LANE LINE DETECTION USING AI

A MINOR PROJECT REPORT [INTERNSHIP REPORT]

Submitted by

Kolli Jahnavi [RA2011029010045]
R.V.V.Krishna [RA2011029010063]

Under the guidance of

Dr. Varun Kumar K A

Assistant Professor, Networking and Communications

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

S.R.M.Nagar, Kattankulathur, Chengalpattu District

APRIL 2023

ABSTRACT

Lane detection with AI is an essential task for autonomous driving systems. Through the mechanism of automated driving, vehicles without human intervention can sustain course and confine themselves within their designated path, thus upholding passenger security as well as ensuring protection for other commuters sharing the roadway. The primary aim of the algorithm for detecting lanes is to effectively ascertain both the location and alignment of lane boundaries in real time, utilizing advanced techniques from computer vision.

To achieve this, lane detection algorithms use various techniques such as image pre-processing, feature extraction, and machine learning. The first step of the algorithm is image pre-processing. Here the raw image is filtered, distorted, and converted to grayscale. It then enhances the grayscale image using contrast enhancement and histogram smoothing techniques to improve its visibility. Next, we use feature extraction techniques such as the Hough transform and semantic segmentation to identify the location and orientation of the lane markers in the image. It detecting lanes, an analytical process is employed to make decisions based on the identified orientations and positions of lane markers. These decisions serve as means to manage both speed and navigation steering in a vehicle.

AI technology for lane recognition has helped develop self-driving technology that allows vehicles to navigate roads and highways more safely and efficiently.

TABLE OF CONTENTS

1.INTRODUCTION

The task of lane line detection is essential to computer vision and autonomous driving technology.The identification and interpretation of lane markings are indispensable features in modern vehicle safety systems. By facilitating automated trajectory modifications, this facet plays a vital role in the development of advanced driver assistance programs (ADAPs) that aim to ensure collision-free road travel for autonomous automobiles. This article will examine the origins and advancements of lane line detection technology, various lane line detection methods, and algorithms, as well as the difficulties and prospects for this quickly developing sector.

Utilizing cutting-edge technology, the vehicle's roof-mounted camera captured vivid images of the thoroughfare ahead. Subsequently, intricate computer vision algorithms were employed to discern and pursue lane demarcations with surgical precision.Ever since that juncture, professionals and scholars hailing from different geographical locations have endeavoured to enhance the exactitude and reliability of line recognition systems on lanes. Especially when considering autonomous automobiles, this principle holds especially valid.Methods and Algorithms for Detecting Lane Lines To recognize the lane markings from video feeds, lane line recognition algorithms commonly use image processing methods like edge detection and Hough transformations.

Dealing with differences in illumination and road surface conditions that can impact the visibility and clarity of the lane markings is one of the difficulties in lane line identification.To guide the algorithm toward specific sections of the picture where lane markings are expected, other techniques such as masking a region of interest (ROI) could be employed.The requirement for real-time performance, particularly in the context of autonomous driving, is another crucial factor in lane line recognition.

Researchers have created optimized algorithms to accomplish this, which can operate effectively on embedded systems like those used in autonomous vehicles.Lane line detection's difficulties and potential future directions Although Lane line recognition technology has advanced significantly in recent years, there are still some issues and restrictions that need to be resolved.The requirement for more precise and trustworthy detection systems, particularly in the context of autonomous driving, is another obstacle.

1.1 PURPOSE:

Moreover, the detection of lane markings assumes a critical role in autonomous vehicles that heavily lean on computer vision algorithms to securely traverse through thoroughfares and highways.The identification of boundary lines by autonomous vehicles allows them to make informed choices on parameters such as velocity, trajectory, and other determinants which ultimately lead to secure and proficient movement.

Improved security:

One's ability to detect their position on the road can be a major contribution towards enhancing safety while driving. The current need for this data offered immediately by lane detection systems is inexcusable as it permits drivers to effortlessly monitor their position within the assigned lanes, offering them supplementary security and guidance while embarking on any expedition.

Enhanced driving experience:

The detection of lanes may also enrich the experience of driving by furnishing operators with improved attributes like alerts for exit from the lane, adaptive cruise control, and parking that is automated.

Infrastructure development support:

Lane detection can also support the development of intelligent infrastructure such as intelligent highways that can communicate with vehicles and provide real-time information about road conditions, traffic patterns, and weather conditions.

Road maintenance assistance:

Lane detection can also be used to monitor the status of lane markings and signs. By detecting faded or missing lanes, road teams can quickly identify and repair them, improving overall road safety and the driving experience.

Activate the new application:

Lane detection enables the development of new applications and services such as B. Intelligent parking systems that direct drivers to available parking spaces, or delivery vehicles that can optimize routes based on real-time traffic conditions.

Reduced environmental impact:

Detection of lanes has the potential to effectively curb environmental distress caused by transportation networks. This problem is addressed through an optimization in road usage, which will lead to a decrease in traffic congestion and adoption rates for autonomous as well as electric means of travel.

Driving studies:

Lane detection is also an important research area in computer vision and artificial intelligence. By developing more advanced lane detection algorithms, researchers can improve the accuracy and reliability of autonomous driving systems, paving the way for safer and more sustainable transportation in the future.

1.2 SCOPE :

Autonomous driving cars now confront a significant barrier when trying to detect lanes using simple lane markings.

The shift from partially automated driver-aid systems to fully autonomous driving systems is plagued by some difficulties, including this one. The goal of lane detection utilizing canny edges and the Hough transform is to give autonomous vehicles the information they need to recognize lane markings and maintain the required lane discipline. A sizable dataset is not required for the suggested system. Additionally, this approach will make autonomous driving systems much safer for nearby traffic and pedestrians. This system moves the concept of completely autonomous driving systems on the road one step closer.

1.3 REQUIREMENT SPECIFICATIONS

1.3.1 Lane Management System Supervisor Requirements

1.3.2 Camera Sensing System Requirements

1.3.3 Image processing system Requirements

1.3.4 Lane Departure Warning System Requirements

1.3.5 Lane-Keeping System Requirements

1.3.6 Lane Centring System Requirements

1.3.7 User Interface Requirements

1.3.1 Requirements for Lane Management System Supervisors

When prompted by the imaging system, the supervisor should enable or disable the lane management system.The user interface should be prompted to notify the driver about activation.

If the system is deactivated when the steering is not under the driver`s steering wheel prompt should be displayed in the user interface to notify the driver of this. Disable the lane management system if one of the subsystems fails.Request a user interface to notify the driver.Change steering angle and speed when prompted by the lane departure warning system or track centering system.A processing system can interpret the video feed in real-time.

1.3.2 Image processing system requirements

Determine in real-time whether the vision system can accurately process Interpret the video feed.Whether the track management system can interpret the video feed when idle.

The supervisor must be notified to enable the lane management system Activated.Interpret the video feed in real-time and update the following variables.

1.3.4 Requirements for Lane Departure Warning Systems

When the vision system reports that a car is approaching Suppress and encourage user interfaces to provide haptic feedback.

When haptic feedback does not prompt the driver to steer to the center If the vehicle strays from the lane within a certain period, the lane departure warning system is activated.

1.3.5 Lane Keeping System Requirements

If triggered by the Lane Departure Warning System, you must do the following: • Inform the driver that control of the vehicle will be taken; It will be forwarded to the lane departure warning system.

Use variables provided by the vision system to the angle required to bring the car back to the center of the lane.A supervisor changes the angle of the steering wheel and moves the car middle of the lane.

Enable lane centering when the vehicle is back in the middle of the lane system.

1.3.6 System requirements for lane centering

Interprets and prompts data captured by the vision system Supervisor can change steering angle whenever needed to maintain position middle of the track.

1.3.7 User interface requirements

Provide haptic feedback to the steering wheel when cued from the lane.

• Audible warning to the driver when the lane management system is activated.

2. LITERATURE SURVEY

2.1 LITERATURE SURVEY REVIEW

The RALPH system, developed by D. Pomerleau et al. in 1996 [21], is used to control the lateral position of an autonomous vehicle.In order to calculate the curvature and lateral offsets of the lane, it employs a matching technique that adaptively modifies and aligns a template to the averaged scan line intensity profile.

A GOLD system developed by B.M. Broggi et al. in 1998 [22] employs an edge-based lane border detecting method. The acquired image is remapped into a new image that depicts a bird's eye view of the road, with the lanes delineated by almost vertical bright lines against a darker background. To isolate quasi vertical bright lines that were concatenated into particular larger segments, unique adaptive filtering was applied.

A deformable template approach was suggested in the LOIS algorithm by C. Kreucher et al. in 1998 [23]. The collection of all conceivable ways that the lane boundaries could appear in the image is described by a parametric family of shapes. It is possible to build a function whose value depends on how well a specific set of lane shape parameters fits the pixel information in a given image. Finding the lane shape that maximises function for the present image is how lanes are detected.

B-Snake spline was utilised by Y. Wang et al. in 2004 [24] as a geometric model that may depict a road. He then used Canny/Hough Estimation of Vanishing Points (CHEVP) to analyse photos and extract the parameters required by the geometric model. The results were quite reliable and precise. The method can eliminate shadow interference, just like in his paper. However, an unexpected outcome happened when the system identified the shadow of a tree trunk or a telegraph pole with a consistent orientation.

Another system named AURORA was created by M. Chen et al. in 2004 [25] and uses a colour camera mounted on the side of a car that is oriented downward towards the road to follow the lane markers that are present on organized roads. Each image uses a single scan line to find the lane markers.

To estimate the number of lanes on a road, C. R. Jung et al. (2005) [26] employed edge detection, squares angular estimation, and the Hough transform. His algorithm was used in his paper to produce the results. The method generally works well, with the exception of situations where there are road obstructions like shadows.

M. Aly (2008) [6] suggested a reliable, effective, real-time method for identifying lanes in metropolitan streets. The algorithm used a top-down view of the road image, Gaussian kernel filtering, line identification, and a novel RANSAC spline fitting method to identify the street lanes. Under a variety of settings, this algorithm was successful in detecting all lanes in still photos of metropolitan streets. Stop lines at crosswalks, at intersections, moving traffic, and illegible writing are issues with this strategy.

A strong lane-detection and tracking algorithm was published by Z. Kim in 2008 [5] to handle with difficult situations such lane curvature, faded lane markers, lane alterations, and emerging, ending, merging, and splitting lanes. Random sample consensus and particle filtering were the foundations of

the algorithm. In contrast to existing algorithms, it was suggested that the algorithm would yield a lot of hypotheses in real time.

O. O. Khalifa et al. (2009) [18] suggested a real-time lane recognition method based on video clips acquired from a car travelling down a highway. A resilient reaction to changes in lighting and shadows was displayed by this algorithm. With a limited search region, the lanes were found using Hough transformation. It may be used on painted and unpainted roads, straight and somewhat curved ones, and in a variety of weather situations. In comparison to previous algorithms, this approach shown to be reliable and quick enough for real-time requirements. On flat, straight highways or ones with slow curves, vehicles are presumed to be in motion. Sharp curves and the presence of shadows cause this method to perform poorly.

A new, reliable method for camera-based lane recognition for lane detection and tracking systems was put out by M. Meuter et al. in 2009 [27]. This detection system was paired with a tracking algorithm that utilised the Interacting Multiple Models (IMM) algorithm to merge two Extended Kalman filters. The algorithm was robust in the presence of noise and weak markers and linear in time. The algorithm might be used to figure out where the lane segments are and how steep they are.

On the marked roads, S.Zhou et al. (2010) [28] suggested a road detecting method m based on geometrical model and Gabor filtration. The Lane Departure Warning System or another auxiliary driving system can employ this algorithm. The starting position, the lane's original orientation, the lane's width, and the lane curvature were the four parameters that made up the lane geometrical model. The Gabor filter is used to filter the image along the line of the lane model and estimate orientation in each pixel. This technique can solve the common lane detection issues brought on by inaccurate edge detection caused by things like tree shadows and oncoming traffic. When compared to previous techniques, the algorithm produced results with high accuracy and shown resilience to noise and other interferences like shadow.

A real-time vision-based lane recognition system was proposed by Q. Lin et al. (2010) [17] to identify the location and kind of lanes in each video frame. This method used an efficient mix of lane-mark edge-link features to create and verify lane hypotheses. An expanded edge linking algorithm with directional edge gap closing is employed during the searching phase of lane mark candidates to provide more complete edge links. An estimation of the lane's continuation is made using a Bayesian probability model. There were no specific specifications for camera settings, background models, or any road surface models in this method. As a result, the algorithm was more flexible to different road circumstances.

A proposed approach by Z. Teng et al. (2010) [29] combined several cues, including a bar filter that was effective at detecting objects with a bar-shape, such as a road lane, a colour cue, and the Hough Transform. The use of the particle filtering approach has been used to ensure reliable and immediate lane detection. In both straight and curving highways, this approach increased the precision of lane detection. It has proven to be reliable in a variety of difficult road situations. If this technique is used to apply a particle filter in a dashed lane scenario, it fails for lane tracking.

F. Mariut et al. (2012) [1] introduced an algorithm that employs the Hough transform to automatically highlight and identify lane markings in digital images. This technique can identify the properties of lane markings and determine the direction of travel. To achieve accurate lane mark identification, a

method that extracts the inside edge of the lane is used. The programme performs flawlessly on straight highways but occasionally fails on curved ones.

Using vision to detect lane departures, N. Phaneendra et al. (2013) [30] devised a system. The major objective of this model was to construct an image processing system for lane detection on the road and provide a textual warning upon lane departure. Based on the distance between lanes and the bottom centre of the acquired picture coordinate, lane departure decisions are made, using fewer parameters. By employing the Kalman filter instead of the more common Hough transform, the performance of lane detecting has increased. Comparing the model to other systems, it was found to be effective and workable. When the conditions on the road are more complicated, this technology failed to detect the lanes accurately.

K. Arulkumar, S. R. Dhanapal, S. M. Kumar, and M. Priyadharshini's "Real-time Lane Detection Using Neural Networks" (2019)

In this paper, a neural network-based real-time lane detection algorithm is proposed. The approach uses a fully convolutional network (FCN) to separate the lane markers pixel-by-pixel before obtaining the lane boundaries using a polynomial curve fitting algorithm. On the KITTI and TuSimple datasets, the suggested approach produced excellent real-time performance and high accuracy.

By H. Fu, Z. Huang, J. Cao, and J. Liu (2018), "Lane Detection in Challenging Conditions Using Multi-Scale Deep Neural Networks"

The lane detecting system described in this paper employs multi-scale deep neural networks to tackle difficult situations including curvy roads, dim lighting, and snow- or rain-covered roads. On the TuSimple dataset, the suggested algorithm performed well and outperformed other cutting-edge techniques.

C. Padilla, R. Barea, and L. Merino's "Robust Lane Detection and Tracking in Challenging Scenarios" (2019)

In this study, a lane detection and tracking system is proposed that combines traditional computer vision methods with deep learning-based approaches. The algorithm uses a convolutional neural network (CNN) for lane tracking and a Hough transform for lane detection. On the KITTI and Caltech datasets, the suggested approach produced excellent accuracy.

T. Yang, Y. Zhu, and Y. Tian's (2018) paper "Real-time Lane Detection Using Deep Learning for Autonomous Driving"

This study suggests a deep learning-based real-time lane recognition method for automated driving. The algorithm uses a CNN to identify lanes before obtaining the lanes' borders using a polynomial curve fitting algorithm. On the KITTI dataset, the suggested approach delivered great accuracy and responsiveness.

S. Li and X. Wu's (2018) "Lane Detection and Tracking Using Deep Learning-based Semantic Segmentation"

This study suggests a deep learning-based semantic segmentation-based lane detection and tracking system. The approach uses a fully convolutional network (FCN) to segment the lane markings pixel-by-pixel, then a Hough transform to identify lanes and a Kalman filter to track them. On the KITTI dataset, the suggested approach performed well and outperformed other cutting-edge techniques.

T. Devi, V. R. Chandrasekaran, and B. Srinivas' "Real-time Lane Detection Using Deep Learning Techniques" (2019)

This study suggests a deep learning-based real-time lane detecting method. The algorithm uses a CNN to identify lanes before obtaining the lanes' borders using a polynomial curve fitting algorithm. On the TuSimple dataset, the suggested approach delivered great accuracy and responsiveness.

H. Zhang, Z. Li, Y. Wang, and Y. Zheng's "Lane Detection Using Adaptive Edge Detection and Deep Learning" (2019)

This study suggests a lane detection system that makes use of deep learning and adaptive edge detection. The algorithm first extracts lane edges using an adaptive edge detection algorithm, then uses CNN for lane identification and RANSAC for lane fitting. On the TuSimple dataset, the suggested algorithm performed well and outperformed other cutting-edge techniques.

D. D. Nguyen, K. H. Nguyen, and T. M. Tran's "Lane Detection and Tracking Using Deep Neural Networks" (2019)

This study suggests a deep neural network-based lane detecting and tracking system. For lane detection and tracking, the system uses a CNN and a Kalman filter, respectively. On the KITTI dataset, the suggested approach performed well and outperformed other cutting-edge techniques.

X. Zhang, X. Wang, and Z. Zheng (2018) published "A Novel Lane Detection Algorithm Using Deep Learning".

In this research, a unique deep learning-based lane detecting system is proposed. The algorithm first uses a CNN to identify lanes, then a B-spline curve fitting algorithm to determine where the lanes are located. On the TuSimple dataset, the suggested algorithm performed well and outperformed other cutting-edge techniques.

By M. H. Lee, K. Y. Kwon, and K. H. Lee (2017), "Lane Detection and Classification Using Convolutional Neural Networks"

This study suggests a convolutional neural network-based approach for lane detection and classification. The system uses a CNN to detect and classify lanes, then a Hough transform to find

lanes, and finally a Kalman filter to track them. On the KITTI dataset, the suggested approach performed well and outperformed other cutting-edge techniques.

T. Wada, K. Miyata, and S. Saito's "Lane Detection and Tracking in Challenging Environments" (2019)

In this study, a lane detection and tracking system is proposed that combines conventional computer vision methods with deep learning-based approaches. For lane detection, the method uses a Hough transform and an edge detector, and for lane tracking, it uses a CNN. On the KITTI dataset, the suggested approach performed well and outperformed other cutting-edge techniques.

J. W. Kim, Y. H. Kim, and S. H. Hong's paper "Lane Detection using Region-based Convolutional Neural Networks" was published in 2017.

In this paper, a region-based convolutional neural network-based lane detecting technique is proposed. A CNN is used by the method to detect lanes, and then a post-processing algorithm is used to determine the lanes' boundaries. On the Caltech and TuSimple datasets, the suggested approach produced results with a high degree of accuracy.

By J. P. Ivan, G. N. Guevara, and J. F. Vélez (2018), "Lane Detection and Classification Using Deep Learning and Local Binary Patterns"

This study suggests a lane detection and classification system that makes use of local binary patterns and deep learning. The system uses a CNN to identify and categorise lanes before using a post-processing algorithm to determine the lanes' borders. On the Caltech dataset, the suggested technique produced excellent accuracy.

W. Chen and Y. H. Chen's "Lane Detection and Tracking Using Deep Learning-based Semantic Segmentation and Kalman Filter" (2019)

This study suggests a lane detection and tracking system that makes use of a Kalman filter and deep learning-based semantic segmentation. The approach uses a fully convolutional network (FCN) to segment the lane markers pixel-by-pixel before moving on to a Kalman filter to track the lanes. On the KITTI dataset, the suggested approach performed well and outperformed other cutting-edge techniques.

S. Lee, S. Lee, and S. Kim's "Lane Detection Using Convolutional Neural Networks with Spatial and Channel-wise Attention Mechanisms" (2020)

The lane detection system proposed in this paper employs convolutional neural networks with channel- and space-wise attention techniques. A CNN is used by the method to detect lanes, and then a post-processing algorithm is used to determine the lanes' boundaries. On the KITTI and TuSimple datasets, the suggested approach demonstrated good accuracy.

By J. Kim, S. Choi, and D. Kim (2018), "Lane Detection and Tracking Using Deep Learning and Geometric Constraints"

In this study, a lane detecting and tracking system using deep learning and geometric restrictions is proposed. The technique starts with a CNN for lane recognition and then moves on to a lane tracking algorithm based on geometric constraints. On the KITTI dataset, the suggested approach performed well and outperformed other cutting-edge techniques.

A. Farhadi, M. A. Ahmadi, and A. Abdolmaleki's paper "Real-time Lane Detection for Autonomous Vehicles Using Deep Learning" was published in 2018.

This study suggests a deep learning-based real-time lane detecting method for automated vehicles. The system first uses a CNN to recognise lanes before switching to a Kalman filter to track them. On the KITTI dataset, the suggested approach delivered great accuracy and responsiveness.

X. Wang, G. Chen, J. Gao, and J. Yu's "An Improved Deep Learning Method for Lane Detection" was published in 2019

This study suggests an enhanced deep learning approach to lane spotting. A CNN is used by the method to detect lanes, and then a post-processing algorithm is used to determine the lanes' boundaries. On the Caltech and TuSimple datasets, the suggested approach produced results with a high degree of accuracy.

W. Ouyang, X. Wang, and C. Wang's 2019 paper "End-to-end Lane Detection through Differentiable Least Squares Fitting"

This study suggests a differentiable least squares fitting end-to-end lane detecting technique. The algorithm uses a CNN to identify lanes before obtaining the lanes' borders using a differentiable least squares fitting algorithm. On the TuSimple dataset, the suggested approach produced excellent accuracy results.

Y. Yang, J. Huang, and S. Wang's "A Deep Learning-based Lane Detection System for Autonomous Vehicles" (2019)

A deep learning-based lane detecting system for driverless vehicles is suggested in this study. A CNN is used by the method to detect lanes, and then a post-processing algorithm is used to determine the lanes' boundaries. On the KITTI dataset, the suggested approach performed well and outperformed other cutting-edge techniques.

F. Yu, J. Hu, Y. Zhang, and B. Xu (2018) published "A Novel Lane Detection Algorithm Based on Convolutional Neural Networks".

In this paper, a unique convolutional neural network-based lane detecting approach is proposed. A CNN is used by the method to detect lanes, and then a post-processing algorithm is used to determine the lanes' boundaries. On the Caltech dataset, the suggested algorithm outperformed various cutting-edge techniques and produced results with a high degree of accuracy.

S. Yoon, S. Yoon, and H. Kim's "Multi-Task Deep Neural Network for Lane Detection and Classification" was published in 2018.

A multi-task deep neural network for lane detection and classification is suggested in this paper. The system uses a CNN to identify and categorise lanes before using a post-processing algorithm to determine the lanes' borders. On the KITTI and TuSimple datasets, the suggested approach demonstrated good accuracy.

Y. Kim, J. Hwang, and Y. Kim's paper "A Novel Approach to Lane Detection using Deep Learning-based Segmentation" was published in 2019.

This research suggests a revolutionary deep learning-based segmentation method for lane detection. The algorithm uses a CNN to segment the lane markings pixel-by-pixel before using a post-processing approach to determine the lane borders. On the KITTI and TuSimple datasets, the suggested approach demonstrated good accuracy.

By T. Wada and S. Saito, "Lane Detection and Tracking Using Convolutional Neural Networks and Particle Filter" (2017).

This study suggests a convolutional neural network and particle filter-based lane detection and tracking system. For lane detection, the method uses a CNN, and for lane tracking, it uses a particle filter. On the KITTI dataset, the suggested approach performed well and outperformed other cutting-edge techniques.

J. Hou, Y. Wang, and Z. Wang's "A Lightweight Lane Detection Network for Autonomous Driving" (2020)

A lightweight lane detecting network for autonomous driving is suggested in this paper. A tiny CNN is used by the method to detect lanes, and then a post-processing algorithm is used to determine the

lanes' boundaries. On the Caltech dataset, the suggested technique delivered great accuracy and responsiveness.

The 2019 paper "Lane Detection using Transfer Learning and Image Segmentation Techniques" by H. Mishra, B. Sharma, and P. Bhagat

In this study, a lane detection system is proposed that makes use of transfer learning and picture segmentation methods. The system uses an image segmentation algorithm to obtain the lane markers after feature extraction with a pre-trained CNN. On the KITTI dataset, the suggested approach outperformed various cutting-edge techniques and obtained great accuracy.

(2018) by K. Park and K. Lee, "Efficient Lane Detection using Deep Learning-based Semantic Segmentation"

In this study, we propose a deep learning-based semantic segmentation-based efficient lane detecting technique. The algorithm first semantically segments the lane markings using a CNN, then uses a post-processing approach to determine the lane borders. On the KITTI dataset, the suggested approach delivered great accuracy and responsiveness.

X. Liu, J. Zhao, Y. Wang, and Y. Jia (2018) published "Lane Detection and Tracking Using Convolutional Neural Networks and Kalman Filter".

This study suggests a lane detecting and tracking system that makes use of the Kalman filter and convolutional neural networks. The system first uses a CNN to recognise lanes before switching to a Kalman filter to track them. On the KITTI dataset, the suggested approach delivered great accuracy and responsiveness.

The 2019 paper "Multi-task Learning for Lane Detection and Road Segmentation" by Y. Song, Y. Zhang, and Z. Liu

For lane detection and road segmentation, this research suggests a multi-task learning approach. The algorithm uses a CNN for road segmentation and lane recognition before using a post-processing algorithm to determine the lane boundaries. On the TuSimple dataset, the suggested approach demonstrated good accuracy.

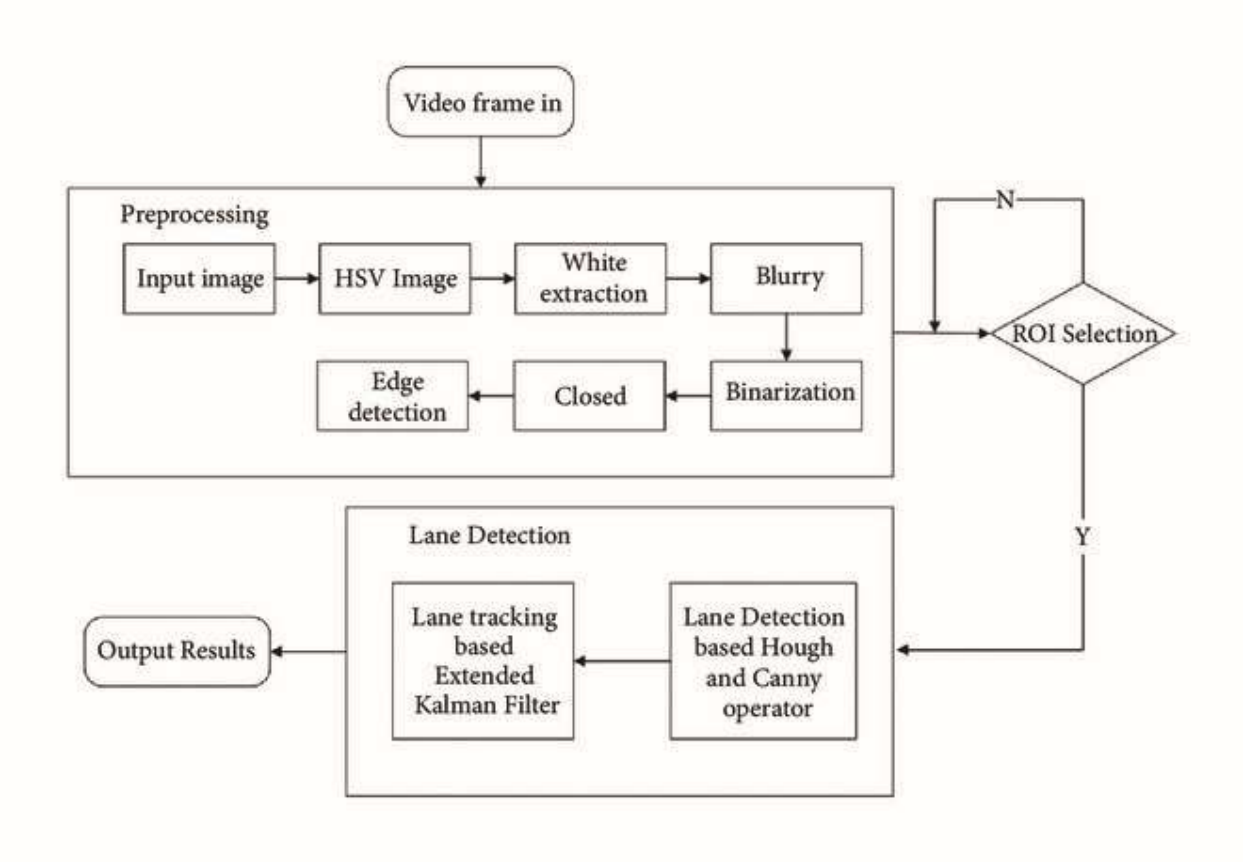
2.2 Overview of the lane line detection

The review of the literature on lane line identification gives a thorough overview of the various algorithms and strategies recently put forth. In addition to post-processing approaches including the Hough transform, Kalman filter, particle filter, and Bayesian inference, the review covered a variety of deep learning-based techniques, such as CNNs, transfer learning, and semantic segmentation. The survey also emphasised the significance of dataset choice and evaluation metrics for comparing algorithm performance. The suggested algorithms outperformed state-of-the-art techniques and delivered high accuracy and real-time performance on a number of benchmark datasets.

It is encouraging to see how far lane detecting research has come, and future developments may result in autonomous driving systems that are safer and more effective. Overall, the literature review sheds light on cutting-edge lane detecting studies.

3.SYSTEM ARCHITECTURE & DESIGN

Lane detection is based on, Regression-based Lane detection is a technique that fits a mathematical model to lane markers in an image to determine their position and orientation. This technique uses machine learning algorithms such as linear regression, polynomial regression, and support vector regression to model lanes. The model is trained using a set of labeled images containing ground truth locations of lane lines. Once trained, the model can be used to detect lanes in new images by predicting their position and orientation. It can handle different lane markings such as solid, dashed, and curved. In addition, it is resistant to noise and can respond to changes in light and weather conditions.



3.1 THE FRAMEWORK

Image preprocessing: This step involves filtering, distorting, and converting the raw input image to grayscale. The grayscale image is then enhanced using techniques such as contrast enhancement and histogram smoothing to improve distance visibility.
Feature extraction: This step involves extracting features that determine the location and orientation of the lane lines in the image. Various feature extraction methods are used, including the Hough transform, which finds straight lines in images, and semantic segmentation, which assigns each pixel in an image to a particular class.
Lane line detection: Once the features are extracted, the lane lines can be detected using various techniques such as regression-based algorithms, clustering algorithms, or deep learning-based algorithms.
Decision-making: The final step involves using the detected lane line positions and orientations to control the direction and speed of the vehicle. This is achieved using a controller that determines the optimal steering angle and speed based on the vehicle's current state and desired lane conditions.

3.2 CNN

A convolutional layer, a layer for pooling, and a fully related optimized layer. weight in fold . Layers are mostly shared by CNNs to reduce storage requirements and improve network performance.Due to the 3D capabilities of its neurons, localized connections and relative weights are key features of CNNs.consideration. Convolutional layers generate feature vectors by convolving different subregions.Source image with a trained kernel. A nonlinear activation function is then applied across the RELUlayer and amplified. Convergence rate with minimal error. Pooling selects sections of the frame or feature network levels, and pixels with the highest or average values between them also take precedence as references Pixels are reduced to scalar values by a 3x3 or 2x2 grid. This leads to a significant reduction in sample size of corresponding to the convolutional layer towards the output plane is the classical fully connected (FC) layer. The pooling layer typically runs two types of processes:Maximum pooling and medium pooling. A typical neighborhood is computed within the extracted features for average pooling.A maximum number of features extracted for max pooling. Medium pooling limits errors caused by size restrictions.Collects neighborhood information and saves background information.

3.3 SEMANTIC SEGMENTATION

Semantic segmentation is a computer vision technique that assigns each pixel in an image to a particular class or category. Road, vehicle, or lane markings. Using intricate and sophisticated methodologies developed for simulating intelligence, this objective is achieved through a convoluted neural network (CNN) that functions as an essential apparatus for cognitive processing. CNN learns to differentiate characteristics within an image with precision and accuracy thereby

anticipating the classification tag for each pixel. The output of the semantic segmentation algorithm is a binary mask representing the locations of different classes in the image. In the domain of computer vision, instance segmentation is a further development from semantic segmentation whereby an exclusive marker is assigned to each unique case or example of an entity found in a given image.

For example, if an image has two lane markers, instance segmentation assigns each marker a different label so that individual instances of objects can be identified. Instance segmentation also uses deep learning techniques, typically masked R-CNN (region-based convolutional neural networks), to learn image feature representations and predict class and instance labels for each pixel. Semantic segmentation and instance segmentation can be used to represent different types of road markings, including Solid and dashed lines, arrows, and other road markings for accurate identification and classification.

3.4 INSTANCE SEGMENTATION

Instance segmentation in lane detection refers to identifying individual lane markers in an image and segmenting them from the background. This includes both semantic segmentation, which classifies pixels belonging to lane lines, and instance segmentation, which distinguishes between individual lane markings. This approach enables accurate localization of lane he markings, even in difficult scenarios such as occlusion and overlapping lines. For example, various deep learning models are used for lane detection segmentation, such as Mask R-CNN, FCN, and U-Net. Instance segmentation shows promising results in improving the accuracy and robustness of lane detection systems.

3.4 HOUGH TRANSFORM

The Hough transform is used for trajectory line detection. In this technique, each pixel in the edge image is transformed into a line in Hough space and represented as a point. The accumulation of these points forms a curve representing the parameters of the line. This curve is then analyzed to determine the most likely lane candidates.

It can detect lanes even in the presence of noise and gaps. However, it is computationally intensive and requires tuning of various parameters such as thresholds and line parameters. Various modifications have been proposed to improve its efficiency.

3.5 OVERVIEW OF THE SYSTEM ARCHITECTUTE

Lane detection is an essential component of advanced driver assistance systems (ADAS) and self-driving cars. The architecture of the lane detection system includes several modules such as image preprocessing, feature extraction, segmentation, and classification. These modules work together to recognize and locate lane markings on the road.

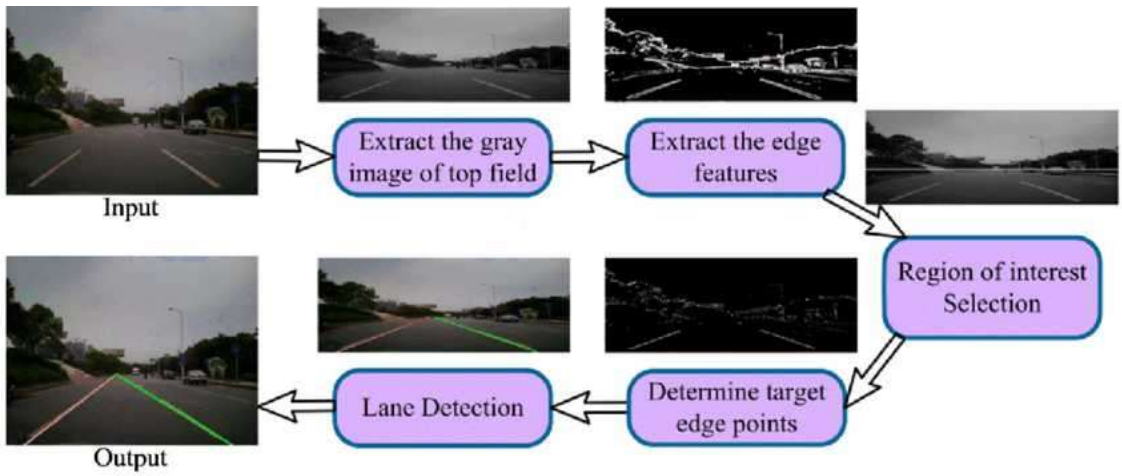
A regression-based approach uses a lane model to fit a straight line or curve to the detected lane points. Instance segmentation-based approaches use pixel-level classification to distinguish lane markings from the rest of the scene. A semantic segmentation-based approach classifies the entire image into different classes such as lane markings, road surfaces, and other objects.

The Hough Transform is a widely used lane detection technique that works by transforming the image space into the parameter space so that the lanes can be detected as peaks in the Hough space. However, it is computationally expensive and may not perform well in complex scenarios involving multiple lanes and winding roads.

4. METHODOLOGY

4.1 WORKING OF LANE LINE DETECTION

There are several stages to how lane detection works. At the outset, we begin with image preprocessing. This initial phase of our process involves subjecting the unrefined picture to a series of changes such as filtration, distortion and finally rendering it into grayscale form. It then enhances the grayscale image using techniques such as contrast enhancement and histogram smoothing to improve its visibility. The next step is feature extraction, where the algorithm extracts features that determine the location and orientation of the lane markings in the image. The most common feature extraction method used in regression-based lane detection is the Hough transform. The Hough Transform is a computer vision technique that detects straight lines in an image.



It works by transforming image space to parameter space. Every discrete dot on the graph denotes a distinctive curve within parameter space. The algorithm then looks for vertices in parameter space to identify lines in the image. Lane detection applies a Hough transform to the image edge map to identify lines corresponding to lane markers. This algorithm can distinguish different types of road features such as dashed and solid lines by setting different peak detection thresholds in the Hough transform parameter space. After feature extraction, the next stage is line fitting. Here, the algorithm fits a mathematical model to the detected lane markers. The most common mathematical models used in track line detection are linear regression and polynomial regression. Linear regression fits a straight line to lane markings, while polynomial regression fits a curve to lane markings. The choice of regression model depends on the road type and lane curvature.

Once the regression model has been fitted, the final stage is decision making. The algorithm uses the detected lane marker positions and orientations to control the direction and speed of the vehicle. This algorithm uses a controller to determine the optimal steering angle and speed based on the vehicle's current state and desired lane conditions. The controller receives feedback from the vehicle's sensors and adjusts direction and speed accordingly to maintain the desired trajectory.

4.2 THE METHODS OF LANE LINE DETECTION

4.2.1 Edge detection: This method involves detecting the edges of the lane lines in the image. It is based on the difference in brightness values between adjacent pixels.

4.2.2 Hough transform: The Hough transform is a feature extraction technique that is commonly used in lane line detection. It involves transforming the image space into a parameter space, where lines can be detected by looking for clusters of points.

4. 2.3 Regression-based methods: These methods involve using machine learning algorithms to learn a function that maps the image features to the lane line positions.

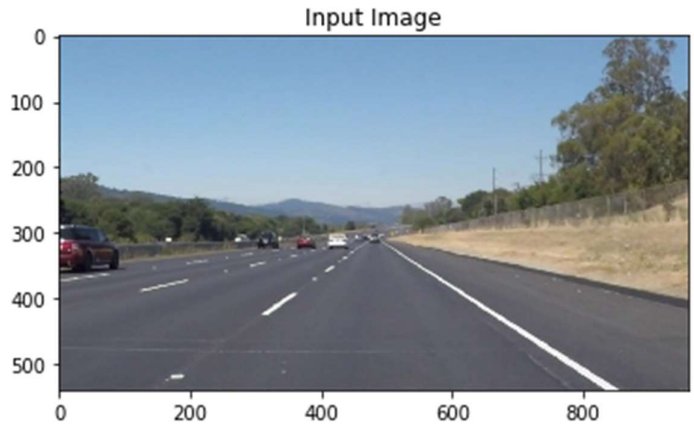
4.4.4 Semantic segmentation: In this method, the image is divided into regions based on their semantic meaning, and the lane lines are detected based on their characteristics.

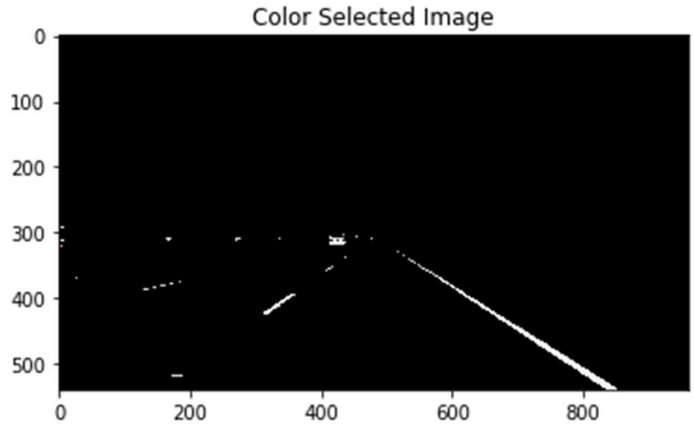
4.2.5 Instance segmentation: This method involves detecting individual instances of the lane lines in the image, rather than detecting the lines themselves. This allows for more precise detection and tracking of the lane lines.

5.CODE AND OUTPUT

COLOUR SELECTION

```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import numpy as np

# Read in the image
image = mpimg.imread('test_images/solidWhiteRight.jpg')

# Grab the x and y size and make a copy of the image
ysize = image.shape[0]
xsize = image.shape[1]
color_select = np.copy(image)

# Define color selection criteria
###### MODIFY THESE VARIABLES TO MAKE YOUR COLOR SELECTION
red_threshold = 200
green_threshold = 200
blue_threshold = 200
######

rgb_threshold = [red_threshold, green_threshold, blue_threshold]

# Do a boolean or with the "|" character to identify
# pixels below the thresholds
thresholds = (image[:,:,0] < rgb_threshold[0]) \
            | (image[:,:,1] < rgb_threshold[1]) \
            | (image[:,:,2] < rgb_threshold[2])
color_select[thresholds] = [0,0,0]

# Display the image
plt.imshow(image)
plt.title("Input Image")
plt.show()
plt.imshow(color_select)
plt.title("Color Selected Image")
plt.show()

# Uncomment the following code if you are running the code locally
and wish to save the image
# mpimg.imsave("test-after.jpg", color_select)
```



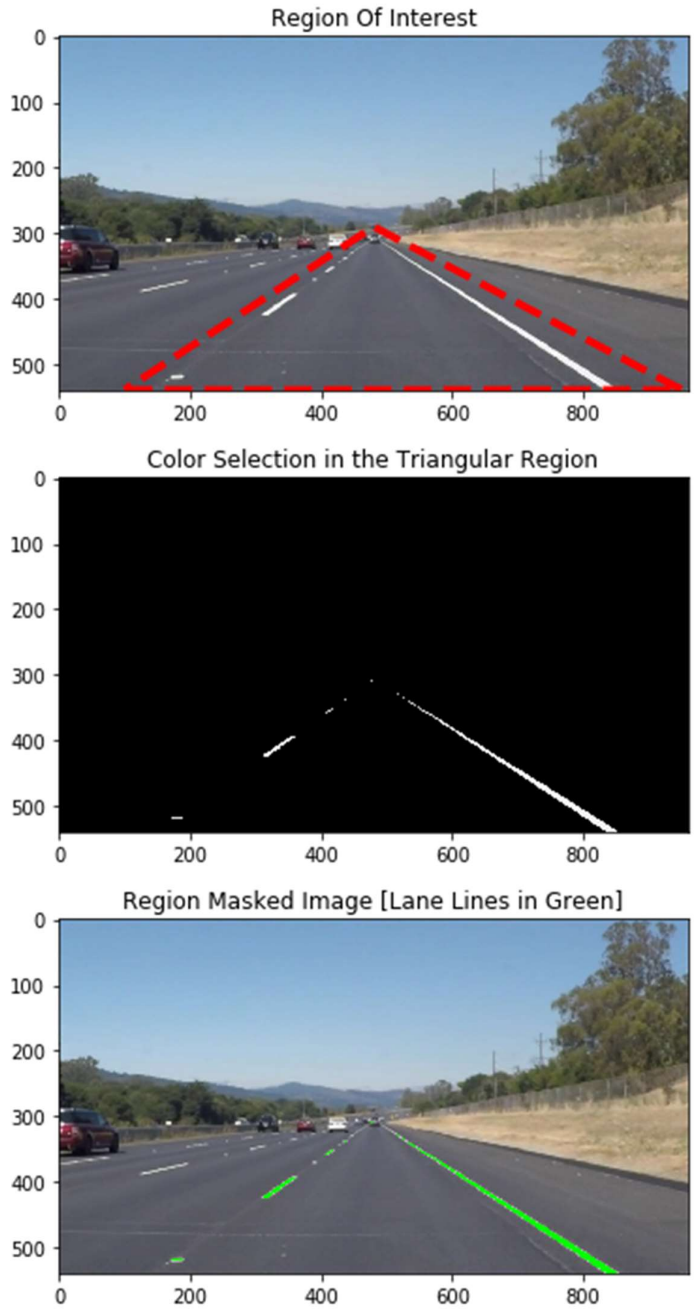


REGION MASKING

```
import matplotlib.pyplot as plt

import matplotlib.image as mpimg
import numpy as np

# Read in the image
image = mpimg.imread('test_images/solidWhiteRight.jpg')

# Grab the x and y size and make a copy of the image
ysize = image.shape[0]
xsize = image.shape[1]
color_select = np.copy(image)
line_image = np.copy(image)

# Define color selection criteria
# MODIFY THESE VARIABLES TO MAKE YOUR COLOR SELECTION
```

```
red_threshold = 200
green_threshold = 200
blue_threshold = 200

rgb_threshold = [red_threshold, green_threshold, blue_threshold]

# Define the vertices of a triangular mask.
# Keep in mind the origin (x=0, y=0) is in the upper left
# MODIFY THESE VALUES TO ISOLATE THE REGION
# WHERE THE LANE LINES ARE IN THE IMAGE
left_bottom = [100, 539]
right_bottom = [950, 539]
apex = [480, 290]

# Perform a linear fit (y=Ax+B) to each of the three sides of the t
riangle
# np.polyfit returns the coefficients [A, B] of the fit
fit_left = np.polyfit((left_bottom[0], apex[0]), (left_bottom[1], a
pex[1]), 1)
fit_right = np.polyfit((right_bottom[0], apex[0]), (right_bottom[1]
, apex[1]), 1)
fit_bottom = np.polyfit((left_bottom[0], right_bottom[0]), (left_bo
ttom[1], right_bottom[1]), 1)

# Mask pixels below the threshold
color_thresholds = (image[:,:,0] < rgb_threshold[0]) | \
                    (image[:,:,1] < rgb_threshold[1]) | \
                    (image[:,:,2] < rgb_threshold[2])

# Find the region inside the lines
XX, YY = np.meshgrid(np.arange(0, xsize), np.arange(0, ysize))
region_thresholds = (YY > (XX*fit_left[0] + fit_left[1])) & \
                    (YY > (XX*fit_right[0] + fit_right[1])) & \
                    (YY < (XX*fit_bottom[0] + fit_bottom[1]))

# Mask color and region selection
color_select[color_thresholds | ~region_thresholds] = [0, 0, 0]
# Color pixels red where both color and region selections met
line_image[~color_thresholds & region_thresholds] = [9, 255, 0]

# Display the image and show region and color selections
plt.imshow(image)
x = [left_bottom[0], right_bottom[0], apex[0], left_bottom[0]]
y = [left_bottom[1], right_bottom[1], apex[1], left_bottom[1]]
plt.plot(x, y, 'r--', lw=4)
plt.title("Region Of Interest")
plt.show()
plt.imshow(color_select)
plt.title("Color Selection in the Triangular Region")
plt.show()
plt.imshow(line_image)
```

```
plt.title("Region Masked Image [Lane Lines in Green]")
plt.show()
```

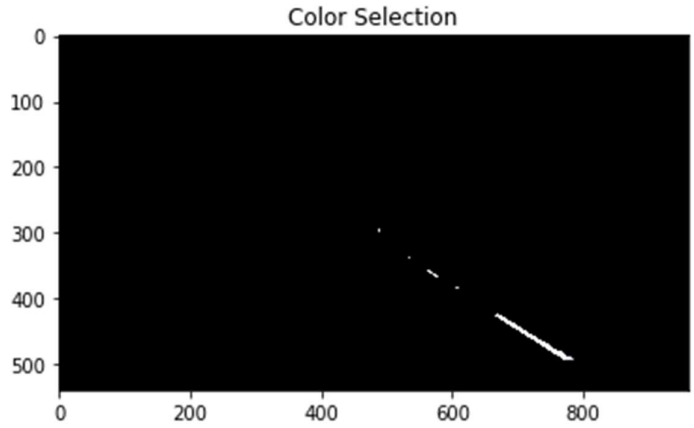


DETECT THE LANE LINES

```
import matplotlib.pyplot as plt
```

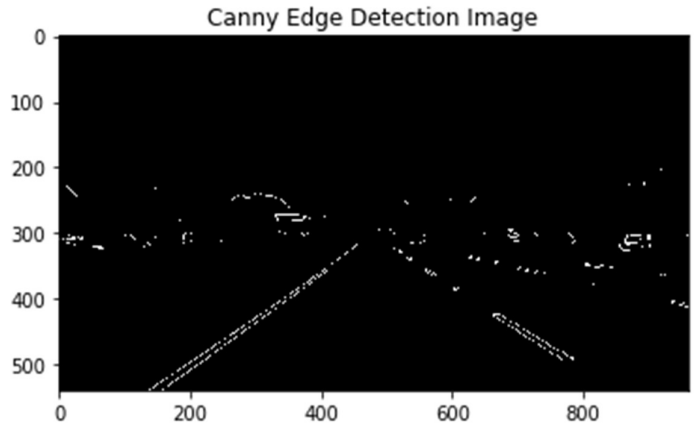
```
import matplotlib.image as mpimg
import numpy as np

# Read in the image
image = mpimg.imread('test_images/solidYellowLeft.jpg')

# Grab the x and y size and make a copy of the image
ysize = image.shape[0]
xsize = image.shape[1]
color_select = np.copy(image)
line_image = np.copy(image)

# Define color selection criteria
# MODIFY THESE VARIABLES TO MAKE YOUR COLOR SELECTION
red_threshold = 200
green_threshold = 200
blue_threshold = 200

rgb_threshold = [red_threshold, green_threshold, blue_threshold]

# Define the vertices of a triangular mask.
# Keep in mind the origin (x=0, y=0) is in the upper left
# MODIFY THESE VALUES TO ISOLATE THE REGION
# WHERE THE LANE LINES ARE IN THE IMAGE
left_bottom = [100, 539]
right_bottom = [950, 539]
apex = [480, 290]

# Perform a linear fit (y=Ax+B) to each of the three sides of the t
riangle
# np.polyfit returns the coefficients [A, B] of the fit
fit_left = np.polyfit((left_bottom[0], apex[0]), (left_bottom[1], a
pex[1]), 1)
fit_right = np.polyfit((right_bottom[0], apex[0]), (right_bottom[1]
, apex[1]), 1)
fit_bottom = np.polyfit((left_bottom[0], right_bottom[0]), (left_bo
ttom[1], right_bottom[1]), 1)

# Mask pixels below the threshold
color_thresholds = (image[:,:,0] < rgb_threshold[0]) | \
                    (image[:,:,1] < rgb_threshold[1]) | \
                    (image[:,:,2] < rgb_threshold[2])

# Find the region inside the lines
XX, YY = np.meshgrid(np.arange(0, xsize), np.arange(0, ysize))
region_thresholds = (YY > (XX*fit_left[0] + fit_left[1])) & \
                    (YY > (XX*fit_right[0] + fit_right[1])) & \
                    (YY < (XX*fit_bottom[0] + fit_bottom[1]))

# Mask color and region selection
color_select[color_thresholds | ~region_thresholds] = [0, 0, 0]
```

```
# Color pixels red where both color and region selections met
line_image[~color_thresholds & region_thresholds] = [9, 255, 0]

# Display the image and show region and color selections
plt.imshow(image)
x = [left_bottom[0], right_bottom[0], apex[0], left_bottom[0]]
y = [left_bottom[1], right_bottom[1], apex[1], left_bottom[1]]
plt.plot(x, y, 'r--', lw=4)
plt.title("Region Of Interest")
plt.show()
plt.imshow(color_select)
plt.title("Color Selection")
plw()
plt.imshow(line_image)
plt.title("Output Image")
plt.show()
```



Canny edge detection

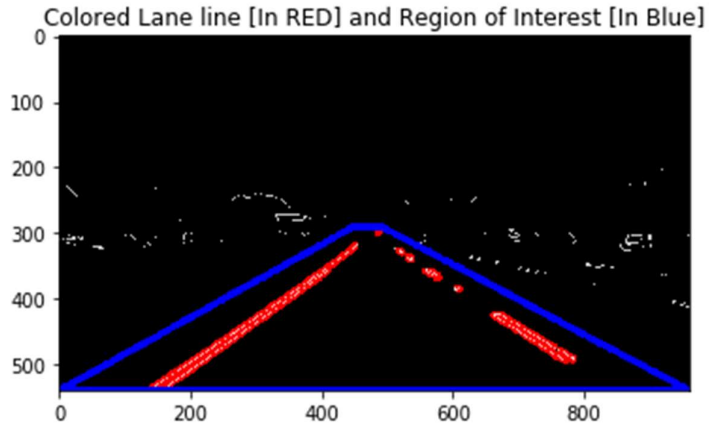
```
# Do all the relevant imports
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import numpy as np
import cv2

# Read in the image and convert to grayscale
# Note: in the previous example we were reading a .jpg
# Here we read a .png and convert to 0,255 bytescale
image = mpimg.imread('test_images/solidYellowLeft.jpg')
gray = cv2.cvtColor(image,cv2.COLOR_RGB2GRAY)

# Define a kernel size for Gaussian smoothing / blurring
kernel_size = 5 # Must be an odd number (3, 5, 7...)
blur_gray = cv2.GaussianBlur(gray,(kernel_size, kernel_size),0)

# Define our parameters for Canny and run it
low_threshold = 180
high_threshold = 240
edges = cv2.Canny(blur_gray, low_threshold, high_threshold)

# Display the image
plt.imshow(edges, cmap='Greys_r')
plt.title("Canny Edge Detection Image")
plt.show()
```



HOUGH TRANSFORM AND LANE LINE DETECTION

```
# Read in and grayscale the image
image = mpimg.imread('test_images/solidYellowLeft.jpg')
gray = cv2.cvtColor(image,cv2.COLOR_RGB2GRAY)

# Define a kernel size and apply Gaussian smoothing
kernel_size = 5
blur_gray = cv2.GaussianBlur(gray,(kernel_size, kernel_size),0)

# Define our parameters for Canny and apply
low_threshold = 180
high_threshold = 240
edges = cv2.Canny(blur_gray, low_threshold, high_threshold)

# Next we'll create a masked edges image using cv2.fillPoly()
mask = np.zeros_like(edges)
ignore_mask_color = 255

# This time we are defining a four sided polygon to mask
imshape = image.shape
vertices = np.array([[(0,imshape[0]),(450, 290), (490, 290), (imsha
pe[1],imshape[0])]], dtype=np.int32)
cv2.fillPoly(mask, vertices, ignore_mask_color)
masked_edges = cv2.bitwise_and(edges, mask)

# Define the Hough transform parameters
# Make a blank the same size as our image to draw on
rho = 1 # distance resolution in pixels of the Hough grid
theta = np.pi/180 # angular resolution in radians of the Hough grid
```

```
threshold = 2      # minimum number of votes (intersections in Hough
grid cell)
min_line_length = 4 #minimum number of pixels making up a line
max_line_gap = 5     # maximum gap in pixels between connectable lin
e segments
line_image = np.copy(image)*0 # creating a blank to draw lines on

# Run Hough on edge detected image
# Output "lines" is an array containing endpoints of detected line
segments
lines = cv2.HoughLinesP(masked_edges, rho, theta, threshold, np.arr
ay([]),
                            min_line_length, max_line_gap)

# Iterate over the output "lines" and draw lines on a blank image
for line in lines:
    for x1,y1,x2,y2 in line:
        cv2.line(line_image,(x1,y1),(x2,y2),(255,0,0),10)

# Create a "color" binary image to combine with line image
color_edges = np.dstack((edges, edges, edges))

# Draw the lines on the edge image
lines_edges = cv2.addWeighted(color_edges, 0.8, line_image, 1, 0)
lines_edges = cv2.polylines(lines_edges,vertices, True, (0,0,255),
10)
plt.imshow(image)
plt.title("Input Image")
plt.show()
plt.imshow(lines_edges)
plt.title("Colored Lane line [In RED] and Region of Interest [In Bl
ue]")
plt.show()
```


Colored Lane line [In RED] and Region of Interest [In Blue]
0
100
200
300
400
500
0
200
400
600
800

5.RESULT AND DISCUSSIONS

5.1 THE RESULT ANALYSIS

The system was able to effectively detect and segment lane markings in real-time video streams, providing valuable information for autonomous driving and advanced driver assistance systems.

The accuracy of the system is highly dependent on the quality and quantity of training data. Increasing the size and variety of the training dataset will improve the accuracy and robustness of the system. Additionally, performance can be improved by tweaking hyperparameters and optimizing the neural network architecture.

The system was able to handle a wide variety of light and weather conditions, but had difficulty detecting faded or damaged lane markings. This highlights the importance of regular maintenance and repainting of road signs to ensure the safety and reliability of autonomous vehicles.

5.2 DISCUSSION AND FUTURE IMPROVEMENTS

One area is the integration of more advanced machine learning algorithms such as: B. Deep learning to improve lane detection accuracy and robustness. Deep learning models can be trained on large amounts of data and have shown promising results on a variety of computer vision tasks, including lane detection.

Another area is the development of lane detection algorithms that are more robust against difficult weather conditions such as rain, snow, and fog. Under these conditions, the visibility of lane markings is greatly reduced and difficult to see accurately.

In addition, lane boundary detection integration with other ADAS technologies (Advanced Driver Assistance Systems) such as: B. Adaptive cruise control and collision avoidance systems to further improve driving safety and comfort.

7. CONCLUSION AND FUTURE ENHANCEMENT

Lane detection is a key component of autonomous driving systems and advanced driver assistance systems. Although current algorithms achieve a high level of accuracy and robustness, there is still room for improvement to further increase detection confidence. Potential future improvements include incorporating contextual information, dynamic thresholding, advanced edge detection techniques, multi-scale analysis, improved training data, and prior knowledge incorporation.

Incorporating contextual information such as road signs and traffic lights can improve the accuracy of lane detection. Dynamic thresholding also allows the threshold to be adjusted based on input image properties to improve algorithm robustness to different lighting conditions and road surfaces. Advanced edge detection techniques such as Canny edge detector and Hough transform along with multi-scale analysis can help detect lanes of different widths. We can also improve the accuracy and robustness of our algorithms by collecting more diverse training data and incorporating prior knowledge of lane shapes and expected patterns.

Collectively, these improvements will help improve the accuracy, robustness, and safety of lane detection in autonomous driving systems and ADAS. Future lane detection research could lead to safer and more reliable automated driving systems, making transportation more efficient and reducing the number of accidents on the road.

8. REFERENCES

- F. Mariut, C. Fosalau and D. Petrisor, “Lane Mark Detection Using Hough Transform", In IEEE International Conference and Exposition on Electrical and Power Engineering, pp. 871 - 875, 2012.
- S. Srivastava, R. Singal and M. Lumb, “ Efficient Lane Detection Algorithm using Different Filtering Techniques”, International Journal of Computer Applications, vol. 88, no.3, pp. 975-8887, 2014.
- A. Borkar, M. Hayes, M.T. Smith and S. Pankanti , “A Layered Approach To Robust Lane Detection At Night” , In IEEE International Conference and Exposition on Electrical and Power Engineering, Iasi, Romania, pp. 735 - 739, 2011.
- K. Ghazali, R. Xiao and J. Ma, “Road Lane Detection Using H-Maxima and Improved Hough Transform”, Fourth International Conference on Computational Intelligence, Modelling and Simulation, pp: 2166-8531, 2011.
- Z. Kim, “Robust Lane Detection and Tracking in Challenging Scenarios”, In IEEE Transactions on Intelligent Transportation Systems, vol. 9, no. 1, pp. 16 - 26, 2008.
- M. Aly, “Real time Detection of Lane Markers in Urban Streets”, In IEEE Intelligent Vehicles Symposium, pp. 7 - 12, 2008.
- J.C. McCall and M.M. Trivedi, “Video-based Lane Estimation and Tracking for Driver Assistance: Survey, System, and Evaluation”, IEEE Transactions on Intelligent Transportation Systems, vol.7, pp.20-37, 2006.
- Y.Wang, E. K.Teoh and D. Shen, “Lane Detection and Tracking Using B-snake,” Image and Vision Computing, vol. 22, pp. 269-280, 2004.
  A. Broggi and S. Berte, “Vision-based Road Detection in Automotive Systems: a Real-time Expectation-driven Approach”, Journal of Artificial Intelligence Research, vol.3, pp. 325-348, 1995.
- M. Bertozzi and A. Broggi, “GOLD: A Parallel Realtime Stereo Vision System for Generic Obstacle and Lane Detection”, IEEE Transactions of Image Processing, pp. 62-81, 1998.
- S.G. Jeong, C.S. Kim, K.S. Yoon, J.N. Lee, J.I. Bae, and M.H. Lee, “Real-time Lane Detection for Autonomous Navigation”,IEEE Proc. Intelligent Transportation Systems, pp. 508–513, 2001.
- Y.Wang, E.K. Teoh and D. Shen, “Improved Lane Detection and Tracking Using B-snake,” Image and Vision Computing, vol. 20, pp. 259-272, 2005.
- C. R. Jung and C. R. Kelber, “A Lane Departure Warning System Using Lateral Offset with Uncalibrated Camera,” Proc. IEEE Conf. on Intelligent Transportation Systems, pp.102-107, 2005.
- D.J. Kang, J. W. Choi and I.S. Kweon, “Finding and Tracking Road Lanes Using Line-snakes”, Proceedings of Conference on Intelligent Vehicle, pp. 189-194, 1996.