**Introduction**

The increasing demand for fully autonomous vehicles has led scientists and inventors to come up with hardware and software architectures to ensure safe and sustainable autonomous driving experiences. In the latter half of the twentieth century, manufacturers accentuated the need for more sophisticated hardware solutions; however, within the last few years, the focus has shifted to cutting-edge software technologies to foster an end-to-end driverless vehicle experience. These technologies have so far sought to unpack what we have come to know about driving in a bid to automate a spectrum of driving tasks - from the most trivial to the most complex. A prime example of such a task would be perfectly aligned vehicle parking which now has numerous Park Assist System solutions from renowned manufacturers. Beyond this, other complex cases surface while a vehicle is in transit. Of all the complexities that researchers seek to control and automate in a moving vehicle, lane detection, and localization estimation emerge as fundamental challenges. In fact, data collated and analyzed by the US Insurance Institute for Highway Safety suggests that the systems that tackle the aforementioned challenges have had a direct impact on reducing crash statistics [[1]](https://paperpile.com/c/faT9wA/2WIW). In this project, we set out to address these cardinal challenges - lane detection and distance estimation - using fast and robust algorithms that do not require a lot of computing power and can run on edge devices like microcontrollers (a Raspberry Pi in this case).

**Problem Definition and Importance of the Problem**

It is imperative to start this study with a clear and concise definition of a lane. A lane is a region bounded by two-line features upon which single-way traffic is allowed to flow. For the sake of simplicity, the lanes referred to in this paper would be clearly defined by white lines on the freeway.

Over the years, lane detection has become increasingly important due to the gradual transition from manufacturing traditional automobiles that require little to no level of automation to the research and development of fully autonomous vehicles. According to the US Department of Transportation’s National Highway Traffic Safety Administration (NHTSA), there are six automation levels (from level 0 to level 5), of which levels 1 through 5 require an Advanced Driver Assistance System (ADAS) or an Automated Driving System (ADS) - in higher levels [[2]](https://paperpile.com/c/faT9wA/uZIHs). Lane detection finds its application in a plethora of areas across the abovementioned automation spectrum, especially in levels 3, 4, and 5.

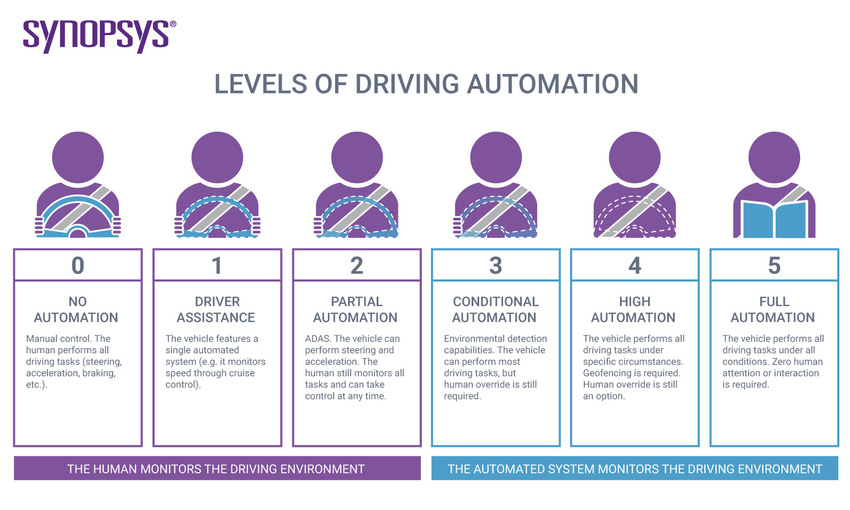


Fig. 1. SAE Automation Levels [[3]](https://paperpile.com/c/faT9wA/4DVPK)

The process of lane detection could be viewed as a “calibration process” for autonomous vehicles in the sense that a crisp identification of the lanes on a motorway is a key precursor for the decision-making process of automated driving systems. A poignant example of this is the case of Simultaneous Localization and Mapping (SLAM) where detected lanes could serve as landmarks for the SLAM algorithm [[4]](https://paperpile.com/c/faT9wA/ucQaA), and the alignment of the vehicle with the lane(s) could be used for accurate pose estimation [[5]](https://paperpile.com/c/faT9wA/zfXp). Equipped with such information amongst other sensor measurements, a near-pristine map of the road network could be constructed. The authors of [[6]](https://paperpile.com/c/faT9wA/St69) identified three broad categories of lane detection algorithms namely, model-based algorithms, feature-based algorithms, and learning-based algorithms. The vital importance of robust lane detection algorithms has been underscored by a host of researchers over the years. However, the likes of [[7]](https://paperpile.com/c/faT9wA/tHWkK) indicated that the constraint on computational resources is a limiting factor for achieving real-time lane detection. In the foregoing sections, we implement a feature-based algorithm that achieves near real-time detection without being computationally demanding.

A system underpinned by a sophisticated lane detection algorithm can only go a long way in solving the complex problem of fully autonomous driving. This is why distance estimation is also considered a primary issue of concern in this discourse. In SAE level 0 where the driver is in complete control of the car, deciphering the distance from the car to oncoming traffic or other obstacles is a task that is completely dependent on the strength of the human eyes. In this case, perception plays an important role in estimating distance. However, as we go higher on the autonomy scale, the goal is to use plausible solutions in tackling this problem; therefore, it is imperative that the solution in question not only performs like the human eye but also offers much more flexibility and robustness. More technically and generally referred to as localization estimation, the challenge of distance estimation has been approached differently by researchers across the years. The upshot of this is a dichotomy of solutions proposed in the literature - active solutions and passive solutions. While the active solutions explore the use of sensors (like LIDAR and ultrasonic-based sensors, to name a few) that inevitably emit energy to the environment to measure some response or reaction based on the age-old time-of-flight principle, passive solutions employ non-energy-emitting sensors [[8]](https://paperpile.com/c/faT9wA/xo2z). For a system constrained to implementing only computer vision solutions (which are passive solutions), a stereo set of cameras could be used in the ego vehicle. This allows a relatively easy elimination of depth ambiguity - a major computer vision problem - through the knowledge of the disparity between simultaneously taken image/video frames of the different cameras involved. Using a monocular camera (like in our case), on the other hand, proves to be a more daunting approach to solving the distance estimation problem. For this reason, reams of publications have been released that postulate different methods (some of which are novel) to approach this. Some of these methods include but are not limited to, road geometry-based methods, object geometry and orientation-based methods [[9]](https://paperpile.com/c/faT9wA/Qnyd), deep learning methods, etc. On the grand scale, depth estimation in vehicles is an integral part of different computer architectures for autonomous driving [[10]](https://paperpile.com/c/faT9wA/C5XR) like emergency braking systems, lane change assistance systems, path planning, collision avoidance systems, Dynamic High Beam (DHB) assist systems, etc.

**Problem Formulation**

**Lane Detection**

As stated in the previous section, the approach we take to this task is purely from a computer vision point of view. We seek to detect lane markings on a freeway using image features with the following key characteristics:

1. Local: these features can be observed around neighborhood pixels. In the implementation of the algorithm, a region of interest will be “isolated” within the image frame under consideration.
2. Meaningful: the features must be proper representations of real aspects of the real scene. Lines would directly translate to lanes within the region of interest.
3. Detectable: based on the above 2 points, the features must be computationally detectable using algorithms. We make use of the probabilistic Hough Line Transform function in OpenCV for this. [[11]](https://paperpile.com/c/faT9wA/bQ89)

Given an image of a road taken from a camera mounted on the dashboard of an ego vehicle, there are key factors and assumptions to be mindful of in formulating the lane detection algorithm:



Fig. 2. A photo from the KITTI Road dataset. [[12]](https://paperpile.com/c/faT9wA/5rti)

1. For a moving vehicle on a freeway, there are two white lines (lane markings) on either side.
2. The camera is mounted in the car such that it is between the two white lines (preferably at the center).
3. Due to perspective projection, the lanes meet at the horizon.
4. Following the above, the region of interest is usually chosen in such a way that it excludes every detail outside the lanes. This implies that we end up with a triangular area, which could be extended to an n-sided polygon for convenience.

Since the feature of interest is lane marking, the seemingly arduous detection problem becomes a case of line detection.

**Line Detection Using Hough Transform**

The Hough Transform, which was initially introduced by Paul Hough in 1959, was based on the notion that a straight line with the general equation in (1) could be mapped to a point represented by its slope and intersect parameters in a different coordinate space [[13]](https://paperpile.com/c/faT9wA/XJlZ).



Fig. 3. A photo from theCULane dataset. [[14]](https://paperpile.com/c/faT9wA/SOPt)

This method, however, was limited in the sense that it could not accommodate special cases like when vertical lines (with infinite slopes) are to be detected (For instance, the lane in Fig. 3. - assuming it is perfectly vertical - would not be detectable using the slope-intercept parameter space). This translated to an unbounded parameter space [[15]](https://paperpile.com/c/faT9wA/rT1D) which made discretization impossible. To remedy this problem, R. Duda and P. Hart in their 1972 paper suggested that the polar equation of a line (2) be used such that the parameters of interest become the orientation of the line and its perpendicular distance from the origin [[16]](https://paperpile.com/c/faT9wA/aJcY).

Going by the rho-theta parameterization, lines in the image space would map to sinusoids in the parameter space. The cornerstone idea behind this is that we count the number of sinusoids that intersect at the same point in the accumulator space, and the line(s) of best fit is/are chosen as the line(s) with cumulative intersections that exceed a user-defined threshold.

**Solution Method**

The following flowchart summarizes the steps involved in the lane detection algorithm.

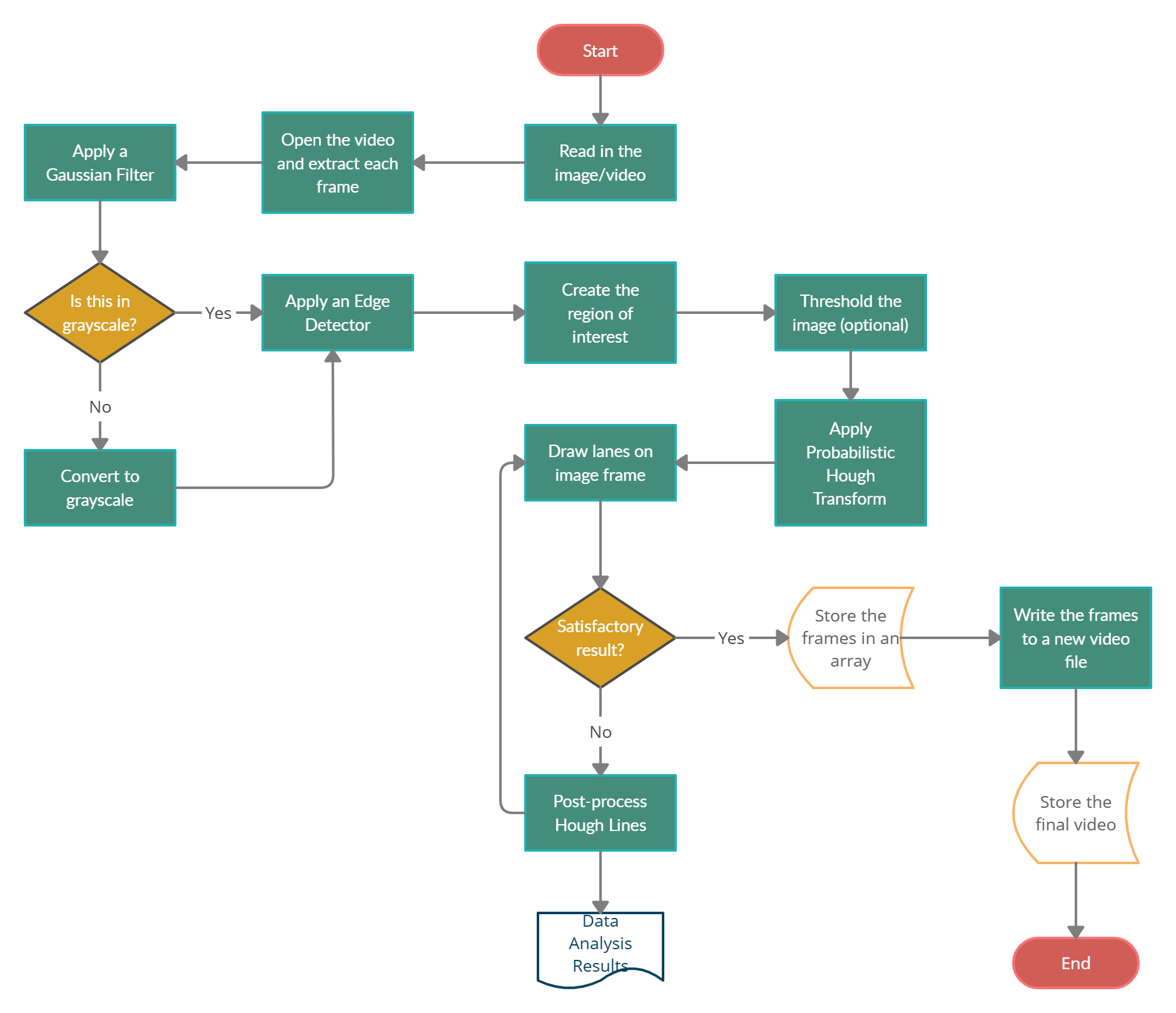


Fig. 4. Flowchart of the Lane Detection Algorithm

**Image Smoothing**

Since we are unsure about the exact distribution of the noise in each image frame, it is not out of place to assume that the noise is normally distributed and apply a Gaussian filter. We apply this filter because it is based on a simple linear convolution and it prevents the further propagation of noise through the image.

Prior to applying the Gaussian filter, the image is converted to grayscale to avoid having to deal with multiple image channels simultaneously. This significantly reduces the computational time.

**Edge Detection**

The Edge detection algorithm of choice is the Canny Edge Detector [[17]](https://paperpile.com/c/faT9wA/CcCZ). We opted for this optimal edge detector because of how well it manages noise smoothing, edge enhancement, and edge localization. More significantly, the Canny algorithm allows us to implement **Hysteresis Thresholding** to eliminate both false positives and false negatives.

**Region of Interest**

Observe the following edge image for a make-shift lane for a Raspberry Pi-based wheeled robot:

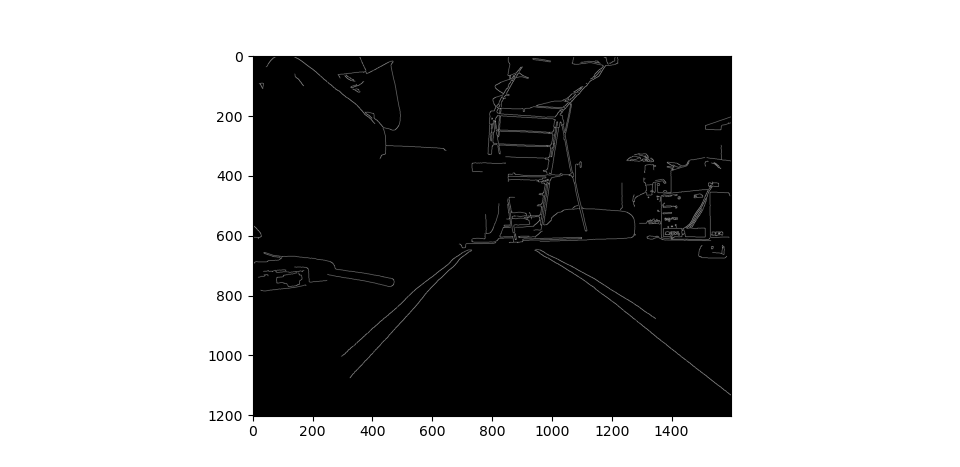


Fig. 5. Edge Image (Using the Canny Edge Detector)

It is apparent that any attempt to detect the lanes (without any machine learning implementation) would be directly affected by the background edges in the image. This necessitates the identification of a region of interest. For simplicity, the camera is assumed to be mounted at the center of the dashboard of a car almost centrally located between two white lane markings. This makes it easier to estimate the location of the region of interest.

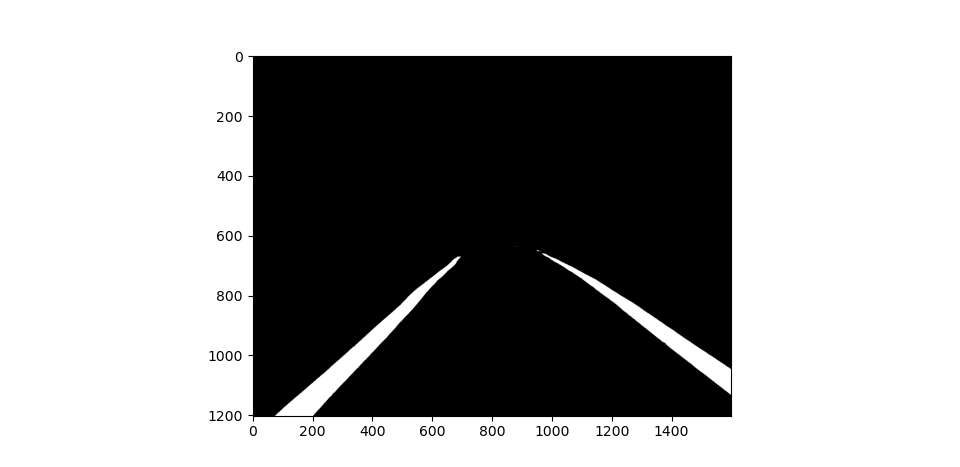
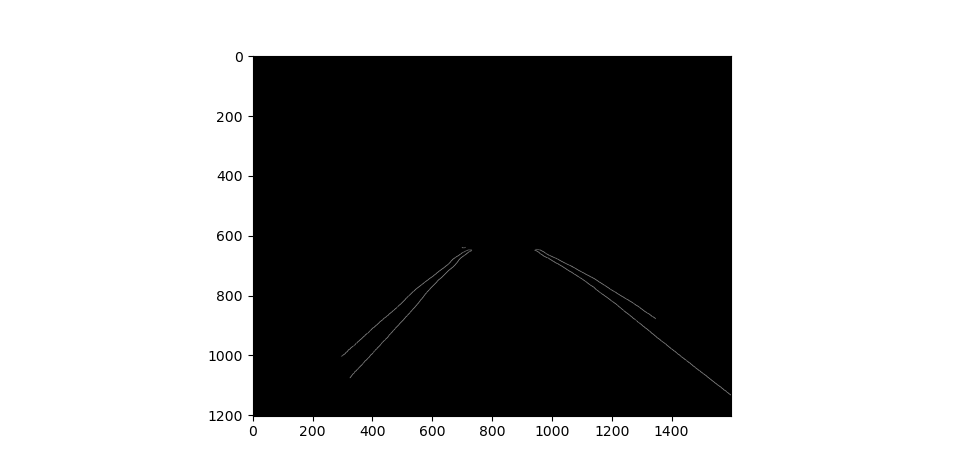
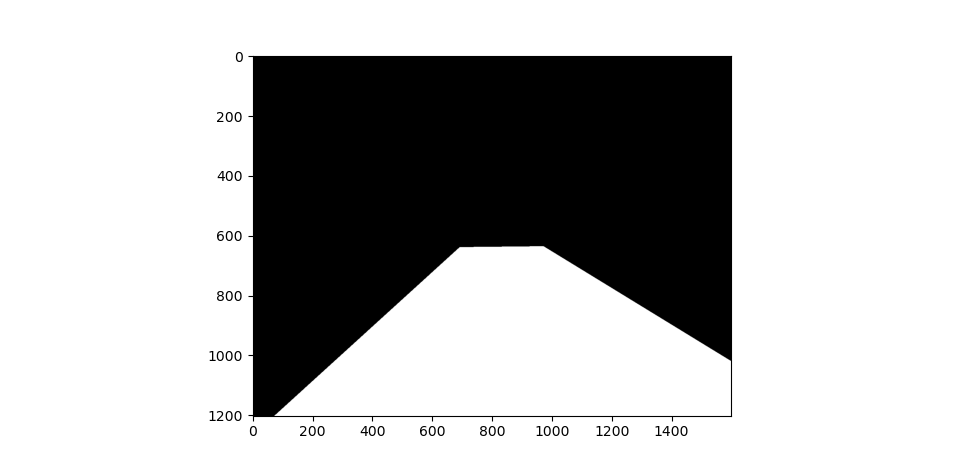


Fig. 6a, b & c., Image Mask; Region of interest; Threshold Result (without edge detection)

A mask with the same resolution as the original frame is created (Fig. 6a.). This is used to cut out the region of interest by the mathematical **bitwise AND operation** (Result: Fig. 6b.). This is an element-by-element binary operation where a 1 is returned if and only if 1 is operated on by another 1.

Another approach would be to simply threshold the region of interest (as opposed to finding the edges) as seen in Fig. 6c. The downside of this approach is that a manually defined threshold is not adaptive; hence, it would fail under different conditions.

**Apply Probabilistic Hough Transform**

We chose the probabilistic Hough Transform over the standard one because it returns the coordinates (x,y) of the line, not the parameters (,) from (2).

The detected lines are shown in Fig. 7a. To avoid inconsistencies, it is imperative to reduce the number of lines to a final lane marking on either side. This is done by a post-processing procedure that involves performing *data analysis* on all the lines detected to choose the optimal one.

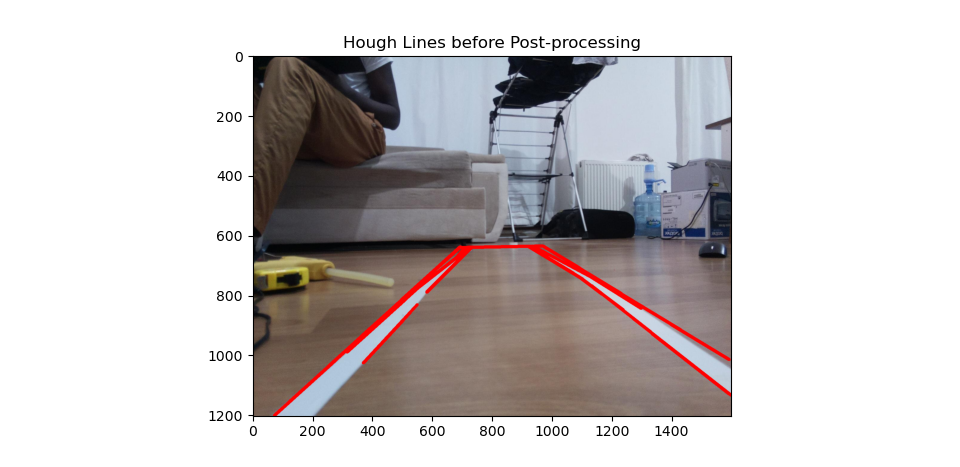
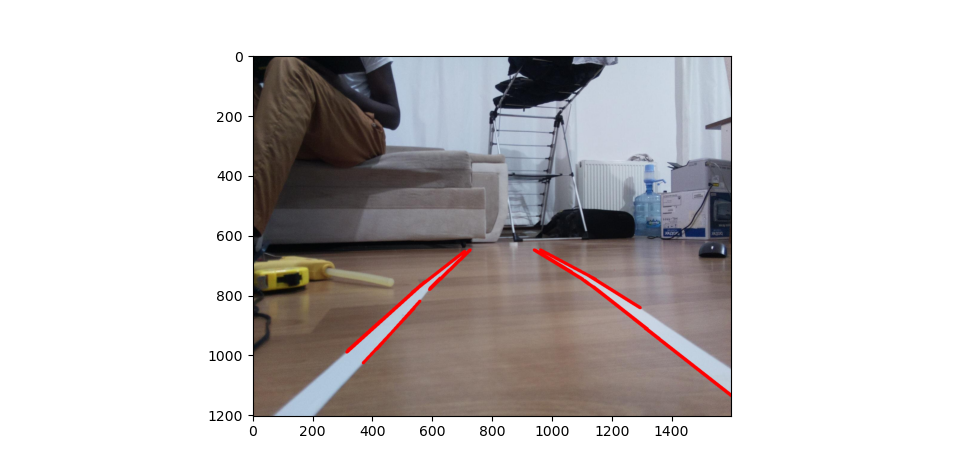


Fig. 7a & b., All detected Hough Lines; Hough Lines with Multiple False Positives.

**Post-processing (Data Analysis) of Hough Lines**

From Fig. 7a, it is clear that simply choosing one of the detected lines may not necessarily yield an ideal outcome. In some extreme cases, some detected lines may be considered as outliers when compared to the others (Fig. 7b. clearly contains some outliers). In such a case, we would need clever methods to get rid of the outliers.

**Data Analysis Methods**

We make use of the convenient Pandas library in Python for this process [[18]](https://paperpile.com/c/faT9wA/jhAE). Pandas is a sophisticated library that offers myriads of features that could be leveraged for data manipulation. Within this library, we use the “DataFrame” object which is ideal for indexing.

Data frames were created for all the detected Hough lines in two groups - lines with negative slopes (representing the left lane) and lines with positive slopes (representing the right lane). Statistical measures of central tendency and dispersion (mean, median, interquartile range, skew) and visualization methods (Box plot, scatter plot) were applied to eliminate outliers. From Fig. 7b., the horizontal line is clearly an outlier, so using the aforementioned tools, it could be eliminated.

Post-processing Fig. 7b.

Interquartile Range

This is a measure of statistical dispersion that corresponds to the difference between the 75th percentile and the 25th percentile.

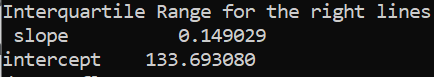
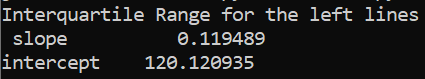


Fig. 8a & b., Interquartile Ranges

To identify the outliers, we apply a logical function in tandem with the IQR such that outliers would return a true value while inliers will return a false value. For the same set of left and right lines:

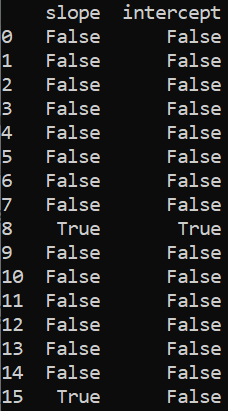
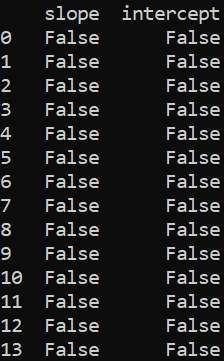
 

Fig. 9a & b., Outlier Identification using IQR

From the above, we can see that two lines in the set of left lines are outliers (one of the lines has an inlying slope).

Skewness

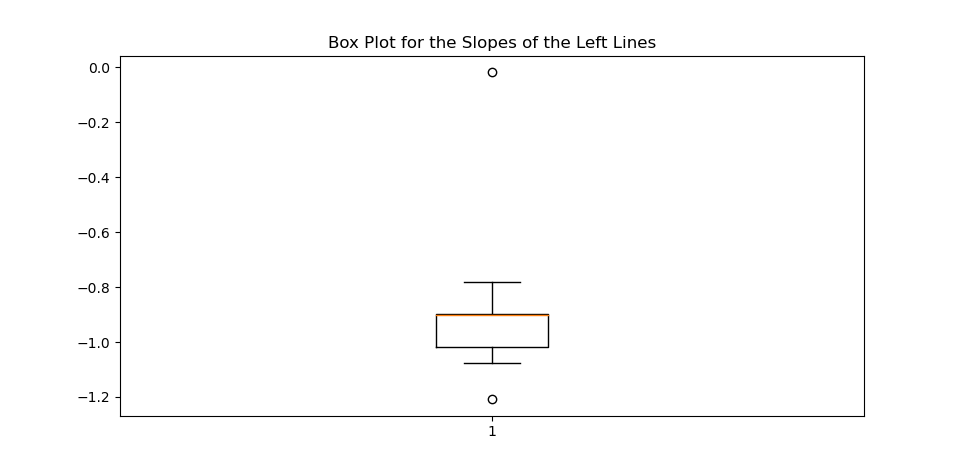
 

Fig. 10a & b., Outlier Identification using Skewness Value

Generally, skewness values outside the range - are considered too high, and such are pointers to the presence of outliers [[19]](https://paperpile.com/c/faT9wA/F5g6). The values above confirm our conclusion from the IQR analysis.

Visualizing Outliers

Box Plot



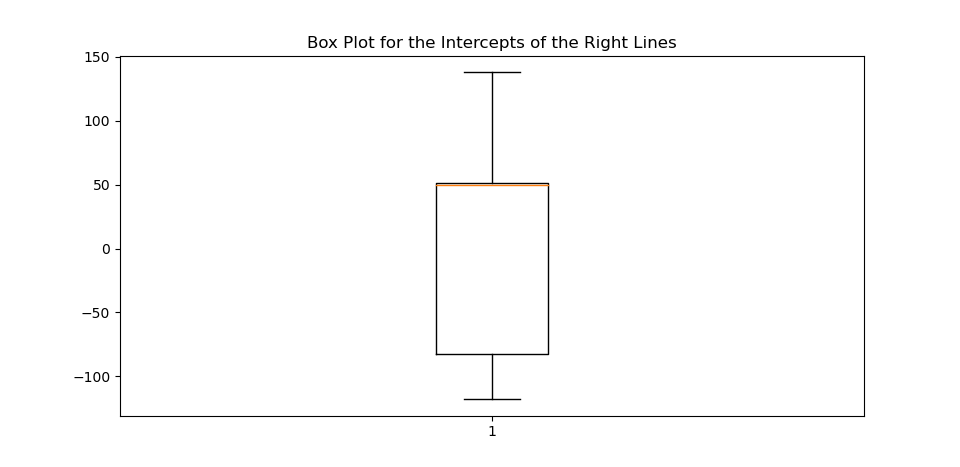
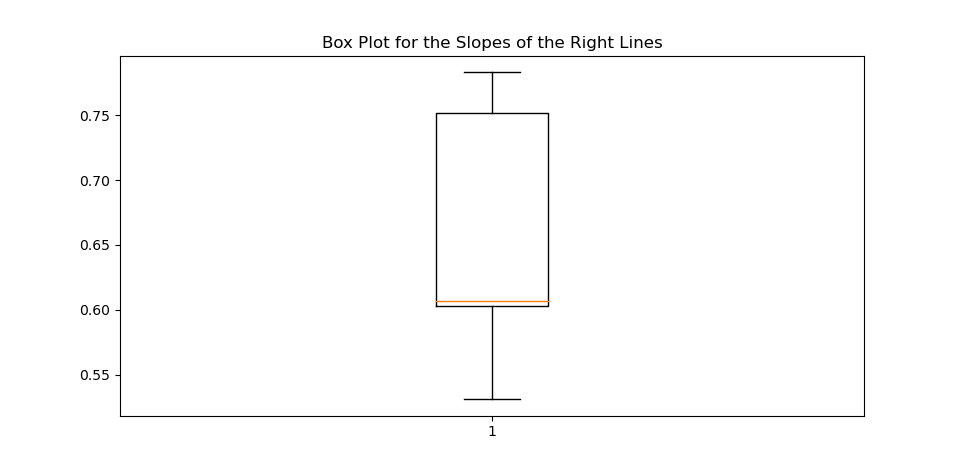


Fig. 11a, b, c & d., Outlier Identification using Box Plots

Consistent with the previous observations, outliers exist in the left lines dataset.

Scatter Plot

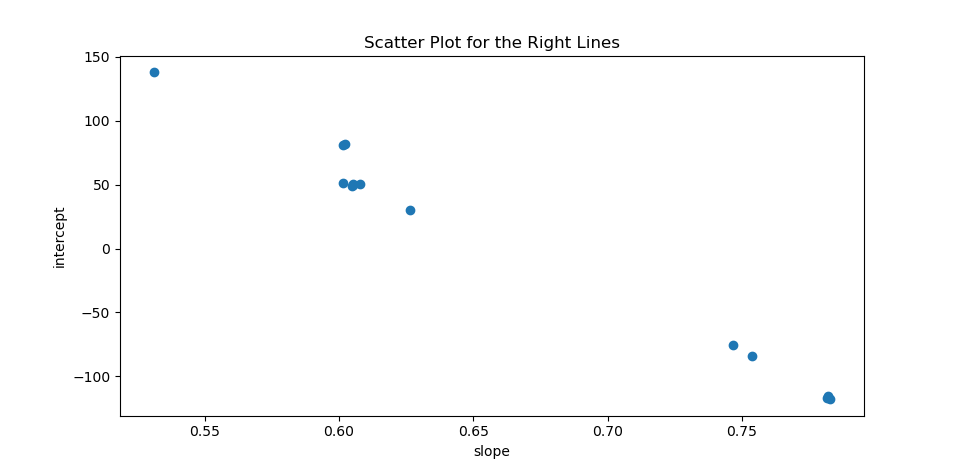
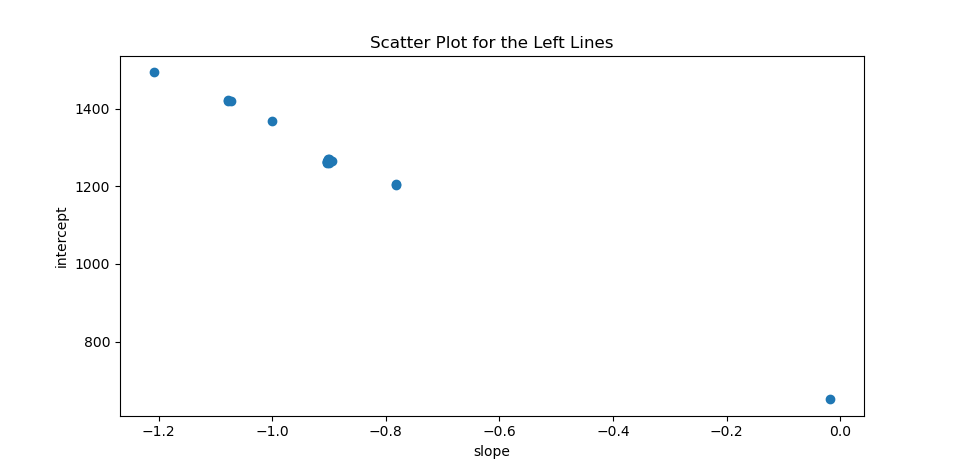


Fig. 12a & b., Outlier Identification using IQR

Handling Outliers

There are several ways to eliminate outliers; however, for the sake of brevity, we implemented quantile-based flooring and capping. This is predicated on the concept that the minimum value in a dataset is flooring at a specific percentile; correspondingly, the maximum value is capped at a certain percentile.

For the lines in Fig. 7b., we applied a 25th percentile flooring and a 75th percentile capping. This eliminates the outliers we observed in the previous sections leaving us with data from which we can take the mean or median slope and intercept to construct the final lanes as seen in Fig. 13a & b. Clearly, using the mean to obtain the final lanes produces better results compared to using the median.

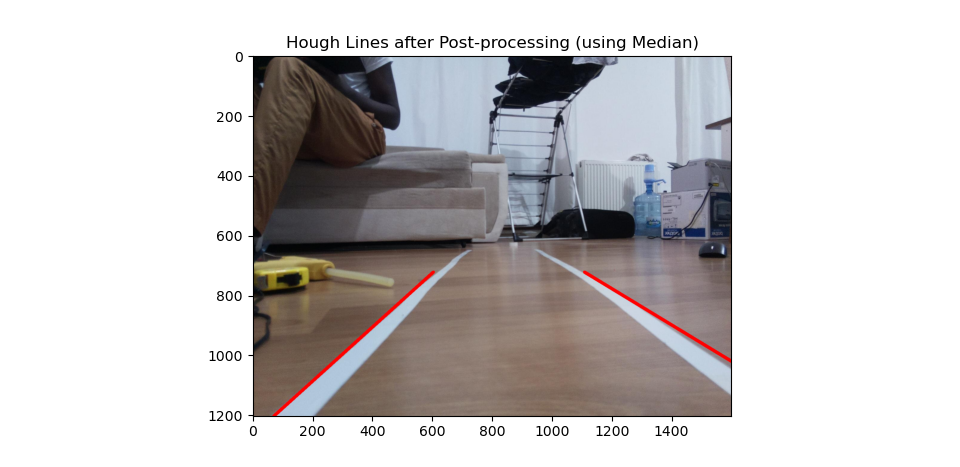
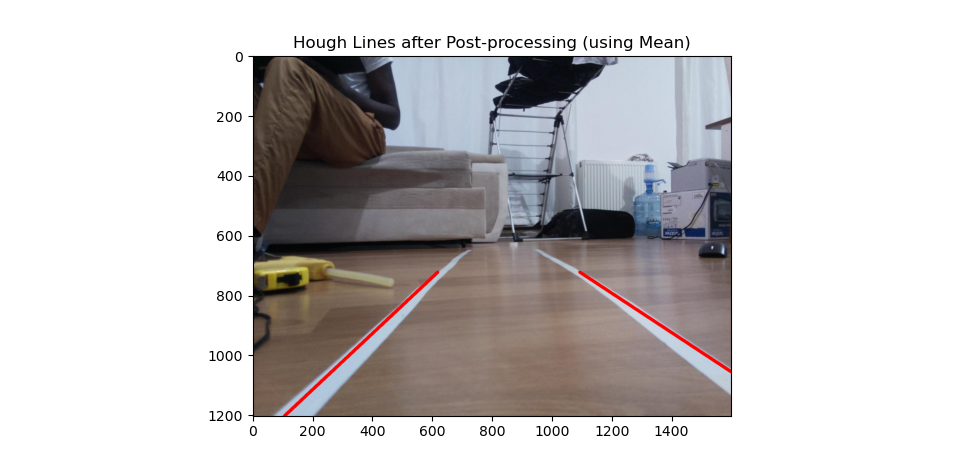
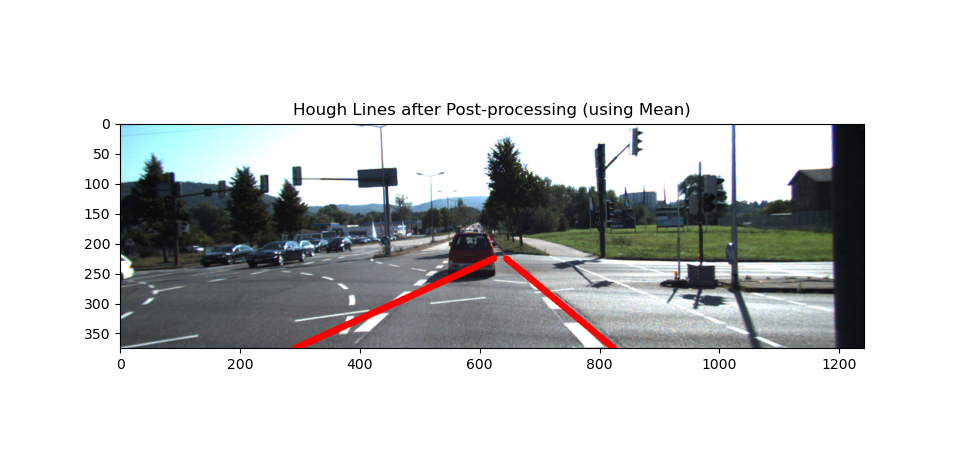


Fig. 13a & b., Detected Lanes after Post-processing

**Performance of Algorithm on Different Datasets**

Using the candidate image from the KITTI Road Dataset (from Fig. 2),



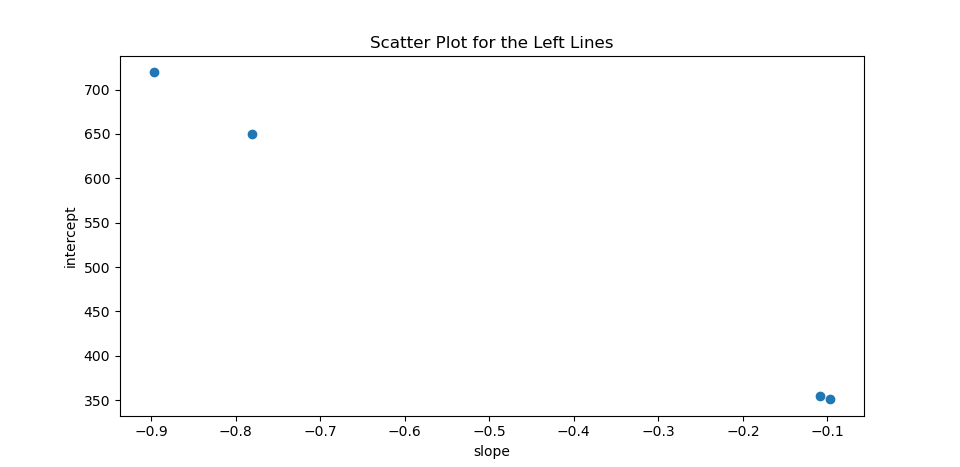


Fig. 14a, b & c., Results from the KITTI Road Dataset

This shows the loopholes in the algorithm. As we can see in above, there are four detected lines (two of which are clearly not our desired lines). Just by looking at the scatter plot, it is impossible to tell which lines are the outliers. Hence, the mean lines chosen for the final output are not what the result we want (especially for the left lane).

The lesson here is that the more the detected lines, the better the statistical elimination process and final result. Here, we can explore two plausible solutions:

1. Alter the criteria for the lines detected: this implies that we adjust the parameters for the probabilistic Hough Transform function. A more acceptable result is shown below.

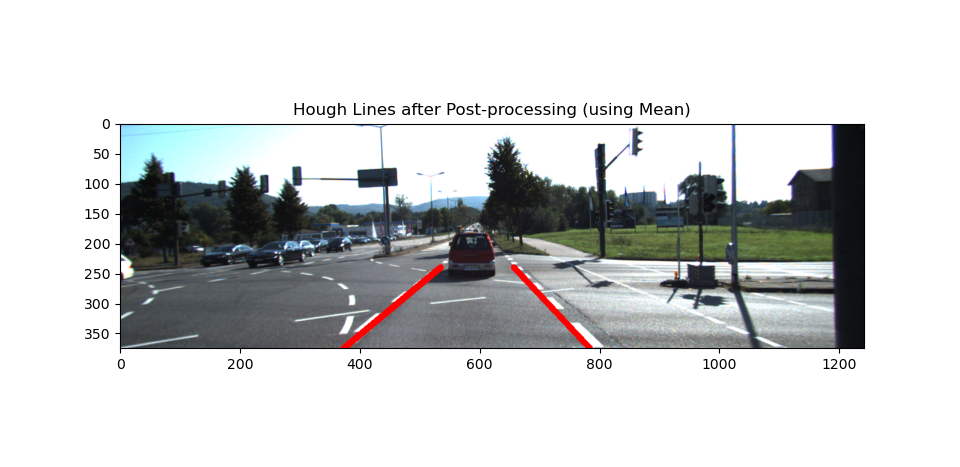
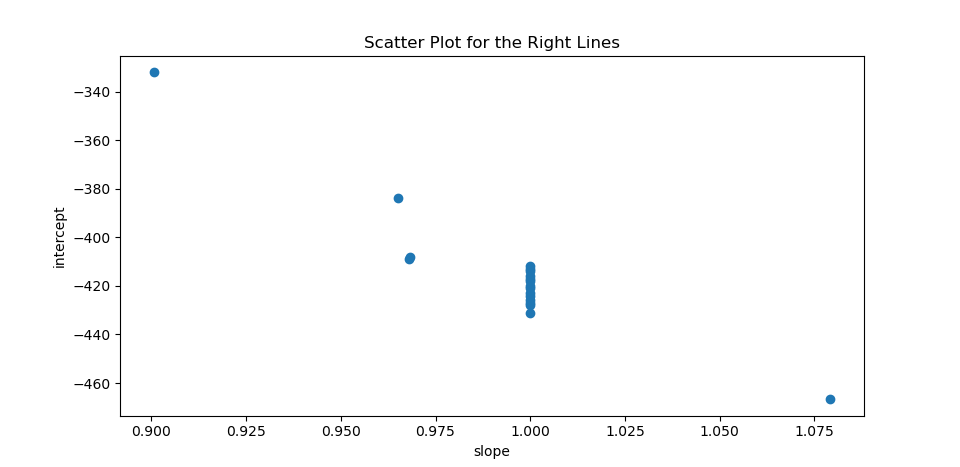
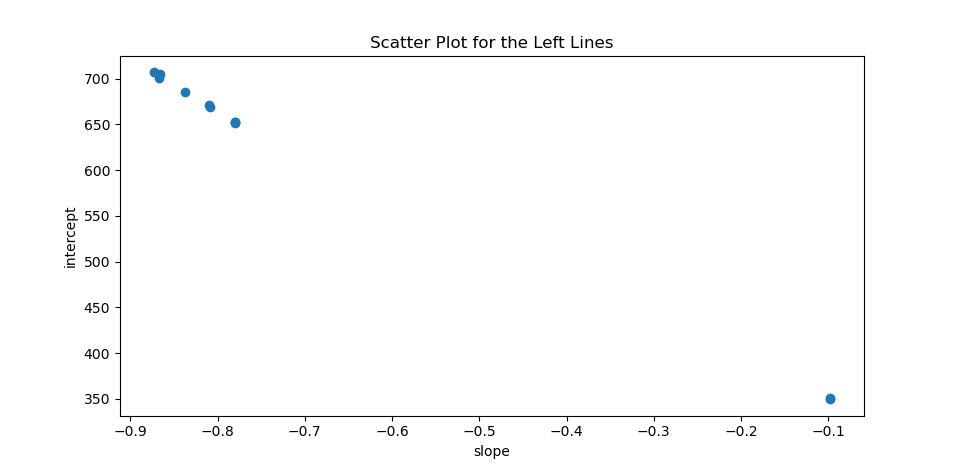


Fig. 15.

1. Replace the edge detection step with simple image thresholding. This increases the number of Hough lines detected (Fig. 16a.); hence, making it more difficult to skew the results (Fig. 16d.). Observe that the outliers are easily identifiable from the scatter plots.





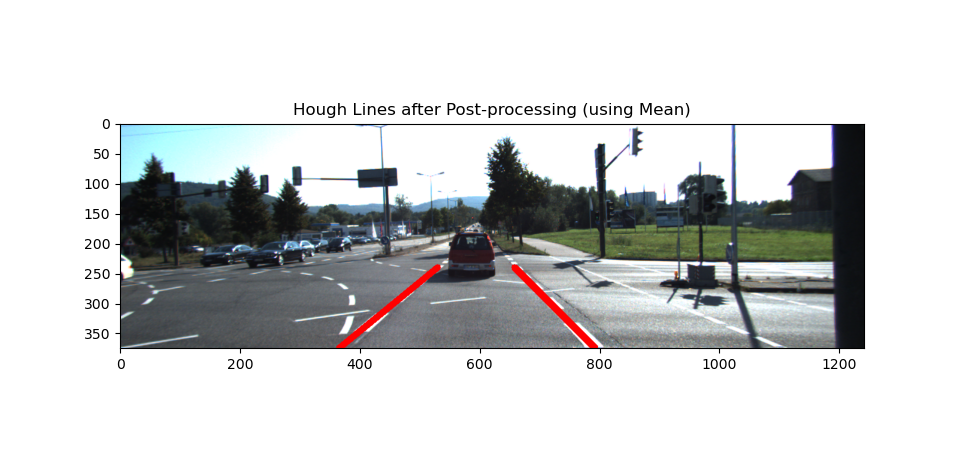


Fig. 16a, b&c, d., All Hough Lines; Scatter plots; Final Lanes

1. [Cicchino JB. Effects of lane departure warning on police-reported crash rates. J Safety Res. 2018;66: 61–70.](http://paperpile.com/b/faT9wA/2WIW)

2. [Automated vehicles for safety. In: NHTSA [Internet]. [cited 24 Dec 2021]. Available:](http://paperpile.com/b/faT9wA/uZIHs) <https://www.nhtsa.gov/technology-innovation/automated-vehicles-safety>

3. [The 6 levels of vehicle autonomy explained. [cited 24 Dec 2021]. Available:](http://paperpile.com/b/faT9wA/4DVPK) <https://www.synopsys.com/automotive/autonomous-driving-levels.html>

4. [Meier K, Chung S-J, Hutchinson S. Visual‐inertial curve simultaneous localization and mapping: Creating a sparse structured world without feature points. J field robot. 2018;35: 516–544.](http://paperpile.com/b/faT9wA/ucQaA)

5. [Li X, Duan C, Zhi Y, Yin P, Wang N. Vehicle pose estimation method for Lane line detection through monocular camera. IOP Conf Ser Earth Environ Sci. 2021;769: 042012.](http://paperpile.com/b/faT9wA/zfXp)

6. [Waykole S, Shiwakoti N, Stasinopoulos P. Review on Lane Detection and Tracking Algorithms of Advanced Driver Assistance System. Sustain Sci Pract Policy. 2021;13: 11417.](http://paperpile.com/b/faT9wA/St69)

7. [Jayasinghe O, Anhettigama D, Hemachandra S, Kariyawasam S, Rodrigo R, Jayasekara P. SwiftLane: Towards Fast and Efficient Lane Detection. arXiv [cs.CV]. 2021. Available:](http://paperpile.com/b/faT9wA/tHWkK) <http://arxiv.org/abs/2110.11779>

8. [Zaarane A, Slimani I, Al Okaishi W, Atouf I, Hamdoun A. Distance measurement system for autonomous vehicles using stereo camera. Array. 2020;5: 100016.](http://paperpile.com/b/faT9wA/xo2z)

9. [Jiangwei C, Lisheng J, Lie G, Libibing, Rongben W. Study on method of detecting preceding vehicle based on monocular camera. IEEE Intelligent Vehicles Symposium, 2004. 2004. pp. 750–755.](http://paperpile.com/b/faT9wA/Qnyd)

10. [Liu S, Tang J, Zhang Z, Gaudiot J-L. Computer Architectures for Autonomous Driving. Computer . 2017;50: 18–25.](http://paperpile.com/b/faT9wA/C5XR)

11. [OpenCV: Hough Line Transform. [cited 1 Jan 2022]. Available:](http://paperpile.com/b/faT9wA/bQ89) <https://docs.opencv.org/3.4/d9/db0/tutorial_hough_lines.html>

12. [Fritsch J, Kühnl T, Geiger A. A new performance measure and evaluation benchmark for road detection algorithms. 16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013). 2013. pp. 1693–1700.](http://paperpile.com/b/faT9wA/5rti)

13. [Hough PVC. Machine analysis of bubble chamber pictures. Proc of the International Conference on High Energy Accelerators and Instrumentation, Sept 1959. 1959. pp. 554–556.](http://paperpile.com/b/faT9wA/XJlZ)

14. [Pan X, Shi J, Luo P, Wang X, Tang X. Spatial as deep: Spatial cnn for traffic scene understanding. Thirty-Second AAAI Conference on Artificial Intelligence. 2018. Available:](http://paperpile.com/b/faT9wA/SOPt) <https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/download/16802/16322>

15. [Wikipedia contributors. Hough transform. In: Wikipedia, The Free Encyclopedia [Internet]. 20 Oct 2021. Available:](http://paperpile.com/b/faT9wA/rT1D) <https://en.wikipedia.org/w/index.php?title=Hough_transform&oldid=1050916528>

16. [Duda RO, Hart PE. Use of the Hough transformation to detect lines and curves in pictures. Commun ACM. 1972. Available:](http://paperpile.com/b/faT9wA/aJcY) <https://dl.acm.org/doi/abs/10.1145/361237.361242>

17. [Canny J. A computational approach to edge detection. IEEE Trans Pattern Anal Mach Intell. 1986;8: 679–698.](http://paperpile.com/b/faT9wA/CcCZ)

18. [pandas. [cited 2 Jan 2022]. Available:](http://paperpile.com/b/faT9wA/jhAE) <https://pandas.pydata.org/>

19. [Are the skewness and kurtosis useful statistics? In: BPI Consulting [Internet]. 1 Apr 2008 [cited 2 Jan 2022]. Available:](http://paperpile.com/b/faT9wA/F5g6) <https://www.spcforexcel.com/knowledge/basic-statistics/are-skewness-and-kurtosis-useful-statistics>