case because we dont want to detect horizontal edges.

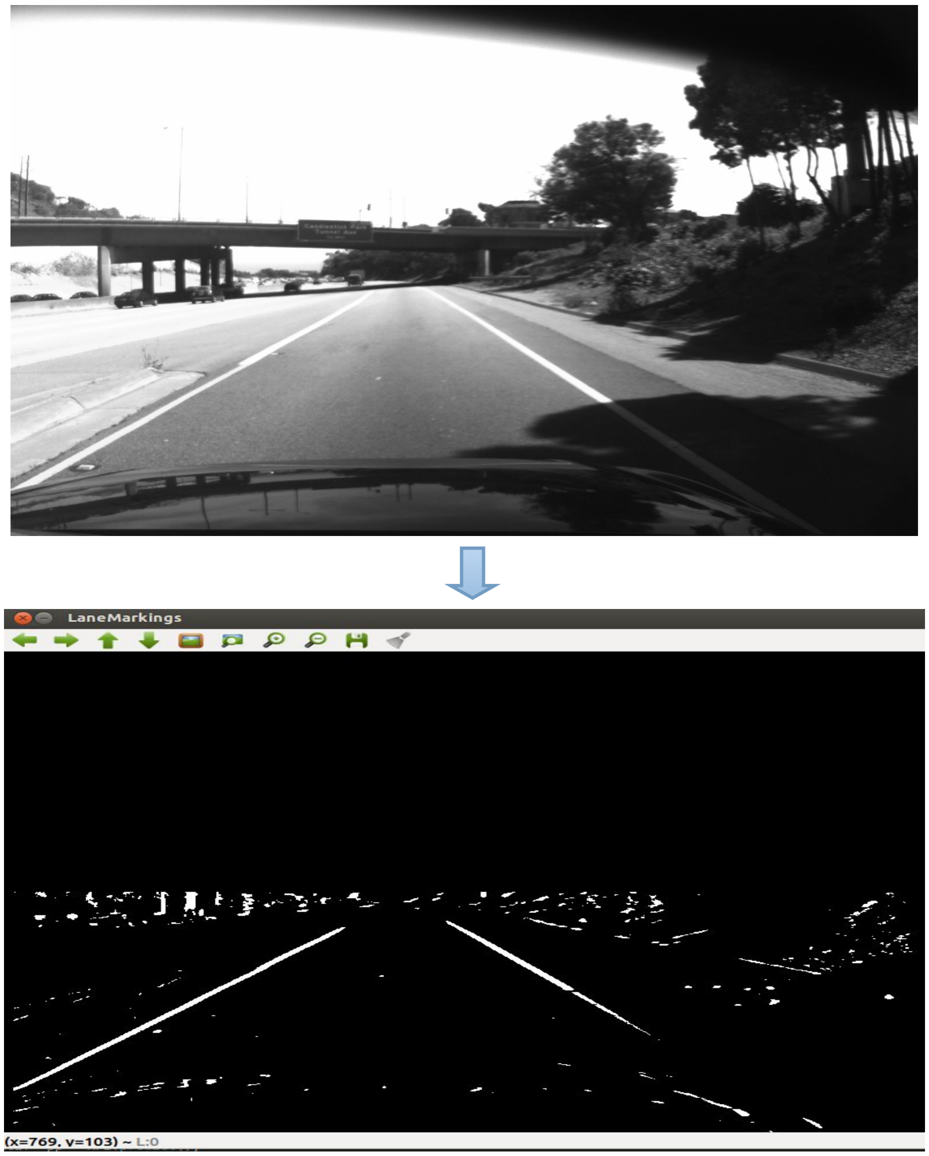


Figure 1. Original Image and its Edge Map

Step - 2 - I generated the hough lines of the binary image generated above which are shown below.

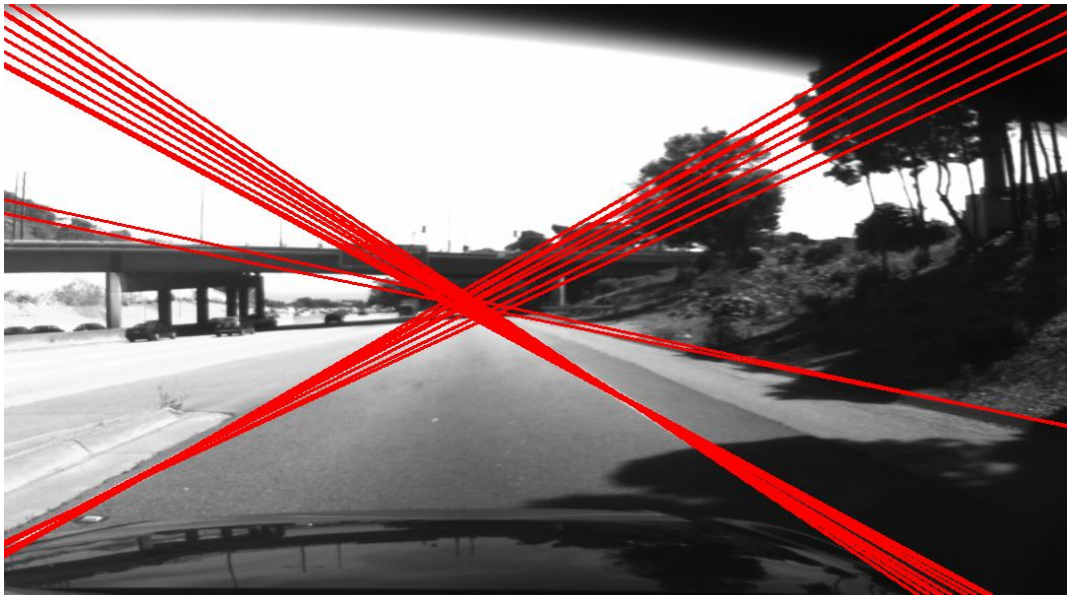


Figure 2. Hough Lines

Step - 3 - Then I used K-means clustering to cluster the hough lines into two clusters (for the left and the right lane marking). The lines in the green and the blue circle show different clusters.

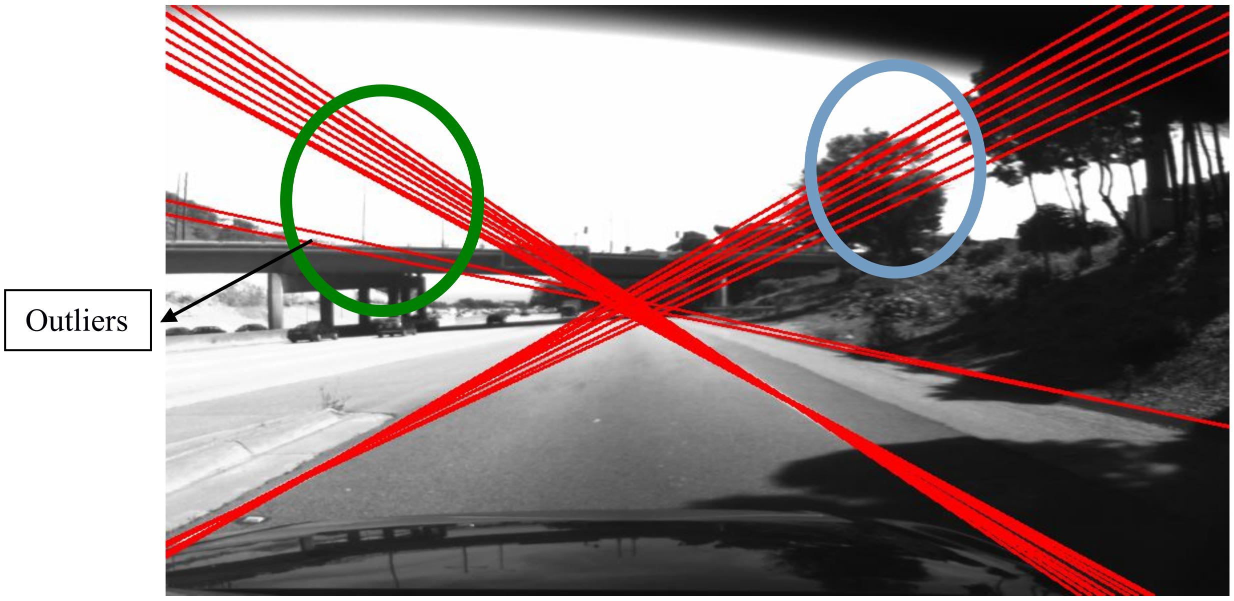


Figure 3. Hough Lines divided into two clusters shown with Green and Blue Circles. Outliers are also indicated above.

Step - 4 - Then, I used RANSAC to remove outliers from these two clusters (shown in the above image) and come up with the best line for each of the two clusters. The final result is shown below.

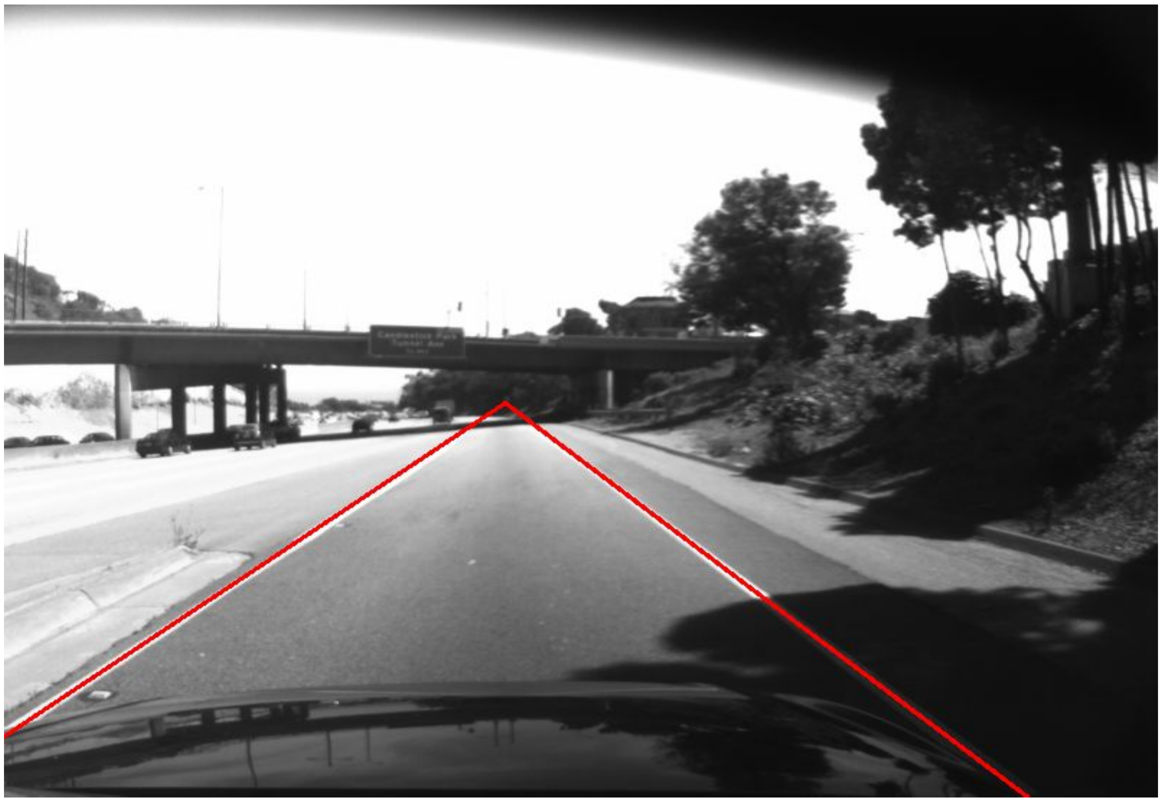
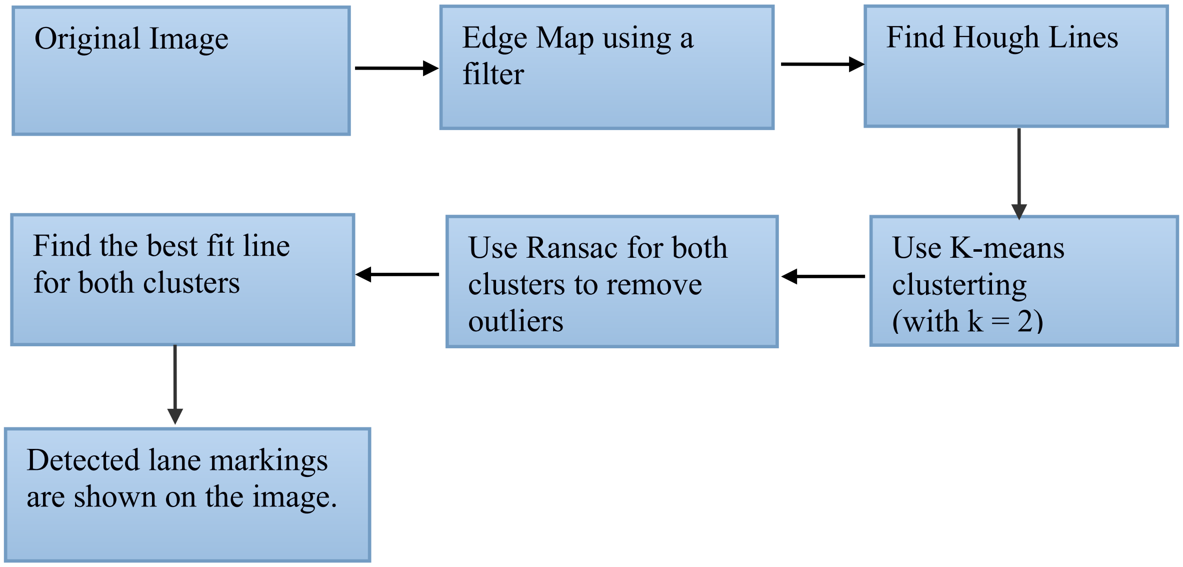


Figure 4. Final detected lane markings

The flowchart is shown below:


Original Image
Edge Map using a filter
Find Hough Lines
Use K-means clusterting (with k = 2)
Use Ransac for both clusters to remove outliers
Find the best fit line for both clusters
Detected lane markings are shown on the image.
Figure 5. Block Diagram showing the DETECTION of lane markings

A) Note : The above approach works well if the lane markings are visible on the road, that is we are able to detect the lane markings. If the lane markings are not visible or not detected properly, then we can use the information from the previous frames to predict the lane markings for the current frame. There are many approaches for that:

1. Optical Flow
2. Kalman Filters
3. Particle Filters (Condensation Algorithm)

Kalman Filter

1. I implemented the standard Kalman Filter algorithm for predicting the lane markings. The dimensionality of the state and measurement is 4(2 for each lane marking).
2. With each detection, I am updating the model and if I am not able to detect the lines in a given image using the technique mentioned above, then I use the lane markings predicted by the Kalman Filter which is being updated with each detection.

If I had more time, I would work on:

1. Getting the Camera calibration parameters and performing Inverse Perspective Mapping to convert the normal view to a Bird eye view and then work on the bird eye view. An example from http://breckon.eu/toby/demos/roadtext/ is shown below:

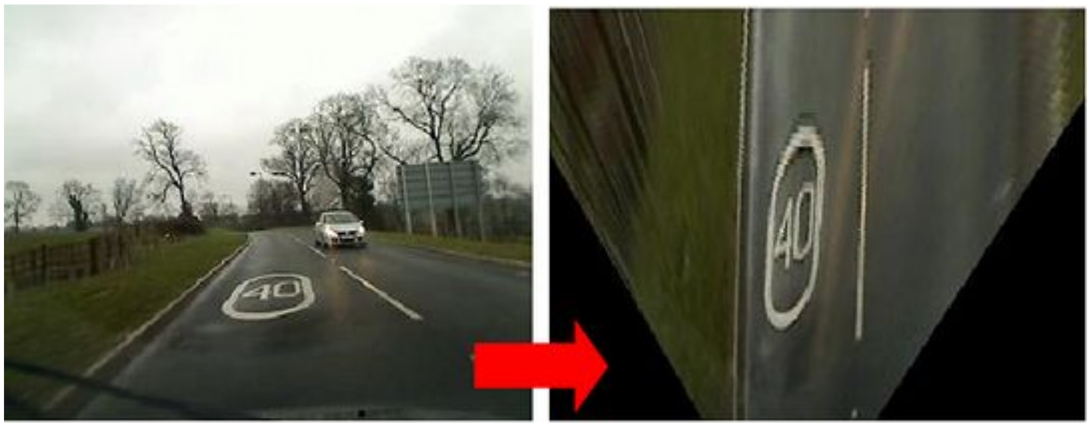


Figure 6. Example of Inverse Perspective Mapping

2. Improve the implementation of Kalman Filters and implement Condensation algorithm (Particle Filters) to perform tracking.

3. Use something better to fit curve lane markings.

How to run:

1. Create a folder and put all the source, header and makefile files in it. Also, put the images folder “images” and the text file containing the name of all the images “imNames.txt” in it. Optionally, create a folder “output” to save the results.
2. Open the terminal and run “make” in the above folder.
3. Run the binary file generated via terminal “./main”. It will show the images with lane markings overlaid on it. Also, it will save the “intercepts.csv” file. If the x-intercept of the left lane marking is less than 0, I write it as 0.