Final Report - Lane Detection

By Diego Freire

Course: CSC 83020 - Human & Computer Vision with Advanced Topics

Instructor: Prof. Zhingang Zhu

Institution: CUNY Graduate Center

Program: MS in Data Science

May 2023

Contents

Problem Statement

Driving requires maintaining lane position, which can be difficult, especially at high speeds, in curves or in adverse weather conditions. As a safety measure, many modern cars are equipped with driving assistance technologies to make the driving experience safer for all drivers. Within these technologies is the Lane Departure Warning System (LDWS), a mechanism used to warn drivers when the vehicle is moving out of a lane on highways. These systems are designed to reduce the number of accidents addressing the main causes of collisions like driver error or distractions. [1]

An important piece of LDWS is Lane Detection Systems, where Computer Vision techniques are applied to identify the lines on the road and alert the driver when the car is leaving a lane. Lane detection algorithms analyze images or video frames captured by cameras mounted on the vehicle and determine the location of the lane markings.[2]

Lane detection is an important technology for several reasons, particularly in the context of advanced driver assistance systems (ADAS), and it is a critical component of Autonomous Driving as it provides the vehicle with the ability to navigate the road and stay within the correct lane. Without accurate and reliable lane detection, it would be difficult for autonomous vehicles to operate safely and effectively on public roads. [3]

Overall, lane detection is an important technology that plays a key role in improving safety, enhancing the driving experience, and enabling the development of autonomous driving. This project intends to Explore the Computer Vision techniques utilized in the development of Lane Detection Systems, developing a program capable of accurately detecting and tracking lane markings in video footage. This will serve to enhance our understanding of Computer Vision and its practical applications in the field of advanced driver assistance systems.

Related work

Lane detection is an important and active research topic in the fields of computer vision and machine learning [4]. Over the years, various approaches have been developed for detecting lanes, including traditional computer vision techniques, machine learning-based methods, and deep learning-based methods. These approaches differ in terms of their accuracy, efficiency, and complexity, and each has its own advantages and disadvantages. As a result, researchers and developers continue to explore new techniques and approaches that can improve lane detection performance and robustness. [5]

Many algorithms have been proposed, using various techniques such as edge detection, color segmentation, and Hough transform. The most advanced approaches use Deep Learning models, and they are capable of detecting objects as well.[3]

Traditional Computer Vision Methods:

These methods are typically based on image processing algorithms that extract features from images to detect edges. They operate on grayscale images using edge detection operators and region-of-interest (ROI) masks to isolate the lane markings. Common techniques for detecting lane edges include Canny edge detection and Hough transform-based methods. While these methods are computationally efficient and simple to implement, they can be sensitive to changes in lighting conditions and may not perform well in different driving scenarios.[3]

Yoo et al. proposed an approach that enhances gradient-based conversion for robust lane detection. Their method involves converting RGB space images into grayscale images and utilizing Canny's algorithm to generate gradients at the boundaries.[4] On the other hand, Son et al. attempted to detect lanes by extracting white and yellow lane lines using a different approach.[5]

In the paper "Lane Detection Techniques using Image Processing" by Vignesh et al., they explore two distinct approaches for detecting lanes on unmarked roads using digital image processing techniques. Their objective is to obtain real-time curve values that can aid drivers or autonomous vehicles in making necessary turns and staying on the road. These approaches involve common steps, including thresholding, warping, and defining a region of interest (ROI). Additionally, they utilize various processing techniques such as pixel summation (histogram analysis), Gaussian blur, image dilation, Canny edge detection, and the sliding window algorithm. The project will follow a similar approach to this work.

Machine Learning and Deep Learning Methods:

Machine learning-based methods use supervised learning algorithms to train models to detect lane markings in images or video streams. These methods typically involve feature extraction, such as Histogram Oriented Gradients (HOG), followed by classification using machine learning algorithms like SVM or neural networks.

Deep learning-based methods use convolutional neural networks (CNNs) to automatically learn features from images or video streams and predict the position of the lane markings. These methods have shown state-of-the-art performance on lane detection tasks and have been widely adopted in the industry.

Machine Learning and Deep Learning methods have improved and have shown significant improvements in accuracy compared to traditional methods, but they require enormous amounts of training data and can be computationally expensive. [5]

Convolutional Neural Networks (CNN) have the capability to extract robust features from road images, allowing for accurate lane detection. These extracted features can be used to train an additional tree-like regression model, which directly estimates the lane line positions based on the learned features, enhancing the effectiveness of the lane detection system. [5]

Hybrid Approaches:

Hybrid methods for lane detection combine traditional computer vision techniques with machine learning or deep learning approaches to improve accuracy and efficiency. These methods extract features from images using computer vision algorithms such as Canny edge detection and Hough transform, and then use machine learning or deep learning algorithms to classify these features as belonging to a lane [6].

By combining the strengths of both approaches, hybrid methods can improve the robustness and accuracy of lane detection systems, particularly in challenging road conditions such as low light and severe weather. However, developing hybrid methods can be challenging due to the need for expertise in both computer vision and machine learning, as well as the complexity of integrating these techniques effectively.

Almotairi shows a comparison of different approaches using a combination of filters, highlighting the advantages and disadvantages of each of them. The Canny Edge, Hough Transform and Sobel Edge are widely applied for Lane Detection, along with the Bird’s eye view which can work well in complex situations. However, the most common disadvantage is that they do not perform well with degraded lane markings. [6]

Proposed Approach

A Lane Detection System used behind the Lane Departure Warning System (LDWS) uses the principle of Hough transform and Canny Edge detection to detect lane lines from real time camera images fed from the front-end camera of the automobile. [1]

My Proposed approach uses python and the OpenCV library to process images from video frames. Here is a diagram of the process.

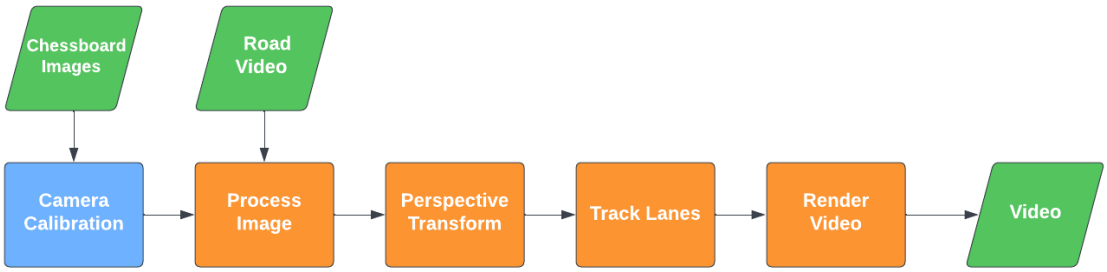

Figure 1 Lane Detection Process Diagram

Use the Open CV library to do the following:

- Camera Calibration using Chessboard Images to find the intrinsic and extrinsic parameters.
- Use the parameters to remove distortion from the road scene images.
- Process the images using filters to enhance lane markings and reduce noise.
    - Use Hue Light Saturation to highlight yellow and white lines.
    - Apply Sobel Filters to find the edges.
    - Combine images and obtain a binary image.
- Apply a perspective transform to the binary image to get the Birds Eye View.
- Use a histogram to find the variations of white pixels in the distribution.
- Track the lanes using a sliding window technique.
- Project the lane in the view.
- Add information to the video (extra)
- Render video

During the presentation of the project, I was suggested that the camera calibration was not necessary in the process, or that the parameters should be applied along with the perspective transform. For this experiment I will do the following changes to the original process:

- Process the images using filters to enhance lane markings and reduce noise.
    - Use Hue Light Saturation to highlight yellow and white lines.
    - Apply Sobel Filters to find the edges.
    - Combine images and obtain a binary image.
- Apply a perspective transform to the binary image to get the Birds Eye View.
- Use a histogram to find the variations of white pixels in the distribution.
- Track the lanes using a sliding window technique.
- Project the lane in the view.
- Render video

Results and Analysis

Before working directly with the video, I conducted preliminary tests screenshots extracted from the footage. The lane detection process implemented in this project draws inspiration from the work of Haque et al. in their paper titled "A computer vision-based lane detection approach." Their research provided valuable insights for the development of this lane detection algorithm.

1) Camera Calibration

Camera calibration is the first step of the process because it allows to remove distortion from the images. It is in practical terms a good method to remove distortion from images that might change the shape and size of the lane, vehicle, and background in the road scene. These changes are not conducive to judging the correct driving direction and determining the exact location of the vehicle. Consequently, converting image distortion is essential. [4]

Camera calibration using OpenCV is a well-documented and straightforward process, with examples available on the official documentation. Following the recommendation from Haque's work, I used the chessboard calibration to calculate the intrinsic and extrinsic parameters of the camera, which are used to align and remove distortion from the images of the road scene.

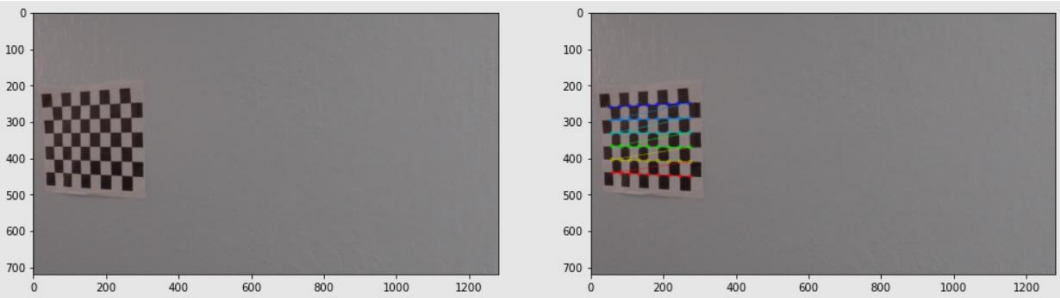

Figure 2 Finding Corner Points

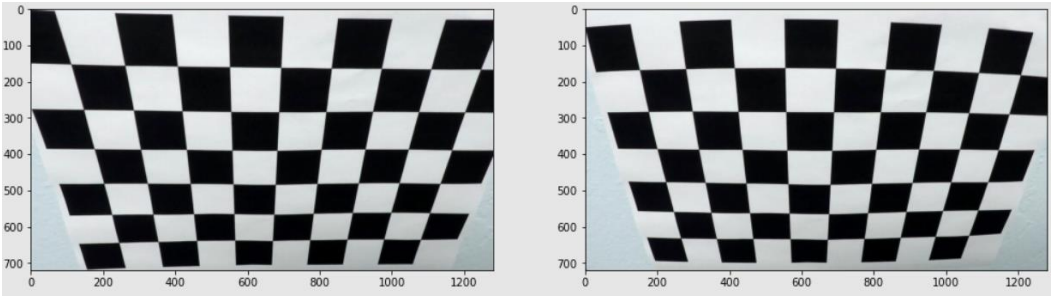

Figure 3 Camera Calibration result

2) Remove Distortion

After obtaining the camera parameters, I proceeded to apply them to the road scene images. Below is a side-by-side comparison displaying the original images on the left and the corrected images on the right, following the application of the calculated parameters. It is noticeable that the corrected images exhibit some cropping and rotation. For instance, in the image featuring two cars, one of the white car's lights is partially cut off from the scene. Additionally, the car's heads-up display (HUD) is almost concealed in the corrected version.

Figure 4 Applying Camera Calibration Parameters to the Road Scene Images

3) Process the images.

In an initial experiment, I implemented Canny Edge detection from OpenCV. However, the resulting image displaying the detected edges lacked significant details from the original image. Edges from the border of the road, trees, cars on the side are visible but the lane lines are lost. For example, the yellow line on the left side of the road is invisible in the edge detection output.

Figure 5 Built-in Canny Edge Detection

Edge Detection

I adopted an alternative approach by employing a combination of functions, following these sequential steps:

1. Extracted the Hue, Lightness, and Saturation channels with a focus on yellow and white colors.
2. Applied the Sobel filter to identify vertical and horizontal edges.
3. Determined the magnitude and direction of the Sobel gradients.
4. Combined the resulting binary matrices into a final binary output.

The outcome of this approach surpassed the results obtained solely by using the Canny function from OpenCV. Nevertheless, upon closer inspection of the yellow line along the road, it appears slightly blurred, indicating the presence of noise in the image.

Figure 6 First Experiment with Sobel Filters

I repeated the same process on an image that focused on a region with noticeable tree shadows. It became evident that the shadows produced additional edges, which were captured by the applied process. This observation raised my concerns for subsequent steps in the overall process, as the presence of shadow-induced edges could potentially pose challenges or issues.

Figure 7 Edges produced by Shadows.

Through iterative adjustments to the kernels and parameter values, I managed to rectify the process and discover an optimal combination. As a result, the following image was generated. Notably, the edges of the lane markings exhibit enhanced clarity compared to the edges of other objects present in the road scene. The primary objective of this process was to emphasize the lines on the road, with particular emphasis on highlighting yellow and white markings.

Figure 8 Result of Selected Combination of Filters in curved road.

To verify the effectiveness of the filters in a different scenario, I applied them to a different road scene image. Just like in the previous image, the filters successfully highlight the edges of the lane markings,

making them more distinguishable compared to other objects present in the environment. The outcome across different road scenes indicates the robustness and reliability of the filters in enhancing lane visibility.

Figure 9 Result of Selected Combination of Filters in straight line.

4) Perspective Transform

Perspective Transform is a computer vision technique used to convert an image from one perspective to another, allowing a change in the viewpoint or projection of the image. In lane detection, this is applied to a Region of Interest (ROI) to obtain a Top-Down or Bird's eye view.[4]

In this step, I utilized a sliding window approach to define a specific area for lane detection in the image. Then, I applied a perspective transform to an area of the image obtaining a bird's-eye view image, providing a top-down perspective of the road. This transformation simplifies the task of detecting and tracking lane markings since they now appear as straight lines rather than curved.

The resulting transformed image not only facilitates lane detection but also enables precise measurements by providing an accurate representation of distances between lines on the road.

Figure 10 Bird's Eye View

5) Histogram

Extracting the histogram of the bird's-eye view image is a key step in the process because it helps to obtain insights for lane detection, such as lane separation, position estimation, adaptability to lighting variations, real-time processing, and lane width estimation. This information is valuable for tasks involving vehicle control and autonomous navigation. [7]

Continuing with the process, after obtaining the histogram of the bird's-eye view image, I proceeded to analyze its distribution, which plays a crucial role in lane detection. By examining this distribution, I was able to identify areas with a higher count of white pixels, showing the presence of lane markings. This step allowed an accurate detection and localization of the lanes in the image.

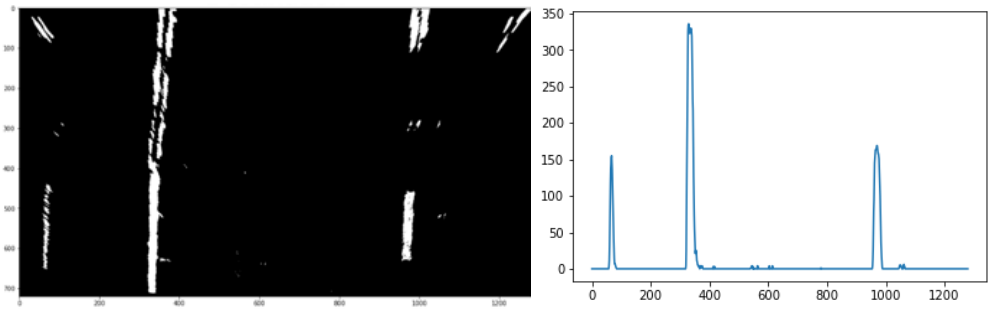

Figure 11 Histogram of Bird's Eye View

Analysing the histogram allowed me to identify the presence of white pixels in the bird's-eye view image. Comparing the image with its corresponding histogram, it is evident that the white straight line exhibits a higher peak due to its continuous nature. In contrast, the broken lines appear with smaller peaks in the histogram. It is important to note that these peaks may vary as the car moves along the lanes, indicating changes in the lane position or alignment.

6) Track the lanes

Following Haque's work, I implemented the Sliding Window technique to acqurately track lanes. This technique involves dividing the image into smaller regions or windows and analyzing them individually.

In the case of lane detection, the Sliding Window technique plays a crucial role in identifying and tracking the lane lines. To begin with, I obtained a histogram of the bird's-eye view image, which provided the pixel distribution along the horizontal axis. By analyzing the histogram, I was able to determine the initial positions of the left and right lane lines.

Starting from the bottom of the image, I defined a starting point and drew rectangular windows to represent the regions of interest for searching white pixels. To ensure comprehensive lane coverage, I used a total of 9 windows for each lane line. By looping through the windows and considering the distribution values from the histogram, I dynamically adjusted the position and size of the windows to align with the lane lines.

Simultaneously, I obtained polynomial curves that fit the lane pixels within each window. These curves represented the estimated paths of the lane lines. By fitting the polynomial curves, I could accurately track the curvature and direction of the lanes throughout the image.

The result of this process was a series of images showing the lane lines highlighted by the windows and the fitted polynomial curves. These images provide valuable insights into the lane tracking process and demonstrate the effectiveness of the Sliding Window technique in accurately detecting and tracking lane lines.

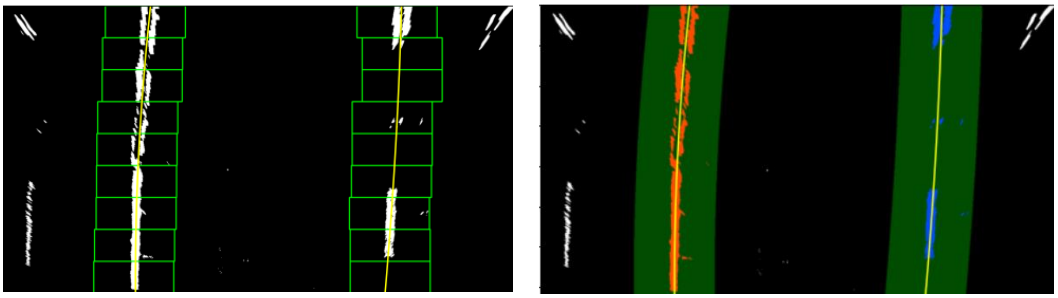

Figure 12 Tracking the Lane Lines

7) Project detected lines onto original image.

After determining the position of the lane lines, the next step involved projecting the detected lane lines onto the original road image. To achieve this, I utilized the polynomial points obtained in the previous step. These points were used to generate an image with the lane lines drawn. Using the warp perspective function, I then transformed this drawing to match the perspective of the original road image.

Finally, the warped drawing was combined with the original image, overlaying the detected lines onto the road scene, resulting in a visual representation of the lanes on the original image.

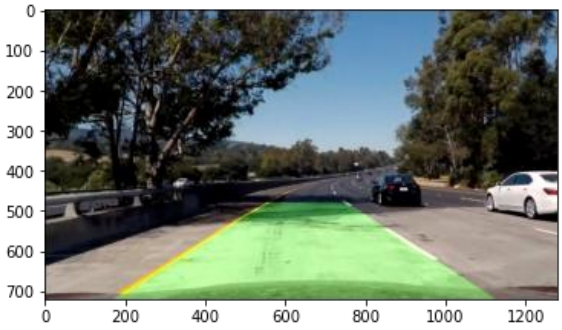

Figure 13 Projection of detected lines onto the Lane

8) Add Information to the video (Extra)

As an addition to the video, I planned to add some information to the video. I decided to include the histogram and the bird's-eye view to provide insights into the changes displayed in the images during the lane detection process as the car moved. Incorporating these elements posed a challenge as I initially intended to overlay the images with reduced opacity to create a transparent effect over the video. However, I encountered difficulties in achieving this effect. As an alternative approach, I replaced the pixels in the top right corner of the video frames, which yielded satisfactory results.

Figure 14 Adding extra information to the video.

9) Render Video

For the last step I used the moviepy videoeditor class from the VideFileClip library to render the video. This allowed me to open a video and use a function with the defined pipeline to process thirty frames per second from the project video input and output a new video with lane lines projected.

Challenges and Improvements

During the development of the program, I encountered a few challenges. The first challenge was selecting the most suitable filters and determining the optimal combination of filters and their parameters. I experimented with various filters and observed their effects on lane detection. While most filters performed well, there was a specific section in the video where a tree casts a shadow, causing the filters to detect it as edges. To address this, I had to fine-tune the filter combination to minimize the impact of the shadow while effectively highlighting the lane markings. It required a careful balance to ensure accurate lane detection while minimizing unwanted detections from environmental factors.

Figure 15 Shadows on the road

The second challenge encountered during the development process was figuring out how to implement the track lane function and accurately reproject the obtained points onto the lane image. This involved determining the appropriate calculation of windows based on the histogram data and sequentially drawing them on top of each other. It required careful consideration and experimentation to ensure the correct representation of the lane lines throughout the process.

The program works well for the experiment, but there are some necessary changes that could improve the performance and reduce the overall processing time. Currently, the program takes approximately five minutes to process and render the result for a one-minute video. To address this, it is necessary to optimize certain aspects of the process that contribute to the execution time. One potential area for improvement is the function responsible for tracking the lanes, as it involves complex calculations that might be time-consuming. Additionally, removing unnecessary additional information that does not

contribute significantly to the output can also help streamline the process. By optimizing these aspects, it is possible to significantly reduce the execution time and enhance the overall efficiency of the program.

Another issue that might be silently affecting the performance is that the program does not currently utilize GPU acceleration, which can significantly impact its running time. By incorporating GPU processing capabilities, the program could experience a notable improvement in performance. Additionally, it is worth considering the hardware limitations of the system used for running the program. In this case, a gaming laptop with seven years of usage might not possess the latest hardware advancements necessary for optimal speed and efficiency. Upgrading to a more powerful system or leveraging cloud-based computing resources could potentially mitigate the impact of hardware limitations and enhance the program's processing speed.

Alternative Process

During the project presentation, there were discussions regarding the necessity of the camera calibration step and its placement within the overall process. Some participants questioned whether camera calibration was essential or if it could be integrated into other stages, such as perspective transformation.

These inquiries prompted further exploration and consideration of alternative approaches to optimize the workflow and improve the overall effectiveness of the lane detection system.

The camera calibration step was used to calculate the camera parameters and then apply them to remove distortion from the video images.

Two additional experiments were performed to obtain results removing or changing the order of the camera calibration step.

To compare the results, two were taken from the rendered results at the same instant (00:24.71).

Figure 16 Calibrated Image

Figure 17 Uncalibrated Image

The main difference between them is that the ‘calibrated’ image is cropped after applying the parameters, resulting in a slightly narrower field of view compared to the ‘uncalibrated’ image.

At this moment, the car was traversing a section of the road that had concrete and asphalt. This caused variations in colors and light reflections, which affected the lane detection process and is evident in both videos. However, the uncalibrated image was more susceptible to noise, resulting in distortion in the lane projection. Although the process worked without camera calibration, the lane detection exhibited some difficulties that became apparent throughout the video projection. While these issues are not so evident, I personally preferred the stability of the projection achieved by applying camera calibration in the process.

Another alternative that I explored was applying the camera calibration parameters to the bird's-eye view. Initially, I anticipated that this approach might introduce a mismatch in the lane detection due to the difference in the field of view. However, to my surprise, I found that it did not significantly affect the result. Despite the variation in the field of view, the lane detection remained accurate and consistent. This observation suggests that applying the camera calibration parameters to the bird's-eye view can be a viable option without compromising the overall quality of the lane detection algorithm.

Figure 18 Applying the Camera Calibration to the Bird's Eye View

Conclusions and Discussions

- The experiment was an enjoyable experience to understand the fundaments of the underlying logic behind Lane Detection Systems.

  The implementation of the program allowed me to gain valuable insight into the principles employed for Lane Detection. It also served as an inspiration to explore other Computer Vision applications that solve real life problems.

- Removing the camera calibration step does not cause major problems in the process.

  The camera calibration as an initial step questioned if this was necessary during the process. Keeping the calibration, moving it to a different point like in the Bird’s Eye View or removing the step still makes lane detection possible. Although the lane projection looks more unstable than in the original process, it is not a major issue in the result.

- Future work could focus on improving the performance of the program in challenging scenarios, such as on roads with poor or faded markings, or in adverse weather conditions.

  Although the program currently manages shadows and changes in brightness well, there is still room for improvement to minimize any distortions in the lane projection caused by these factors.

- Additional enhancements can be implemented to optimize the execution time of the program, particularly when operating in real-time scenarios.

  Strategies such as algorithmic optimizations, parallel processing techniques, or hardware acceleration using GPUs can be explored to reduce the computational load and achieve faster lane detection. Improving the program's efficiency will deliver real-time performance and effectively keep up with the speed and dynamics of real-world environments.

Sources

[1] Wikipedia contributors. (2021, November 24). Lane departure warning system. In Wikipedia. https://en.wikipedia.org/wiki/Lane_departure_warning_system

[2] Wang, Y., & Chen, C.-S. (2020). Lane Detection: A Survey with New Results. Journal of Computer Science and Technology, 35(4), 857–878. https://doi.org/10.1007/s11390-020-0476-4

[3] Kumar, S., & Singh, S. K. (2020). Vision-Based Lane Detection for Advanced Driver Assistance Systems: A Review. In S. K. Singh & A. K. Singh (Eds.), Advances in Intelligent Systems and Computing (Vol. 1149, pp. 419–429). Springer Singapore. https://doi.org/10.1007/978-981-15-7241-8_40

[4] Son J, Yoo H, Kim S, Sohn K (2015) Real-time illumination invariant lane detection for lane departure warning system.

[5] Yoo H, Yang U, Sohn K (2013) Gradient-enhancing conversion for illumination-robust lane detection. IEEE Trans Intell Transp Syst.

[4] Haque, Md & Islam, Md & Alam, Kazi & Iqbal, Hasib & Shaik, Md. (2019). A Computer Vision based Lane Detection Approach. International Journal of Image, Graphics and Signal Processing. 11. 27-34. 10.5815/ijigsp.2019.03.04.

[5] Wei Wang, H. L. (2020). CNN based lane detection with instance. Journal of Cloud Computing: Advances, Systems.

[6] Almotairi, K. H. (2022). Hybrid adaptive method for lane detection of degraded road surface. Journal of King Saul University - Computer and Information Sciences.

[7] Jingwei Cao, C. S. (2019). Lane Detection Algorithm for Intelligent Vehicles in Complex Road Conditions and Dynamic Environments. Semantischolar.