Computer Vision in Autonomous Vehicles

Jai Mehta

Department of Computer Science and Engineering, Sardar Vallabhbhai Patel,

National Institute of Technology,

Surat, Gujarat, India

p24ds017@coed.svnit.ac.in

Ayush Panchal

Department of Computer Science and Engineering, Sardar Vallabhbhai Patel,

National Institute of Technology,

Surat, Gujarat, India

p24ds013@coed.svnit.ac.in

***Abstract****:* **Autonomous vehicles represent a transformative advancement in modern transportation, combining cutting-edge technologies to enhance safety, efficiency, and convenience. This paper, titled *"Computer Vision in Autonomous Vehicles,"* focuses on the critical components of lane detection and car detection, which are essential for enabling autonomous navigation and collision avoidance. Utilizing computer vision techniques, we explore robust methodologies for lane detection, leveraging edge detection, Hough Transform. For Car Detection we used Histogram of Oriented Gradients (HoG), Heatmaps and Linear SVM model. This research provides a foundation for future advancements in computer vision applications for autonomous driving, contributing to the development of safer and more intelligent transportation systems.**

1. INTRODUCTION

Lane detection involves identifying white or similar markings on roads using image processing techniques, typically with a front-mounted camera in vehicles. It is a key component of vision-based driver assistance systems, designed to prevent accidents caused by unintended lane departures. Structured roads, marked with lane lines, facilitate easier detection compared to non-structured roads, which lack markings. Lane detection faces challenges such as varying lighting conditions, shadows, road textures, weather conditions, and obstructions. While previous methods focused on detecting obstacles and evaluating edge detection performance, modern approaches aim to address these complexities to enhance reliability in diverse driving environments.

As urban populations and transportation systems grow rapidly, managing traffic, accidents, and related challenges has become increasingly complex. Traditional methods like traffic lights and signs are often insufficient on their own, necessitating the adoption of advanced technologies. Intelligent Transport Systems (ITS), which leverage object detection and tracking, have emerged as a key solution to improve traffic flow, prevent accidents, and enforce traffic regulations. ITS components such as vehicle and lane detection rely on machine learning techniques, including Support Vector Machines (SVM) and Decision Trees, to identify and classify vehicles effectively.

This paper explores lane detection and car detection using various Computer Vision and Image Processing techniques. Noise reduction is implemented using Gaussian filters to preprocess images and enhance detection accuracy. Canny Edge Detection is employed for identifying edges, which is a critical step in detecting lane boundaries, followed by the application of Hough Transformation to detect and highlight straight lines corresponding to lane markings. For car detection, Histogram of Oriented Gradients (HoG) is used to extract robust features that capture object shapes and edges effectively. Heatmaps are employed to localize regions of interest, further improving detection precision for vehicles. Finally, Linear SVM is utilized for classification, enabling accurate distinction between vehicles and non-vehicle entities. These techniques collectively address challenges such as varying lighting, road textures, and occlusions, thereby enhancing the overall reliability and accuracy of lane and car detection systems.

1. METHODOLOGY

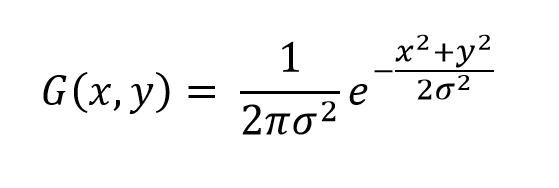
The lane detection system used in this paper follows a structured pipeline approach, especially designed for processing video data. Since videos are sequences of images (frames) displayed rapidly, each frame is processed sequentially to extract the lane markings. The processing steps are as follows:

The first step in the pipeline is the conversion of the input image to grayscale. This simplifies the image by removing colour information and retaining only intensity variations, which is crucial for the subsequent edge detection process. The grayscale conversion is achieved using a weighted sum of the red, green, and blue channels of the image, according to the formula:

Igray= ​=f(x1)⋅R + f(x2)⋅G + f(x3)⋅B

where R, G, and B are the red, green, and blue channels, respectively. This transformation reduces the computational complexity while retaining the necessary details for lane detection.

Next, Gaussian blur is applied to the grayscale image to reduce noise and smooth out the image. The Gaussian filter is defined by the equation:



where σ determines the amount of smoothing. This step is vital for eliminating high-frequency noise and enhancing the edges that correspond to the lane markings.

Following the blurring step, Canny edge detection is applied to identify the edges in the image. The Canny algorithm works by first calculating the gradients in both the x and y directions using Sobel filters, then determining the edge magnitude G and direction θ.

After gradient calculation, non-maximum suppression is used to thin out the edges, and double thresholding is applied to classify the edges into strong, weak, or non-edges. This process highlights the most prominent lane markings.

To focus on the area of interest where the lanes are likely to be, a region of interest (ROI) mask is applied. The ROI is defined by a polygon formed by four points, corresponding to the bottom-left, bottom-right, top-right, and top-left corners of the expected lane area. By masking the irrelevant parts of the image, this step reduces computational complexity and ensures that the algorithm focuses only on the portion of the image that contains the lanes.

Once the edges are detected and the region of interest is masked, the Hough Line Transform is applied to detect straight lines in the image. This technique works by representing each straight line in the image using polar coordinates (ρ, θ), where ρ is the distance from the origin to the line, and θ is the angle of the line. The Hough Transform detects peaks in this coordinate space, corresponding to the prominent lines in the image. Lines are filtered based on their length and gap, ensuring that only the relevant lane markings are detected.

After detecting the lines, they are classified based on their slopes. Lines with positive slopes are categorized as right lane markings, while lines with negative slopes are classified as left lane markings. This step ensures that the algorithm can detect lane markings on both sides of the road.

To further improve the accuracy of lane detection, outliers are removed. This is done by filtering out lines that fall outside the expected slope range for each side of the road. This step eliminates any erroneous lines that may be caused by noise or other factors unrelated to the lane markings.

Once the lines are filtered, the remaining lines are merged and extended. The lines are averaged to form a single line for each lane, and the endpoints of the lines are extrapolated to extend the lane markings across the entire image. This is particularly useful when lane markings are partially occluded or when the lines are not fully visible in the frame.

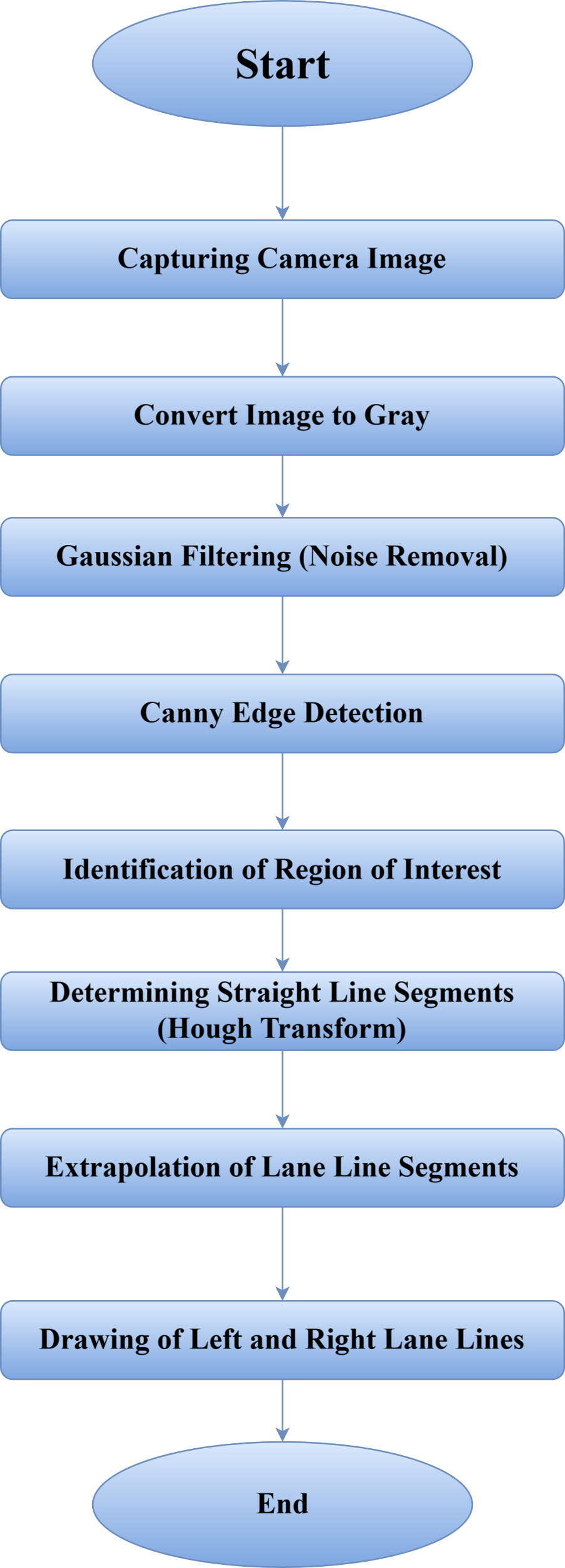


Figure 1 Pipeline Architecture for lane Detection

The extrapolation ensures that the lanes are consistently represented across the entire road.

Finally, the detected lane lines are drawn onto the original image. The left and right lane markings are typically drawn in different colours to distinguish them clearly. These lane lines are then overlaid onto the original image using a weighted sum, resulting in the final output where the lane markings are clearly visible. The processed image is then returned as the result of the pipeline.



Figure 2 Lane Detection

To sum up for lane detection, the above methodology provides an effective approach using image processing techniques. By employing Gaussian blur, Canny edge detection, Hough Line Transform, and linear extrapolation, the system is capable of detecting and tracking lane markings with high accuracy, even in challenging conditions such as partial occlusions and varying road textures.

For Car Detection, we used a combination of Histogram of Oriented Gradients (HOG) feature extraction and sliding window techniques for vehicle detection within video streams. The process involved several stages including feature extraction, classifier training, vehicle detection, heatmap generation, and tracking.

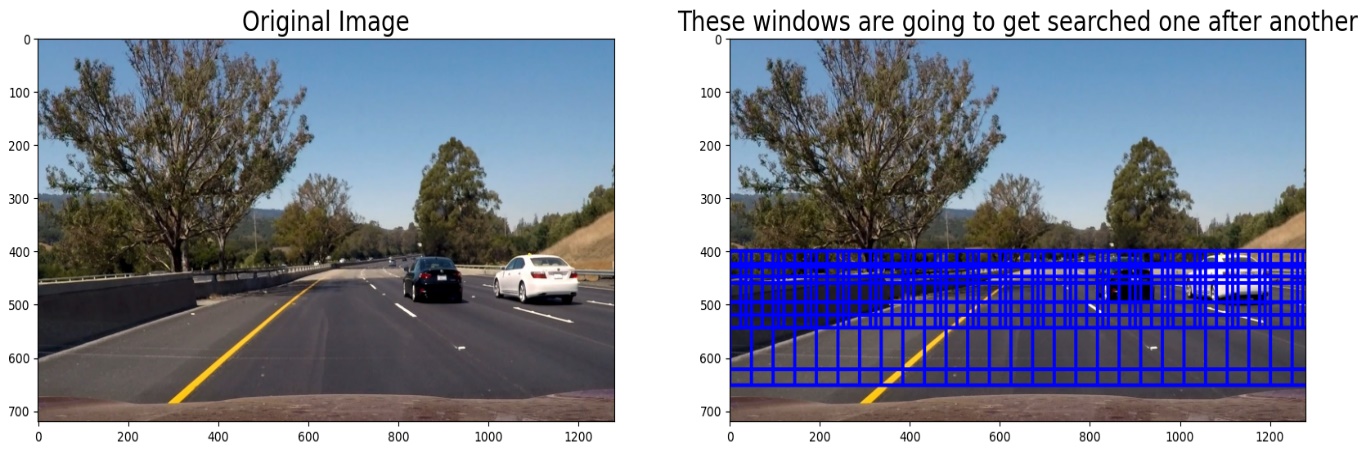
The vehicle detection system employed Histogram of Oriented Gradients (HOG) feature extraction and a sliding window technique, applied to video frames. Initially, HOG features were extracted from images using all three-color channels (YCrCb) to improve selectivity. Optimal parameters for HOG were determined to be 32 orientations, 16 pixels per cell, and using all three channels. These features were used to train a Linear Support Vector Machine (SVM) classifier to distinguish between vehicles and non-vehicles. We have used the [GTI Vehicle Image Database](http://www.gti.ssr.upm.es/data/Vehicle_database.html) and Udacity Extras dataset to extract the car features using HoG.

Figure 3 Sliding Window Approach for Detecting Cars

Hog sub-sampling is more efficient method for doing the sliding window approach as it

computes individual channel HOG features for each entire image and uses the scaled features to make prediction. Each window is defined by a scaling factor where a scale of 1 would result in a window that’s 8 x 8 cells then the overlap of each window is in terms of the cell distance; this means that a cells\_per\_step = 2 would result in a search window overlap of 75% (Dimililer et al., 2020)[2].

The hog sub-sampling helps to reduce calculation time for finding HOG features and thus provided higher throughput rate (Dimililer et al., 2020) [2]. Hog sub-sampling was used for implementing the sliding window approach with starting position of the window search from 350px to 656px and cells\_per\_step to 1 as mentioned in Dimililer et al. (2020) [2] and in Karunakaran (2017) [4].

To reduce false positives, a heatmap was generated by accumulating detection counts across frames. A threshold was applied to retain only those boxes with sufficient counts, and temporal consistency was used to track vehicles across multiple frames. The final step involved drawing bounding boxes around detected vehicles, providing real-time detection results.

1. RESULTS

In the HOG feature descriptor, the distribution of directions of oriented gradients are used as features. Fig 4 shows A vehicle image was randomly selected from the positive dataset and is shown on the right with its extracted hog feature.

A randomly selected non-vehicle image was taken from the negative dataset and is shown on the left with its hog feature.



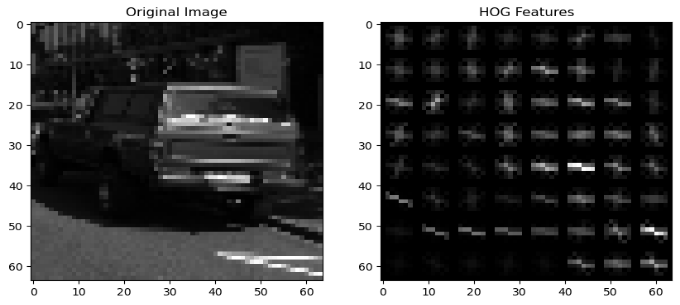


Figure 4 Extracted Features using HoG for Vehicle and No-Vehicle images of GTI Vehicle Image Database and Udacity Extras dataset

The score obtained after training LinearSVC as the test accuracy was 0.9932, that is 99.32%. After running Stratified-KFold cross-validation, we have acquired values for the area under ROC curve (AUROC) for the model shown in Figure 5. We can observe that LinearSVC has a great True positive rate and a mean AUROC of 0.99 ± 0.01.



Figure ROC curves for LinearSVC. The blue line indicates the average

The results of heat map and thresholding for the sliding window technique after combining together is faster with hog sub-sampling. Figure 6 shows the cars found using hog subsampling search window with its heat map after thresholding. We can observe from the figure that heat map is black (with no heat) when there is no car on the road.

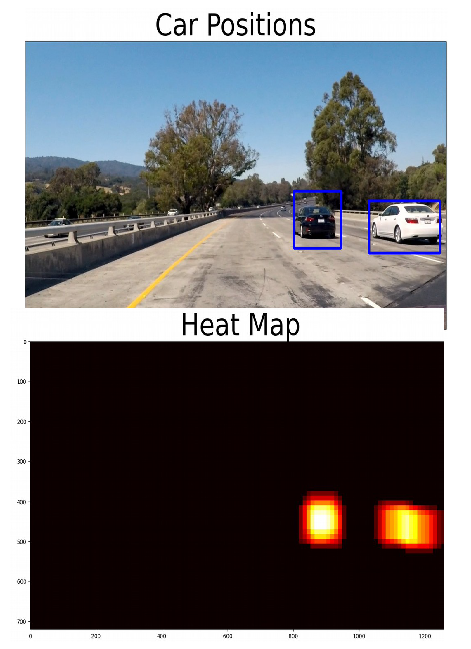
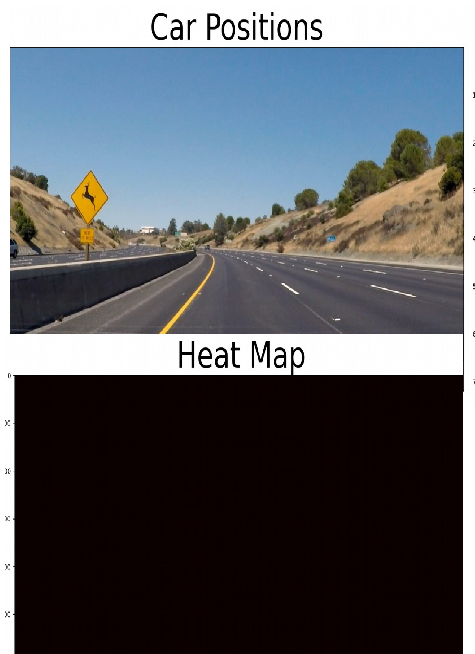


Figure Hog sub-sampling with heat map for Vehicle and No-Vehicle images.

1. CONCLUSION AND FUTURE WORK

In conclusion, this research highlights the integration of machine learning techniques, such as sliding window searches and Support Vector Machine (SVM) classifiers, in solving real-world problems related to vehicle and lane detection. The study demonstrated the practical application of these methods, showcasing their potential to address complex challenges in autonomous vehicle systems. Through the utilization of specialized datasets like the GTI

Vehicle Image Database and Udacity Extras, the research provided a robust framework for training and testing detection models. Additionally, the research emphasized the importance of evaluating model accuracy and efficiency, considering both training performance and real-time detection scenarios. These findings contribute to advancing the field of computer vision and machine learning, offering valuable insights into the development of reliable, real-time detection systems.

Vehicle Image Database and Udacity Extras, the research provided a robust framework for training and testing detection models. Additionally, the research emphasized the importance of evaluating model accuracy and efficiency, considering both training performance and real-time detection scenarios. These findings contribute to advancing the field of computer vision and machine learning, offering valuable insights into the development of reliable, real-time detection systems.

We implemented the methods discussed in Dimililer et al. (2020) [2] for vehicle detection and tracking to build a system capable of detecting and tracking cars.

Using SVM, we achieved good results; however, the model's performance cannot be fully generalized. Therefore, this project could be further explored using other machine learning algorithms to enhance robustness and accuracy.

Lane Detection sometimes gives poor output when there exist curvy lanes. This needs to be addressed by creating a color mask to highlight

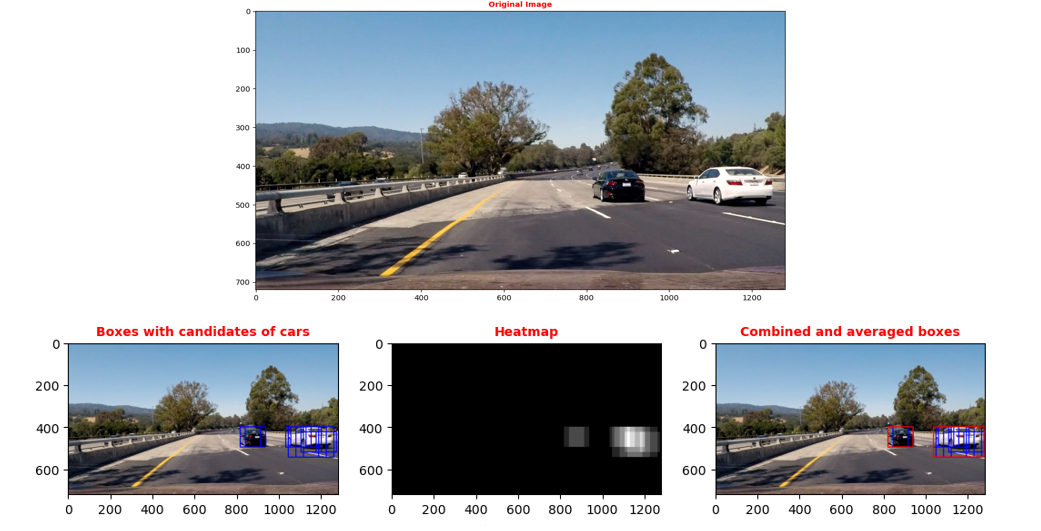


Figure 7 Use of Heatmaps to find out the Region of Interest

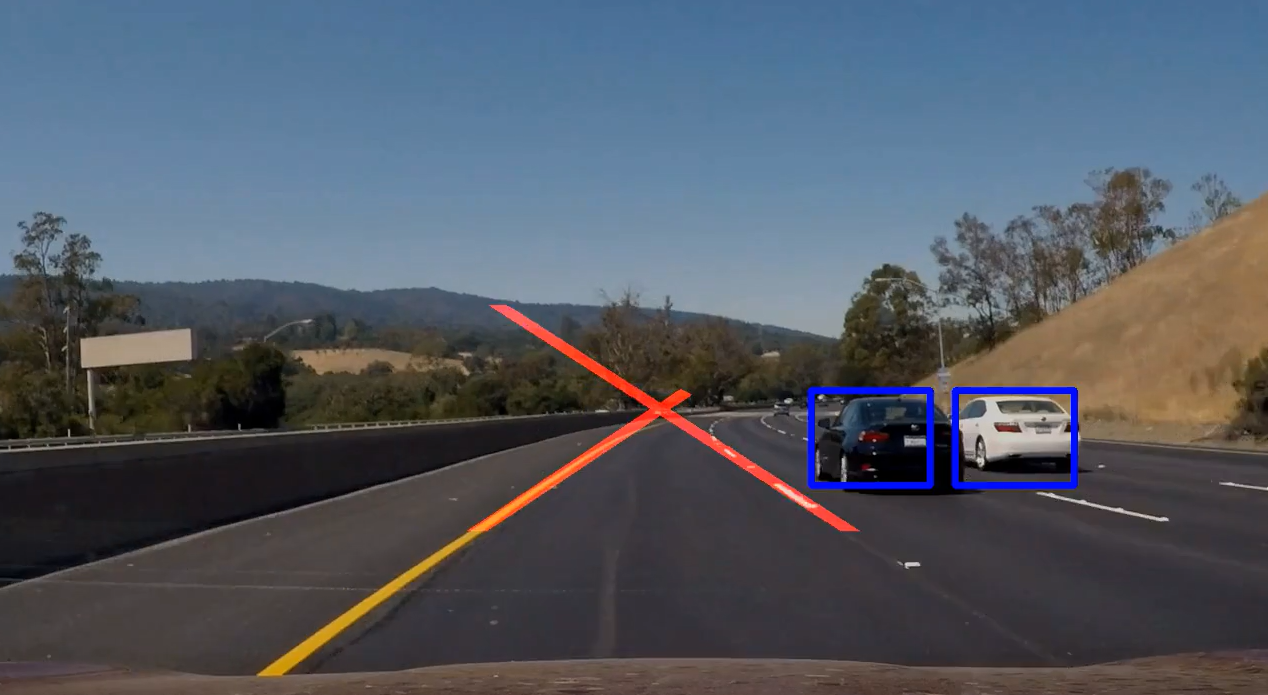


Figure 8 Lane Detection and Care Detection together

It took 29.41 Seconds to train SVC...

Test Accuracy of SVC = 0.9932

The system successfully detects and tracks lane lines and vehicles with a high degree of accuracy in the provided test videos.

the whites and yellow might help and use of polynomial line detection.

As part of future work, this system can be extended to estimate the depth of objects on the road, as well as their speed and distance, to provide a more comprehensive understanding of the driving environment.

1. REFERENCES

[1] Farag, Wael & Saleh, Zakaria. (2018). Road Lane-Lines Detection in Real-Time for Advanced Driving Assistance Systems. 1-8. 10.1109/3ICT.2018.8855797.

[2] Dimililer, K., Ever, Y.K., Mustafa, S.M. (2020). Vehicle Detection and Tracking Using Machine Learning Techniques. In: Aliev, R., Kacprzyk, J., Pedrycz, W., Jamshidi, M., Babanli, M., Sadikoglu, F. (eds) 10th International Conference on Theory and Application of Soft Computing, Computing with Words and Perceptions - ICSCCW-2019. ICSCCW 2019. Advances in Intelligent Systems and Computing, vol 1095. Springer, Cham. <https://doi.org/10.1007/978-3-030-35249-3_48>

[3] S. K. Vishwakarma, Akash and D. S. Yadav, "Analysis of lane detection techniques using openCV," 2015 Annual IEEE India Conference (INDICON), 2015, pp. 1-4, doi: 10.1109/INDICON.2015.7443166.

[4] Karunakaran, D. (2017, December 10). *Vehicle Detection and Tracking: Udacity’s Selfdriving Car Nanodegree*. Medium.

<https://medium.com/intro-to-artificial-intelligence/vehicle-detection-and-trackingudacitys-self-driving-car-nanodegree-ca02330820ee>

[5] Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*(3),

273–297.

[6] Bradski, G. (2000). The OpenCV Library. *Dr. Dobb&#x27;s* Journal of Software Tools.