ROAD LANE LINE DETECTION

A MAJOR PROJECT-I (Synopsis)
Submitted in Partial Fulfillment of the Requirement for the Award of the Degree of
BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE & ENGINEERING-DATA SCIENCE
SUBMITTED TO

Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal (M.P.)

SUBMITTED BY
AVINASH GIRI (0176CD211032)
SAMDHISH YADAV (0176CD211104)
VIRAT RAGHUWANSHI (0176CD211133)
PUSHPENDRA CHOUDHARY (0176CD211090)

UNDER THE SUPERVISION OF
Prof. Satish Choudhary
Department of Computer Science & Engineering-Data Science

Department of Computer Science & Engineering-Data Science

Lakshmi Narain College of Technology Excellence, Bhopal (M.P.)

MAY 2025

Lakshmi Narain College of Technology Excellence, Bhopal

Department of Computer Science & Engineering-Data Science

ACKNOWLEDGEMENT

At the outset, I would like to link to thank the Almighty who made all the things possible. Writing this project report would not have been possible without the support of several people whom we need to whole heartedly thank. I express a deep sense of gratitude to my supervisor **"Satish Choudhary", Dept. of CSE-Data Science** for the valuable and inspirational guidance from the initial to the final level that enabled me to develop an understanding of this Project work.

I would like to give my sincere thanks to Prof. (**Dr.) Neetsh Kumar Gupta, Head, Dept. of CSE-Data Scienc**e for their kind help, encouragement and co-operation throughout my Project period and I owe my special thanks to Principal **Dr. Anil Kumar Saxena** for their guidance and suggestions during the Project work.

Lastly, I want to thank my parents, friends and all those people who contributed to my/our project directly or indirectly for their moral and psychological support.

Lakshmi Narain College of Technology Excellence, Bhopal

Department of Computer Science & Engineering-Data Science

CANDIDATE'S DECLARATION

We, **AVINASH GIRI, SAMDHISH YADAV, PUSHPENDRA CHOUDHARY and VIRAT RAGHUWANSHI,** Students of **Bachelor of Technology, Computer Science & Engineering-Data Science, Lakshmi Narain College of Technology Excellence, Bhopal** session 2024-25 hereby declare that the work presented in the Major Project entitled **"ROAD LANE LINE DETECTION"** is outcome of my/our own Bonafide work, which is correct to the best of my/our knowledge and this work has been carried out taking care of Engineering Ethics. The work presented does not infringe on any previous work and has not been submitted to any University for the award of any degree / diploma.

We also declare that **"A check for plagiarism has been carried out on the Minor Project-II and is found within the acceptable limit and report of which is enclosed herewith".**

AVINASH GIRI
0176CD211032
SAMDHISH YADAV
0176CD1211104
PUSHPENDRA CHOUDHARY
0176CD211090
VIRAT RAGHUWANSHI
0176CD211133

Lakshmi Narain College of Technology Excellence, Bhopal

Department of Computer Science & Engineering-Data Science

CERTIFICATE

This is to certify that the work embodies in this Minor Project-II entitled **"ROAD LANE LINE DETECTION"** being submitted by **"AVINASH GIRI" (0176CD211032) ,SAMDHISH YADAV (0176CD211104),PUSHPENDRA CHOUDHARY(0176CD211090) AND VIRAT RAGHUWANSHI (0176CD211133)"** for partial fulfillment of the requirement for the award of degree of "**Bachelor of Technology** in **Computer Science & Engineering-Data Science"** discipline to "**RAJIV GANDHI PROUDYOGIKI VISHWAVIDYALAYA, BHOPAL (M.P.)**" during the academic year 2024-25 is a record of real piece of work, carried out by him/her/them under my supervision and guidance in the **"Department of Computer Science & Engineering-Data Science", Lakshmi Narain College of Technology Excellence, Bhopal (M.P.)**

SUPERVISED BY

Prof. Satish Choudhary

Department of CSE-Data Science

LNCTE, Bhopal

FORWARDED BY

Prof. (Dr.) Neetesh Kumar Gupta

Head, Department of CSE-Data Science)

LNCTE, Bhopal

APPROVED BY

Dr. Anil Kumar Saxena Principal

LNCTE, Bhopal

ABSTRACT

Lane detection is a challenging problem. It has attracted the attention of the computer vision community for several decades. Essentially, lane detection is a multi- featured detection problem that has become a real challenge for computer vision and machine learning techniques. Although many machine learning methods are used for lane detection, they are mainly used for classification rather than feature design. But modern machine learning methods can be used to identify the features that are rich in recognition and have achieved success in feature detection tests. However, these methods have not been fully implemented in the efficiency and accuracy of lane detection. In this paper, we propose a new method to solve it. We introduce a new method of pre-processing and ROI selection. The main goal is to use the HSV colour transformation to extract the white features and add preliminary edge feature detection in the pre-processing stage and then select ROI based on the proposed preprocessing. This new pre-processing method is used to detect the lane. By using the standard KITTI road database to evaluate the proposed method, the results obtained are superior to the existing preprocessing and ROI selection techniques.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABBREVIATION	EXPANSIONS
ACC	Adaptive Cruise Control
HSV	Hue Saturation Value
ITS	Intelligent Transportation System
LDW	lane departure warning
ROI	Region of Interest
RGB	Red Green Blue

CHAPTER 1
INTRODUCTION

With the rapid development of society, automobiles have become one of the transportation tools for people to travel. In the narrow road, there are more and more vehicles of all kinds. As more and more vehicles are driving on the road, the number of victims of car accidents is increasing every year. How to drive safely under the condition of numerous vehicles and narrow roads has become the focus of attention. Advanced driver assistance systems which include lane departure warning (LDW), Lane Keeping Assist, and Adaptive Cruise Control (ACC) can help people analyse the current driving environment and provide appropriate feedback for safe driving or alert the driver in dangerous circumstances. This kind of auxiliary driving system is expected to become more and more perfect. However, the bottleneck of the development of this system is that the road traffic environment is difficult to predict. After investigation, in the complex traffic environment where vehicles are numerous and speed is too fast, the probability of accidents is much greater than usual. In such a complex traffic situation, road colour extraction and texture detection as well as road boundary and lane marking are the main perceptual clues of human driving.

Lane detection is a hot topic in the field of machine learning and computer vision and has been applied in intelligent vehicle systems. The lane detection system comes from lane markers in a complex

environment and is used to estimate the vehicle's position and trajectory relative to the lane reliab. At the same time, lane detection plays an important role in the lane departure warning system. The lane detection task is mainly divided into two steps: edge detection and line detection.

Qing et al. proposed the extended edge linking algorithm with directional edge gap closing. The new edge could be obtained with the proposed method. Mu and Ma proposed Sobel edge operator which can be applied to adaptive area of interest (ROI. However, there are still some false edges after edge detection. Wang et al. proposed a Canny edge detection algorithm for feature extraction. The algorithm provides an accurate fit to lane lines and could be adaptive to complicated road environment. In 2014, Srivastava et al. proposed that the improvements to the Canny edge detection

can effectively deal with various noises in the road environment. Sobel and Canny edge operator are the most commonly used and effective methods for edge detection.

Line detection is as important as edge detection in lane detection. With regard to line detection, we usually have two methods which include feather-based method and model-based methods.

Niu et al. used a modified Hough transform to extract segments of the lane profile and used DBSCAN (density based spatial application noise clustering) clustering algorithm for clustering. In 2016, Mammeri et al. used progressive probabilistic Hough transform combined with maximum stable extreme area technology to identify and detect lane lines and utilized Kalman filter to achieve continuous tracking. However, the algorithm does not work well at night.

we propose a lane detection method that is suitable for all kinds of complex traffic situations, especially as driving speed in roads is too fast. First, we pre-processed each frame image and then selected the area of interest (ROI) of the processed images. Finally, we only needed edge detection vehicle and line detection for the ROI area. In this study, we introduced a new preprocessing method and ROI selection method. First, in the preprocessing stage, we converted the RGB colour model to the HSV colour space model and extracted white features on the HSV model. At the same time, the preliminary edge feature detection is added in the preprocessing stage, and then the part below the image is selected as the ROI area based on the proposed preprocessing. Compared with the existing methods, the existing preprocessing methods only perform operations such as graying, blurring, X-gradient, Y-gradient, global gradient, thresh, and morphological closure. And the ways to select the ROI area are also very different. Some of them are based on the edge feature of the lane to select the ROI area, and some are based on the colour feature of the lane to select the ROI area. These existing methods do not provide accurate and fast lane information, which increases the difficulty of lane detection.

Traffic accidents have become one of the most serious problems in today's world. Roads are the mostly chosen modes of transportation and provide the finest connections among all modes. Most frequently occurring traffic problem is the negligence of the drivers and it has become more and more serious with the increase of vehicles. These road accidents can be reduced with the help of road lanes or white markers that assist

the driver to identify the road area and nonroad area. A lane is a part of the road marked which can be used by a single line of vehicles as to control and guide drivers so that the traffic conflicts can be reduced. Increasing the safety and saving lives of human beings is one of the basic function of Intelligent Transportation System (ITS). Intelligent transportation systems are advanced applications which aim to provide innovative services relating to different modes of transport and traffic management. This system enables various users to be better informed and make safer, more coordinated, and smarter use of transport networks. These road accidents can be reduced with the help of road lanes or white markers that assist the driver to identify the road area and non-road area. A lane is a part of the road marked which can be used by a single line of vehicles as to control and guide drivers so that the traffic conflicts can be reduced. Most roads such as highways have at least two lanes, one for traffic in each direction, separated by lane markings. Major highways often have two roadways separated by a median, each with multiple lanes. To detect these road lanes some system must be employed that can help the driver to drive safely. Lane Line detection is a critical component for self-driving cars and also for computer vision in general. This concept is used to describe the path for self-driving cars and to avoid the risk of getting in another lane. Using computer vision techniques in Python, we will identify road lane lines in which autonomous cars must run. This will be a critical part of autonomous car.

Lane lines are being drawn as the car drives. Also, you can see the radius of curvature is being calculated to help the car steer. It is cheap to equip cars with a front facing camera. Much cheaper than RADAR or LIDAR. Once we get a camera image from the front facing camera of self-driving car, we make several modifications to it.

CHAPTER 2

LITERATURE SURVEY

2.1 Proposed System

Lane detection is a process that is used to locate the lane markers on the road. With the help of this lane markers presents these locations to an intelligent system. This system decreases the road accidents and also helps to improves traffic conditions. Lane detection consists of specific types of primitives such as road markings etc. lane detection represents the margins of path into a single framework. It supports various applications like lane departure warning, lane keeping assists, lane centering etc. Lane departure warning gives us a warning when the vehicle is veering off the lane without signaling. Lane detection also plays an important role in advanced driver assistant system. This system helps the drivers in the driving process. This system is developed for safety and better driving. This system based upon vehicle to vehicle or vehicle to infrastructure system etc. Advanced driver assistance system consists of collision avoidance system, blind spot system and many more systems. In lane detection there are many approaches that are applied like feature based and model based.

Feature based approach are used to detect edges and model based approach is a type of curve model

The lane detection technology to improve the efficiency and accuracy of real-time lane detection. the lane detection module is usually divided into two steps: (1) image preprocessing and (2) the establishment and matching of line lane detection model.

The overall diagram of our proposed system where lane detection blocks are the main contributions of this paper. The first step is to read the frames in the video stream. The second step is to enter the image preprocessing module. What is different from others is that in the preprocessing stage we not only process the image itself but also do colour feature extraction and edge feature extraction. In order to reduce the influence of noise in the process of motion and tracking, after extracting the colour features of the image, we need to use Gaussian filter to smooth the image. Then, the image is obtained by binary threshold processing and morphological closure.

These are the preprocessing methods mentioned in this project. Next, we select the adaptive area of interest (ROI) in the preprocessed image. The last step is lane

detection. Firstly, Canny operator is used to detect the edge of lane line; then Hough transform is used to detect line lane. Finally, we use Extended Kalman Filter to detect and track lane line in real time.

2.2 Existing System and Limitation of the Existing System

In order to reduce the influence of noise in the process of motion and tracking, after extracting the colour features of the image, we need to use Gaussian filter to smooth the image. Then, the image is obtained by binary threshold processing and morphological closure.

The existing approaches are failed to detect the lanes which are degraded with sand hid, tree shadows and high illumination condition. The proposed method is implemented to resolve the afore mentioned drawbacks encountered in existing approaches.

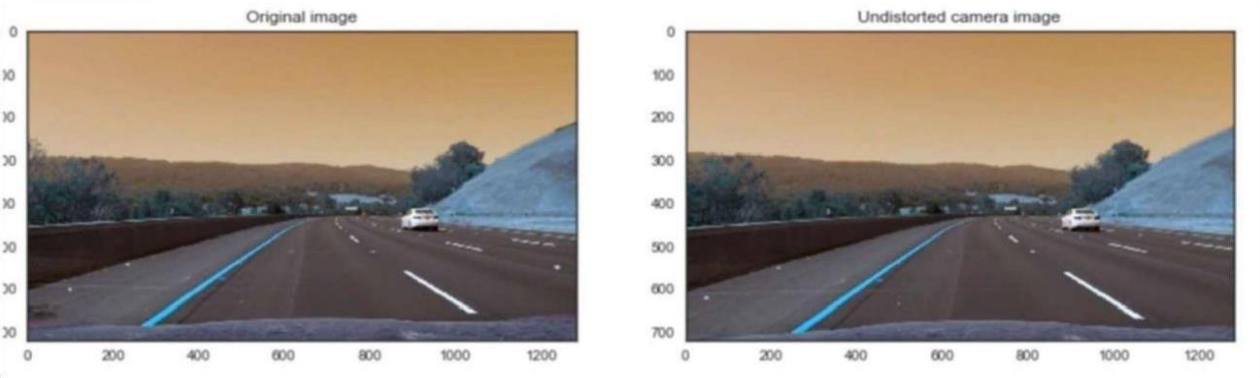


Fig .1 -Describing the original image and undistorted image

Sobel Operator

After smoothing the image and eliminating the noise, the next step is to find the edge strength by taking the gradient of the image. The Sobel operator performs a 2D spatial gradient measurement on an image.

Then, the approximate absolute gradient magnitude (edge strength) at each point can be found by the formula which is simpler to calculate compared to the above exact gradient magnitude.

Approximate gradient magnitude:

$$|G| = |G_x| + |G_y|$$

The Sobel operator uses a pair of 3x3 convolution masks, one estimating the gradient in the x-direction (columns) and the other estimating the gradient in the ydirection (rows).

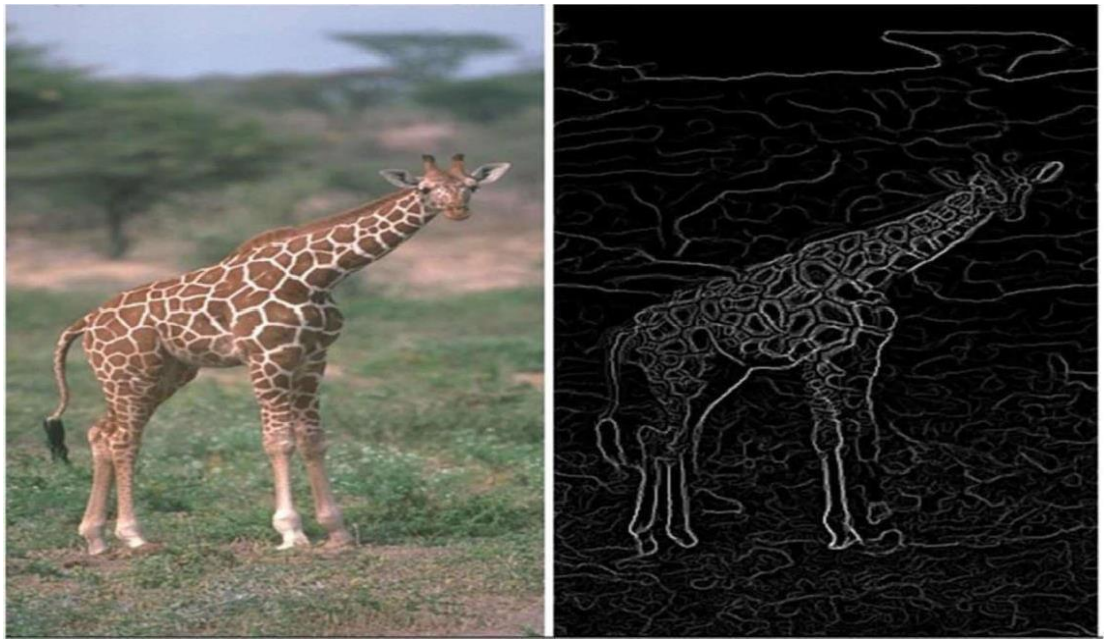


Fig.2- convert image to gray scale

DRAW BACKS

In poor illumination scenes, lane markings are unobvious and noises may appear on the image. Moreover, vehicle headlights, tail lights, road lamps and reflected light appear in scenes and will disturb the detection result.

The Hough transform is only efficient if a high number of votes fall in the right bin, so that the bin can be easily detected amid the background noise. This means that the bin must not be too small, or else some votes will fall in the neighboring bins, thus reducing the visibility of the main bin

CHAPTER 3

REQIREMENT ANALYSIS

3.1 Feasibility Study

Feasibility studies and risk analysis are two important aspects of project planning and management. feasibility study is an assessment of the viability of a proposed project, while risk analysis involves identifying and assessing potential risks that may impact the project.

Market Feasibility: Identify the market potential for the product or service that the project aims to offer. This includes conducting market research and analyzing consumer demand.

Technical Feasibility: Determine the technical feasibility of the project by analyzing the available resources, such as technology, skills, and equipment.

Financial Feasibility: Analyze the financial feasibility of the project by estimating the costs and revenue streams associated with the project.

Legal Feasibility: Evaluate the legal and regulatory requirements that the project must comply with, such as licenses, permits, and environmental regulations.

Risk Analysis

Identify Risks: Identify potential risks that may affect the project, such as technical, financial, legal, or environmental risks.

Assess Risks: Assess the likelihood and impact of each identified risk on the project's objectives, timeline, and budget.

Mitigate Risks: Develop strategies to mitigate or reduce the impact of identified risks. This includes contingency plans, risk transfer, risk avoidance, or risk sharing.

Monitor Risks: Continuously monitor the identified risks throughout the project life cycle and update the risk management plan accordingly.

The feasibility study and risk analysis help project managers to evaluate the viability and potential challenges of the project before investing resources. By considering both aspects, project managers can identify the most feasible and sustainable projects while minimizing potential risks.

3.2 Software requirements specification document for lane detection

A software requirements specification document for lane detection would typically include the following sections:

Introduction: This section provides an overview of the software requirements specification document and the purpose of the lane detection system.

Scope: This section defines the scope of the lane detection system, including the specific features and functionalities that the system should provide.

3.2.1. Functional Requirements

This section describes the functional requirements of the lane detection system. This includes the following:

Image and Video Input: The system should be able to accept input from a camera or video stream.

Lane Detection: The system should be able to detect lane markings on the road using image processing techniques.

Lane Tracking: The system should be able to track the detected lanes in real-time to provide information about the vehicle's position relative to the lane.

Lane Departure Warning: The system should be able to provide a warning to the driver if the vehicle deviates from the lane.

Calibration: The system should allow for calibration of camera or sensor parameters to ensure accurate lane detection.

3.2.2. Non-functional Requirements

This section describes the non-functional requirements of the lane detection system. This includes the following:

Performance: The system should be able to process images or video streams in real-time to provide accurate and timely feedback to the driver.

Accuracy: The system should be able to accurately detect and track lane markings under different lighting and weather conditions.

Reliability: The system should be able to operate reliably without crashes or errors.

Security: The system should protect against unauthorized access or manipulation of the input or output data.

3.2.3. Hardware and Software Requirements

This section describes the hardware and software requirements of the lane detection system. This includes the required camera or sensor specifications, processing power, and any necessary software libraries.

Assumptions and Constraints: This section identifies any assumptions or constraints that may impact the development or operation of the lane detection system.

User Interfaces: This section describes the user interface of the lane detection system, including any input or output formats.

System Architecture: This section describes the overall system architecture of the lane detection system, including the algorithms and data structures that will be used.

By including these sections in the software requirements specification document, the development team can ensure that the lane detection system meets the requirements of the stakeholders and operates reliably in real-world condition.

3.3 SYSTEM USECASE

A system use case is a description of how a user interacts with a system to accomplish a specific goal or task. It typically includes a series of steps or actions that the user takes to achieve a desired outcome. Here is an example of a system use case for a lane detection system:

Use Case: Lane Departure Warning System

Primary Actor: Driver

Goal: To receive a warning if the vehicle deviates from the lane

Preconditions:

- The lane detection system is installed and operational.
- The vehicle is in motion.
- The lane markings are clearly visible and distinguishable from the surrounding environment.

Basic Flow:

- The driver starts the vehicle and initiates the lane detection system.

- The system captures an image or video stream from the camera or sensor.
- The system processes the image or video stream using lane detection algorithms.
- The system detects the lane markings on the road and tracks the vehicle's position relative to the lane.

If the vehicle deviates from the lane, the system provides a warning to the driver, such as a visual or auditory signal.

The driver takes corrective action to return the vehicle to the center of the lane.

The system continues to monitor the vehicle's position relative to the lane and provides additional warnings if necessary.

Alternate Flow

1a. If the camera or sensor fails to capture a clear image or video stream, the system notifies the driver and prompts them to take corrective action, such as cleaning the camera lens or adjusting the camera angle.

2a. If the lane markings are obscured or difficult to distinguish from the surrounding environment, the system may not provide a warning to the driver.

3a. If the driver ignores the warning and does not take corrective action, the system may escalate the warning, such as by increasing the volume or frequency of the alert.

Post conditions:

The lane detection system continues to operate and monitor the vehicle's position relative to the lane. The driver is able to maintain a safe and consistent position within the lane.

CHAPTER 4

DESCRIPTION OF PROPOSED SYSTEM

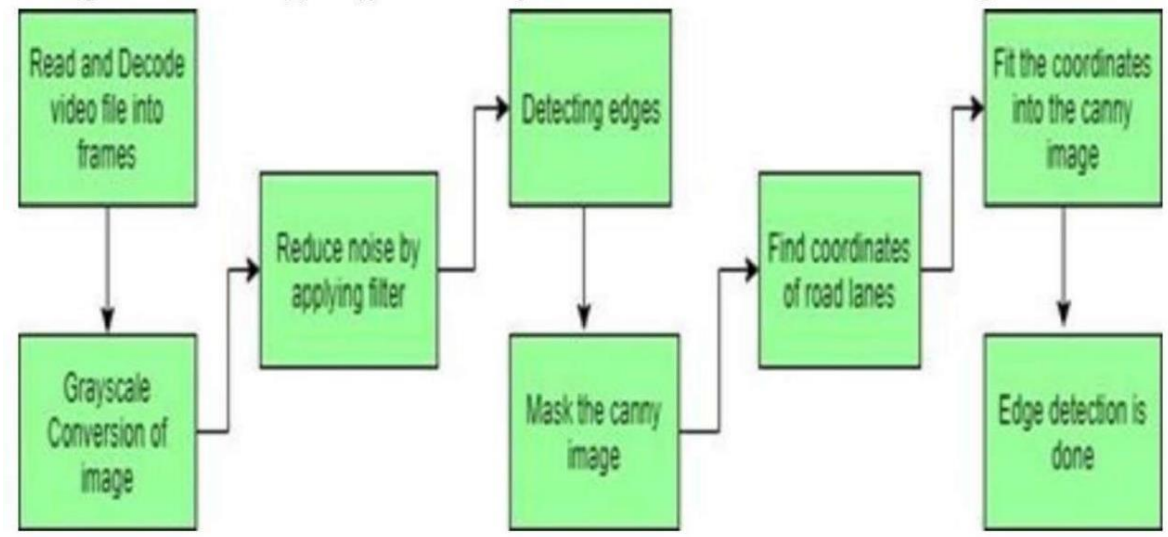
METHODOLOGY

4.1 MODULES

Pre-processing is a common name for operations with images at the lowest level of abstraction of both input and output are intensity images. The aim of pre-processing is an improvement of the image data that suppresses unwanted distortion.

Some of the point processing techniques include: contrast stretching, global thresholding, histogram equalization, log transformations and power law transformations. Some mask processing techniques include averaging filters, sharpening filters, local thresholding… etc.

The most important process in the project. Single Image from testing dataset is taken such a way that it reaches our implementation of a model. Each model we implement takes a resultant image as an input and process it further to produce an output. This selection of image is more important because implementation of each model requires an image input for processing. And if the processing is done, then output is produced. If output produced for the testing image is same as required, then the resultant image is sent to next process that we need to develop further. In order to observe the clear output, the best suitable image should be selected such a way that the testing image should be able to produce the clear required output at the end of processes



4.1.1 MORPHOLOGY

Morphological image processing is a collection of non-linear operations related to the shape or morphology of features in an image. morphological operations rely only on the relative ordering of pixel values, not on their numerical values, and therefore are especially suited to the processing of binary images.

Morphological operations can also be applied to greyscale images such that their light transfer functions are unknown and therefore their absolute pixel values are of no or minor interest. A morphological operation on a binary image creates a new binary image in which the pixel has a nonzero value only if the test is successful at that location in the input image.

4.1.2 NORMALIZATION

To remove any non-uniformities, present in the imaging system, dark field image subtraction and bright field image normalization were performed.

A dark field image (or background image) for each run was taken immediately before the laser was fired

Pre-processing plays a major role in producing the required output in sufficient required amount of time. Pre-processing of selected image mainly undergo the gray scale convert converted into gray scale through the open source computer vision package. And then smoothening is applied by implementing the Gaussian Blur algorithm on the selected gray scale image. A gray scale image mainly consists of change in variants from white to black that represents the color mixes of red, green and blue. The normalization is main process of Gaussian Blur process conversion which is done through multiplying each intensities of a pixels by their corresponding normalized matrix values. Thus, pre-processing is done on the selected image. This conversion of gray scale image and reducing noise in the image can help by reducing the processing time in the next large processes. In machine learning projects in general, you usually go through a data pre-processing or cleaning step.

The goal of this step is to make your data ready for the ML model to make it easier to analyze and process computationally, as it is with images. Based on the problem you're solving and the dataset in hand, there's some data massaging required b you feed your images to the ML model.

Image processing could be simple tasks like image resizing. In order to feed a dataset of images to a convolutional network, they must all be the same size. Other processing tasks can take place like geometric and colour transformation or converting colour to grayscale and many more.

The acquired data are usually messy and come from different sources. To feed them to the ML model (or neural network), they need to be standardized and cleaned up. More often than not, pre-processing is used to conduct steps that reduce the complexity and increase the accuracy of the applied algorithm. We an image, we tend to convert it into a form that allows a general algorithm to solve it

4.1.3 Grey Scale Image Conversion

Convert color to gravy scale to reduce completion complexity .in certain problems you ’ll it useful to lose unnecessary information from your images to reduce space or computational complexity. For example, converting your colored images to grayscale images. This is because in many objects,

Color isn’t necessary to recognize and interpret an image. Grayscale can be good enough for recognizing certain objects. Because color images contain more information than black and white images, they can add unnecessary complexity and take up more space in memory (Remember how color images are represented in three channels, which means that converting it to grayscale reduces the number of pixels that need to be processed).

4.1.4 EDGE DETECTION

Edge detection is an image processing technique for finding the boundaries of objects within images. It works by detecting discontinuities in brightness. Edge detection is used for image segmentation and data extraction in areas such as image processing, computer vision, and machine vision. Edge detection allows users to observe the features of an image for a significant change in the gray level.

Based on these criteria, the canny edge detector first smoothens the image to eliminate noise. It then finds the image gradient to highlight regions with high spatial derivatives.

The algorithm then tracks along these regions and suppresses any pixel that is not at the maximum using non-maximum suppression. The gradient array is now further reduced by hysteresis to remove streaking and thinning the edges

4.1.5 Hough Transformations

Hough Transformations require a Hough transformation space which is used to rotate the angles of a trigonometric line equation and then specify the lines present in an edge detected image. If the trigonometric line in rotation meets the edges in the image, then it may consider for applying the model trained for detecting roads in an image. The training is done in Hough transformation space which is used to detect the actual road lines from an image. When the Hough transformations and training is done, then the road lines are detected on the selected image. The training of model is improvised until the correct output is observed from a selected image. The Hough transform is a technique which can be used to isolate features of a particular shape within an image. Because it requires that the desired features be specified in some parametric form, the classical Hough transform is most commonly used for the detection of regular curves such as lines, circles, ellipses, etc. A generalized Hough transform can be employed in applications where a simple analytic description of a feature(s) is not possible. Due to the computational complexity of the generalized Hough algorithm, we restrict the main focus of this discussion to the classical Hough transform. Despite its domain restrictions, the classical Hough transform (hereafter referred to without the classical prefix) retains many applications, as most manufactured parts (and many anatomical parts investigated in medical imagery) contain feature boundaries which can be described by regular curves. The main advantage of the Hough transform technique is that it is tolerant of gaps in feature boundary descriptions and is relatively unaffected by image noise.

The Hough technique is particularly useful for computing a global description of a feature(s) (where the number of solution classes need not be known a priori), given (possibly noisy) local measurements. The motivating idea behind the Hough technique for line detection is that each input measurement (e.g. coordinate point) indicates its contribution to a globally consistent solution (e.g. the physical line which gave rise to that image point).

As a simple example, consider the common problem of fitting a set of line segments to a set of discrete image points (e.g. pixel locations output from an edge detector). shows some possible solutions to this problem. Here the lack of a priori knowledge about the number of desired line segments (and the ambiguity about what constitutes a line segment) render this problem under-constrained.

We can analytically describe a line segment in a number of forms. However, a convenient equation for describing a set of lines uses parametric or normal notion:

$X \cos t + Y \sin t = r$

4.1.7 SEGMENTATION

Image segmentation is a method in which a digital image is broken down into various subgroups called Image segments which helps in reducing the complexity of the image to make further processing or analysis of the image simpler. Segmentation in easy words is assigning labels to pixels. All picture elements or pixels belonging to the same category have a common label assigned to them.

4.1.8 THRESHOLDING

Automatic thresholding is a great way to extract useful information encoded into pixels while minimizing background noise. This is accomplished by utilizing a feedback loop to optimize the threshold value before converting the original grayscale image to binary. The idea is to separate the image into two parts; the background and foreground.

4.2 SYSTEM ARCHITECTURE

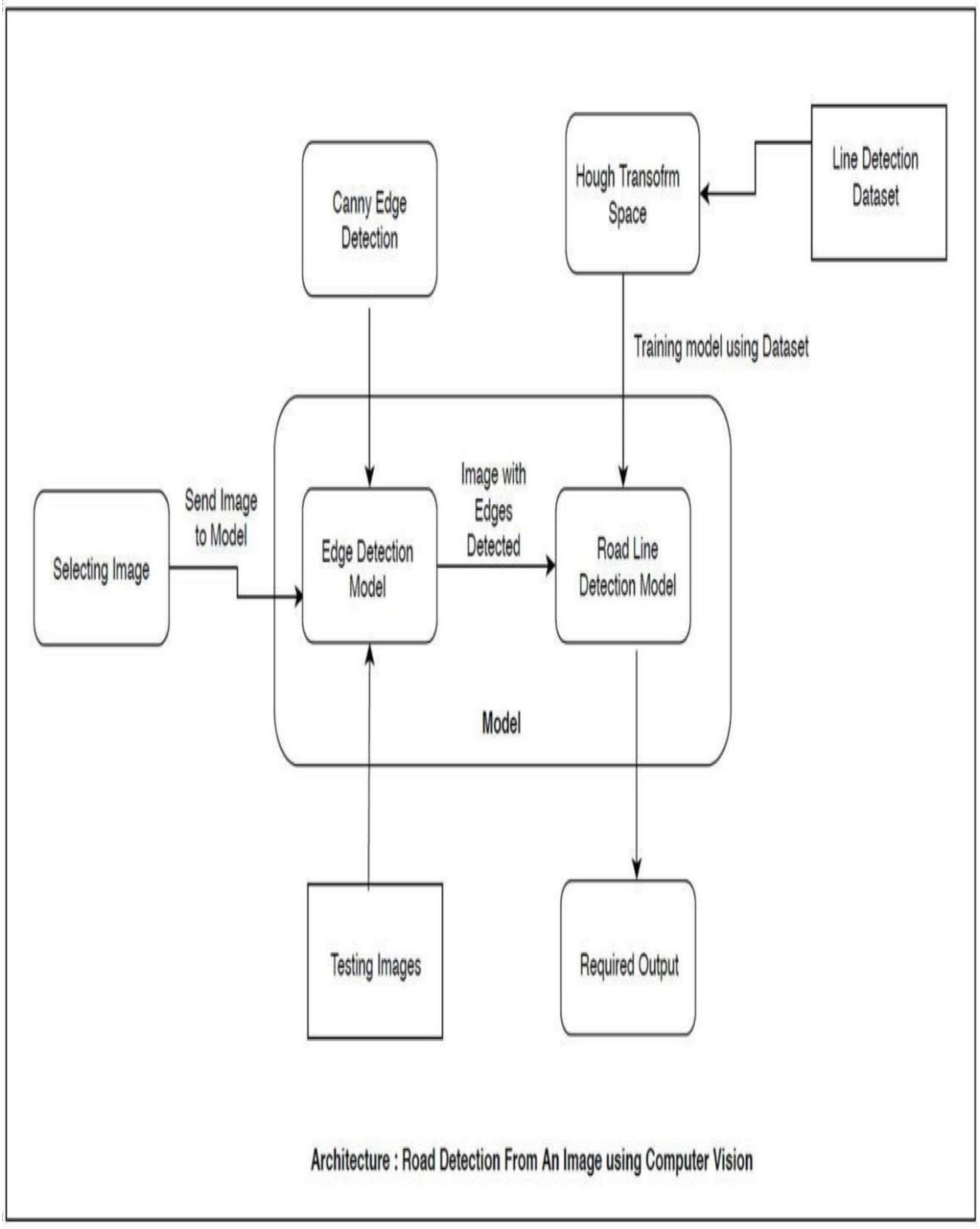
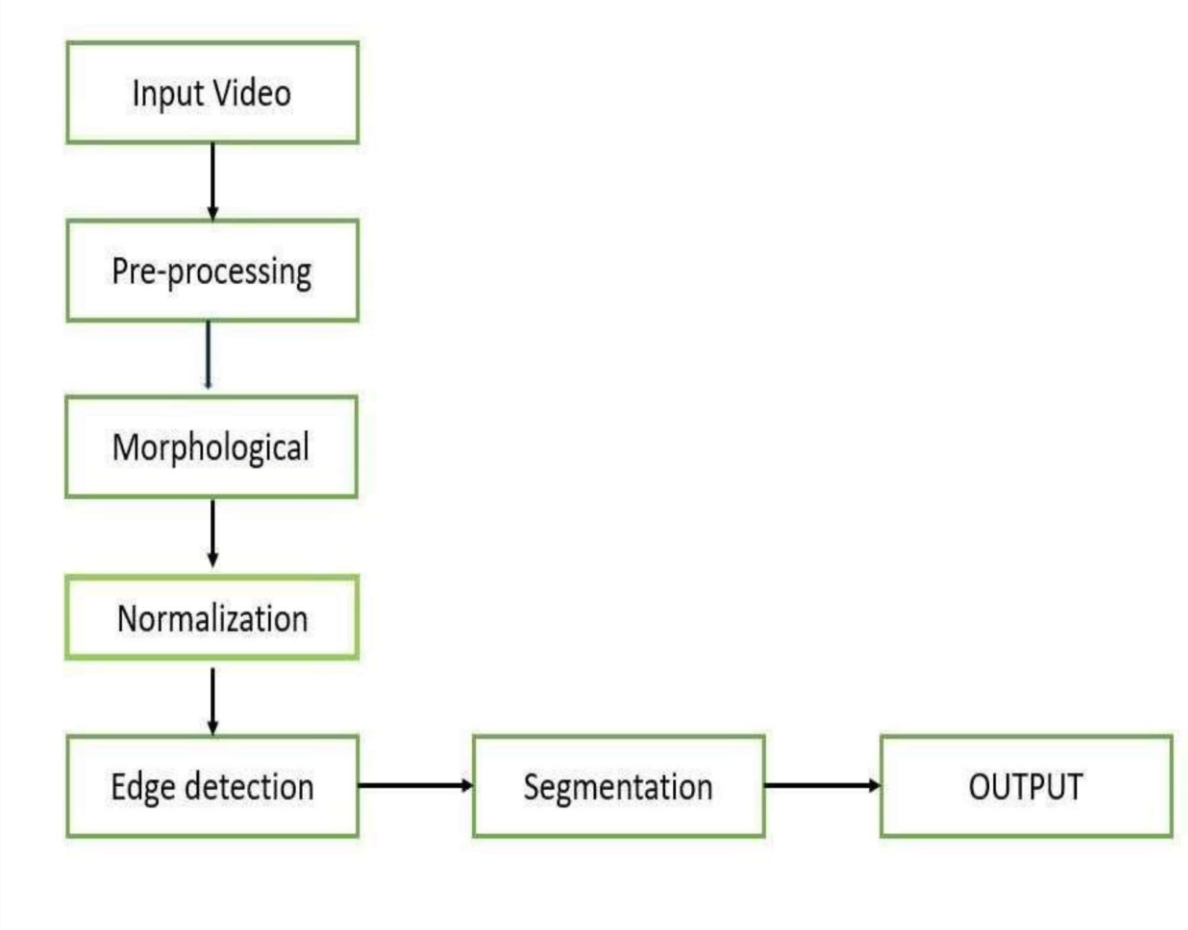


Fig.3: Road Detection from an Image



. ***Fig. 4: flow work***

4.3 Description of Software for Implementation and Testing plan of the Proposed Model/System

The software for implementation and testing plan of the proposed model/system for lane detection would typically involve several components, such as:

Image Processing Libraries: Image processing libraries such as OpenCV or PIL would be required for image processing tasks such as converting images to grayscale, edge detection, thresholding, etc.

Machine Learning Libraries: Machine learning libraries such as Tensor Flow, Keras, or PyTorch would be required for implementing the deep learning model for lane detection.

Dataset: A large dataset of images and labeled lanes would be required for training the deep learning model.

Development Environment: A development environment such as Anaconda, Jupyter Notebook, or PyCharm would be required for developing and testing the code.

Code Implementation: The code implementation would involve designing the architecture of the deep learning model, preprocessing the images, training the model, and testing the model on new images.

Testing Plan: A testing plan would be required to test the accuracy and robustness of the deep learning model. The testing plan would typically involve testing the model on a large dataset of images, testing the model on real-time video streams, and evaluating the performance of the model on different lighting conditions and road surfaces.

Performance Metrics: Performance metrics such as accuracy, precision, recall, and F1-score would be used to evaluate the performance of the deep learning model.

Overall, the software for implementation and testing plan of the proposed model/system for lane detection would involve a combination of image processing and machine learning techniques, along with a comprehensive testing plan to ensure the accuracy and robustness of the deep learning model.

4.4 PROJECT MANAGEMENT PLAN

Image Processing in Python: Algorithms Tools, and Methods

Images define the world, each image has its own story, it contains a lot of crucial information that can be useful in many ways. This information can be obtained with the help of the technique known as Image Processing.

It is the core part of computer vision which plays a crucial role in many real-world examples like robotics, self-driving cars, and object detection. Image processing allows us to transform and manipulate thousands of images at a time and extract useful insights from them. It has a wide range of applications in almost every field.

Python is one of the widely used programming languages for this purpose. Its amazing libraries and tools help in achieving the task of image processing very efficiently.

Through this article, you will learn about classical algorithms, techniques, and tools to process the image and get the desired output.

As the name says, image processing means processing the image and this may include many different techniques until we reach our goal. The final output can be either in the

form of an image or a corresponding feature of that image. This can be used for further analysis and decision making.

An image can be represented as a 2D function F(x,y) where x and y are spatial coordinates. The amplitude of F at a particular value of x,y is known as the intensity of an image at that point. If x,y, and the amplitude value is finite then we call it a digital image. It is an array of pixels arranged in columns and rows. Pixels are the elements of an image that contain information about intensity and color. An image can also be represented in 3D where x,y, and z become spatial coordinates.

Pixels are arranged in the form of a matrix. This is known as an RGB image.

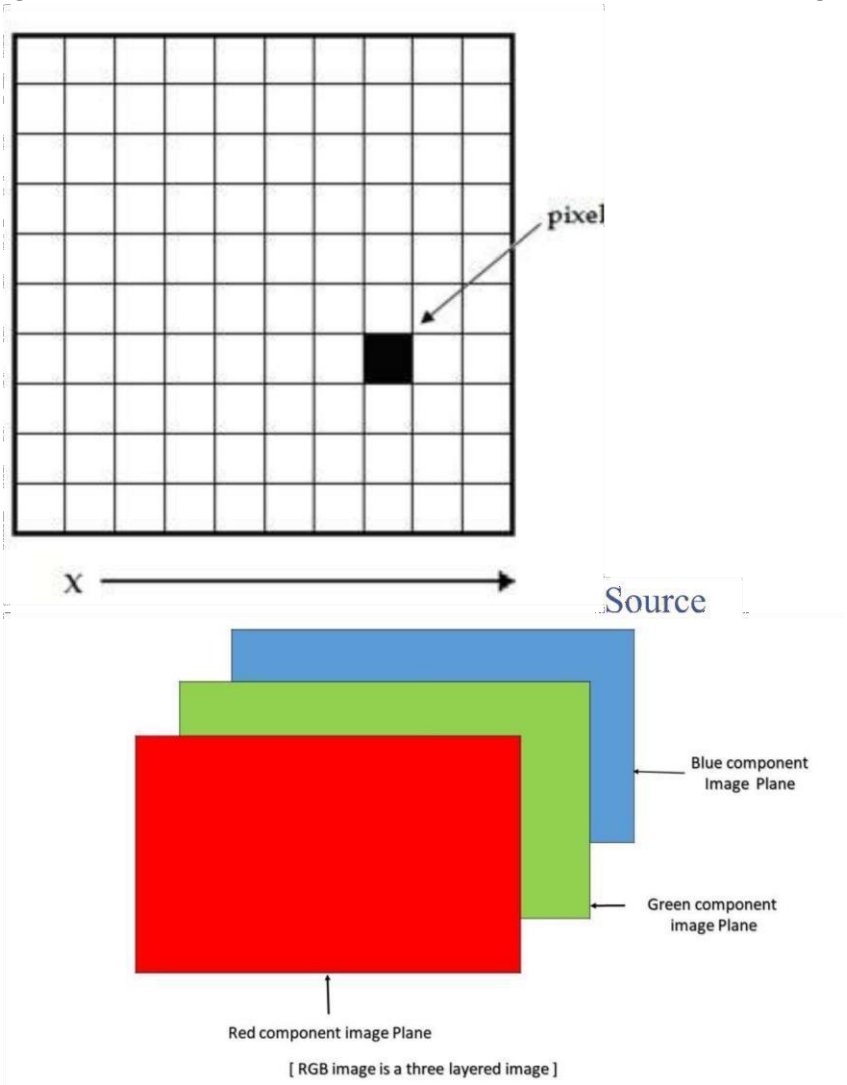


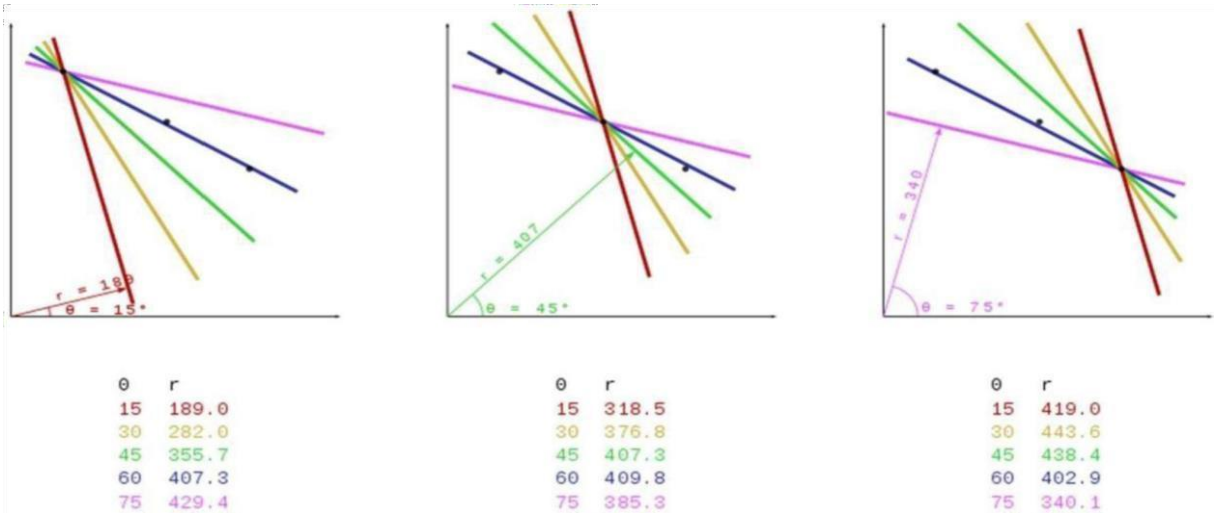
Fig 5 RGB image

4.4.1 Morphological Image Processing

Morphological image processing tries to remove the imperfections from the binary images because binary regions produced by simple thresholding can be distorted by noise. It also helps in smoothing the image using opening and closing operations.

Morphological operations can be extended to grayscale images. It consists of non-linear operations related to the structure of features of an image. It depends on the related ordering of pixels but on their numerical values. This technique analyzes an image using a small template known as structuring element which is placed on different possible locations in the image and is compared with the corresponding neighborhood pixels. A structuring element is a small matrix with 0 and 1 values.

The number of pixels removed or added to the original image depends on the size of the structuring element. It is positioned at all possible locations in the image and it is compared with the corresponding neighborhood of pixels.



θ	r
15	189.0
30	282.0
45	355.7
60	407.3
75	429.4

θ	r
15	318.5
30	376.8
45	407.3
60	409.8
75	385.3

θ	r
15	419.0
30	443.6
45	438.4
60	402.9
75	340.1

4.4.2 Colour Transform

Colour model transform is an important part of machine vision, and it is also an indispensable part of lane detection in this paper. The actual road traffic environment and light intensity all produce noise that interferes with the identification of colour. We cannot detect the separation of white lines, yellow lines, and vehicles from the background. The RGB colour space used in the video stream is extremely sensitive to light intensity, and the effect of processing light at different times is not ideal. In this

paper, the RGB sequence frames in the video stream are colour converted into HSV colour space images. the images of RGB colour space and HSV colour space, respectively. HSV represents hue, saturation, and value.

the values of white and yellow colours are very bright in the V-component compared to other colours and are easily extracted, providing a good basis for the next colour extraction. Experiments show that the colour processing performed in the HSV space is more robust to detecting specific targets.

Fig 6. RGB to HSV

4.4.3 Adding Edge Detection in Preprocessing

Edge detection is an image processing technique for finding the boundaries of objects within images. It works by detecting discontinuities in brightness This could be very beneficial in extracting useful information from the image because most of the shape information is enclosed in the edges. Classic edge detection methods work by detecting discontinuities in the brightness.

It can rapidly react if some noise is detected in the image while detecting the variations of grey levels. Edges are defined as the local maxima of the gradient. It has carried out edge detection two times successively; the first time is to perform a wide range of edge detection extraction in the entire frame image. In the second, the edge detection is performed again after the lane detection after ROI selection. This detection further improves the accuracy of lane detection. This section mainly performs the overall edge detection on the frame image, using the improved Canny edge detection algorithm. The concrete steps of Canny operator edge detection are as follows: First, we use a Gaussian filter to smooth the image (preprocessed image), and then we use the Sobel operator to calculate the gradient magnitude and direction. Next step is to suppress the non-maximal value of the gradient amplitude. Finally, we need to use a double-threshold algorithm to detect and connect edges. the image after extraction with Canny edge detection

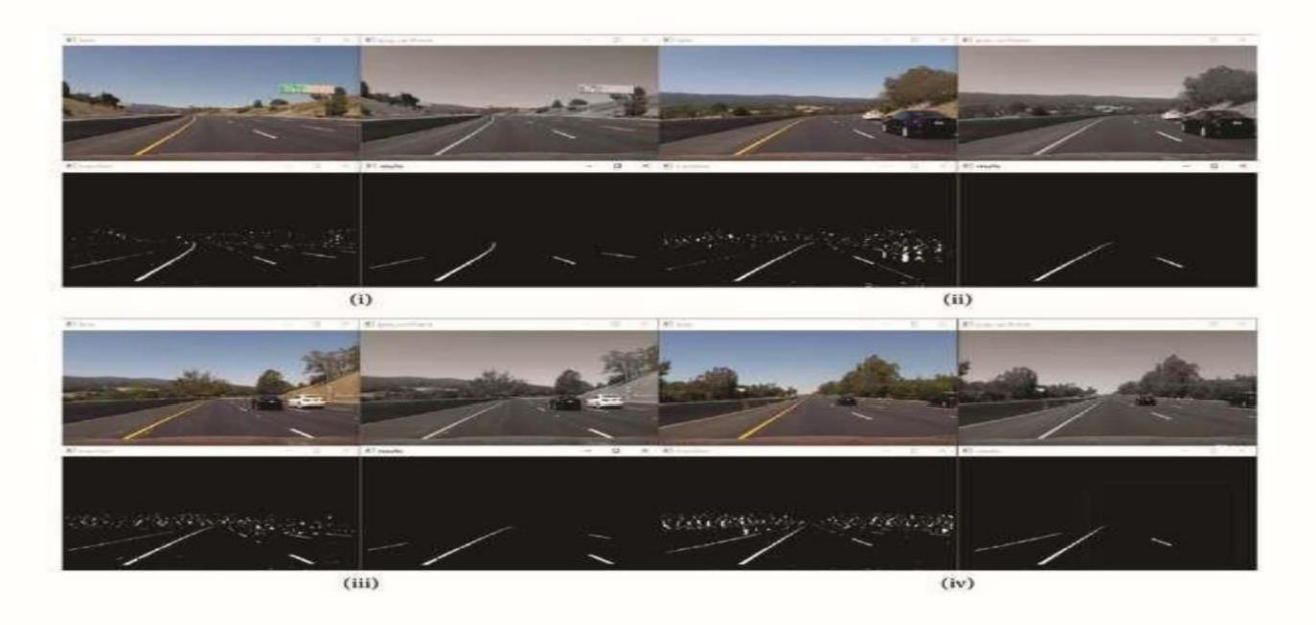


Fig .7 Edge detection

4.4.4 ROI SELECTION

After edge detection by Canny edge detection, we can see that the obtained edge not only includes the required lane line edges, but also includes other unnecessary lanes

and the edges of the surrounding fences. The way to remove these extra edges is to determine the visual area of a polygon and only leave the edge information of the visible area. The basis is that the camera is fixed relative to the car, and the relative position of the car with respect to the lane is also fixed, so that the lane is basically kept in a fixed area in the camera.

In order to lower image redundancy and reduce algorithm complexity, we can set an adaptive area of interest (ROI) on the image [19]. We only set the input image on the ROI area and this method can increase the speed and accuracy of the system. we use the standard road database. We divide the image of each frame in the running video of the vehicle into two parts, and one-half of the lower part of the image frame serves as the ROI area. Figure 9 shows the ROI selection of sample frames (a), (b), (c), and (d) which are processed by the proposed preprocessing. The images of the four different sample frames have been able to substantially display the lane information after being processed by the proposed preprocessing method, but not only the lane information but also a lot of nonlane noise is present in the upper half of the image. So we cut out the lower half of the image (one-half) as the ROI area.

Fig .8 ROI selection

4.4.5 FUNDAMENTAL STEPS IN DIGITAL IMAGE PROCESSING

1. IMAGE ACQUISITION

This is the first step or process of the fundamental steps of digital image processing. Image acquisition could be as simple as being given an image that is already in digital form. Generally, the image acquisition stage involves pre-processing, such as scaling etc

2. IMAGE ENHANCEMENT

Image enhancement is among the simplest and most appealing areas of digital image processing. Basically, the idea behind enhancement techniques is to bring out detail that is obscured, or simply to highlight certain features of interest in an image. Such as, changing brightness & contrast etc.

3. IMAGE RESTORATION

Image restoration is an area that also deals with improving the appearance of an image. However, unlike enhancement, which is subjective, image restoration is objective, in the sense that restoration techniques tend to be based on mathematical or probabilistic models of image degradation.

4. COLOR IMAGE PROCESSING:

Color image processing is an area that has been gaining its importance because of the significant increase in the use of digital images over the Internet. This may include color modelling and processing in a digital domain etc.

5. WAVELETS AND MULTIRESOLUTION PROCESSING

Wavelets are the foundation for representing images in various degrees of resolution. Images subdivision successively into smaller regions for data compression and for pyramidal representation.

6. COMPRESSION

Compression deals with techniques for reducing the storage required to save an image or the bandwidth to transmit it. Particularly in the uses of internet it is very much necessary to compress data.

7. REPRESENTATION AND DESCRIPTION

Representation and description almost always follow the output of a segmentation stage, which usually is raw pixel data, constituting either the boundary of a region or all the

points in the region itself. Choosing a representation is only part of the solution for transforming raw data into a form suitable for subsequent computer processing. Description deals with extracting attributes that result in some quantitative information of interest or are basic for differentiating one class of objects from another.

4.4.6 Shadow Edge Detection Method

On the basis of the new chrominance properties of shadows, a shadow detection method for onboard road detection is proposed. As long as onboard systems deal with still images, there is not a known non-shadowed reference road region in the incoming image to compare the pixel properties. Thus, the shadow detection method focuses on comparing pixel properties across image edges, where the darker region of an image edge is the candidate shadow region and the brighter one is assumed to be the non-shadowed reference region. The method comprises four main stages: extraction of the image edges, selection of the bright and dark regions across the edges, extraction of the strong edges, and shadow edge classification.

In the image edge extraction stage, we break T- and X-junctions that connect different edges, thus obtaining an edge map consisting of individual edges. In order to achieve robustness in the edge classification, we exploit regions across each edge instead of single pixels. we use two regions of pixels along both sides of the edges to compute the chrominance properties of the surface. Prior to edge classification, an intensity filtering is applied to eliminate noisy edges on the asphalt, thus retaining only the strong ones in the image. Finally, edge classification is carried out by verifying whether the regions on both sides of the strong edges satisfy the six constraints associated with the proposed three chrominance properties of shadow, thus classifying each image edge as a shadow edge or a material-change edge. Since the method addresses the detection of shadow edges on the road, in order to simplify a captured road scene, as well as reduce the number of false positive detections outside the road surface, an ROI in the incoming color images is defined on the road by using knowledge of the scene perspective and assuming flat road surface

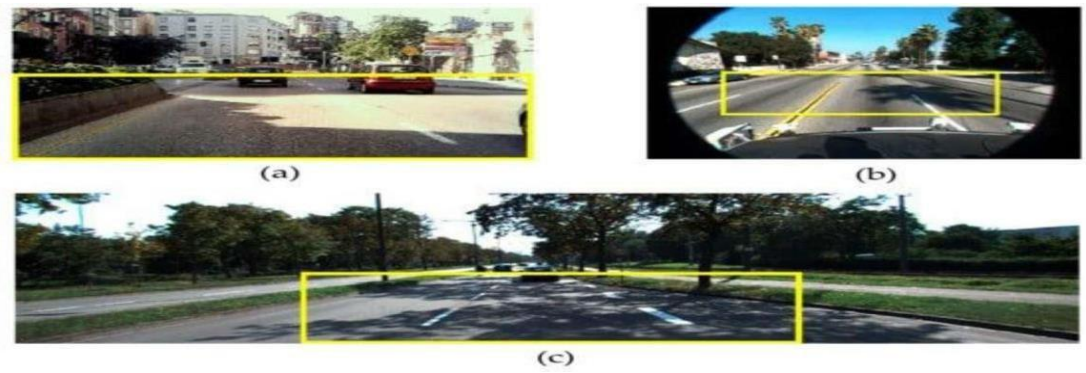


Fig 4. Shadow affected Road Lane-Line Images

After an averaging low-pass filtering to reduce image noise, the edges in the ROI of the incoming RGB image are extracted by applying the owing to its robustness. The edge map consists of edges due to both shadow boundaries and material changes. However, a troublesome effect of the edge extraction is the generation of T- and X-junctions, which affect the shadow edge classification since they connect different edge The edge classification requires separating edges into two regions only; thus, individual edges are generated by removing X- and T-junctions. To this end, the edge map is scanned bottom-up and a 3 × 3 kernel centered on each edge pixel is matched with a total of 18 T- and X-masks, For a positive match, a junction is broken by removing from the edge map the pixels involved in the junction. The result is an edge map consisting of individual edges that only separate two regions of the image.

Edge classification is the final stage of the method, where each individual edge D_k is classified as a shadow edge if the reflectance components of its darker (R_{sha}, G_{sha}, B_{sha}) and brighter ($R_{non\text{-}sha}$, $G_{non\text{-}sha}$, $B_{non\text{-}sha}$) regions satisfy the six constraints associated with the three chrominance properties of shadow, i.e., the six chrominance constraints. Otherwise, if one of the constraints is not satisfied, then the edge is classified as an edge due to a material change. Apply Gaussian filter to smooth the image and reduce noise. Compute gradient magnitude and orientation using Sobel, Scharr, or other derivative operators.

Apply non-maximum suppression to thin the edges and retain only the strongest edge pixels.

Apply hysteresis thresholding to obtain a binary edge map:

Define two threshold values: a high threshold T_high and a low threshold T_low.

Set all edge pixels with magnitude greater than T_high to be strong edges.

Set all edge pixels with magnitude between T_low and T_high to be weak edges.

Set all remaining pixels to be non-edges.

Trace edge contours starting from strong edges and including weak edges that are connected to strong edges.

Property 1. The relationship between the red and green surface reflectants due to sunlight is higher or equal to that due to skylight.

Property 2. The red component of the road reflectance due to sunlight is dominant, being higher than the blue, and higher or equal to the green one. In the same way, the green component of the road reflectance is higher than the blue one.

Property 3. The change in the red green proportion of the road reflectance due to skylight and sunlight is smaller than the change in the red blue one. This observation is also valid when comparing the changes of the surface relationships green red and green blue.

4.5 FINANCIAL REPORT ON EXTIMATED COSTING

To prepare a financial report on estimated costing for road lane detection, you should consider the following cost factors:

Development Team: The size and experience of your development team will affect the cost of your project. A larger and more experienced team will typically cost more.

Development Time: The amount of time required to develop the software will affect the cost of
your project. The longer the development time, the higher the cost.

Hardware and Software: You will need to consider the cost of hardware and software required to develop the software. This includes computers, servers, development tools, and third-party libraries.

Data Collection and Labeling: Collecting and labeling data is a critical part of developing a road lane detection system. You will need to consider the cost of collecting and labeling data, which can be expensive.

Machine Learning Model Development: The development of machine learning models requires specialized expertise and tools, which can add to the cost of the project.

Testing and Quality Assurance: You will need to test and validate your software to ensure that it performs as expected. This includes testing for accuracy, robustness, and

performance. Testing and quality assurance can be a significant cost factor in software development.

Maintenance and Support: You will need to provide maintenance and support for your software after it is deployed. This includes bug fixes, updates, and customer support.

Based on these factors, you can estimate the cost of your road lane detection project. To prepare a financial report, you should create a detailed breakdown of the costs associated with each factor mentioned above. You should also prepare a timeline for the project, including milestones and deliverables.

To give you an idea of the costs involved, here are some sample estimates:

Development Team: Depending on the size and experience of the team, the cost can range from $100,000 to $500,000 or more.

Development Time: Depending on the complexity of the project, the development time can range from 6 to 12 months or more.

Hardware and Software: The cost of hardware and software can range from $10,000 to $50,000 or more.

Data Collection and Labeling: Depending on the size of the dataset and the quality of the labeling, the cost can range from $50,000 to $100,000 or more.

Machine Learning Model Development: Depending on the complexity of the model, the cost can range from $50,000 to $200,000 or more.

Testing and Quality Assurance: Depending on the extent of testing required, the cost can range from $20,000 to $50,000 or more.

Maintenance and Support: The cost of maintenance and support can range from $10,000 to $50,000 or more per year.

Overall, the total estimated cost of a road lane detection project can range from $250,000 to $1,000,000 or more, depending on the complexity of the project and the size and experience of the development team.

4.6 SOFTWARE TO OPERATION PLAN:

Define project requirements: Start by defining the requirements for your road lane detection software. This includes defining the use cases, features, and functionality required for the system.

Develop a project plan: Develop a project plan that includes timelines, milestones, and deliverables. This plan should outline the steps required to develop, test, and deploy the software.

Determine the hardware and software requirements: Determine the hardware and software requirements for the software. This includes identifying the necessary computer hardware, sensors, and other components required for the software to operate.

Develop and test the software: Develop the software using your chosen programming language and tools. Test the software thoroughly to ensure that it performs as expected.

Train the machine learning models: Train the machine learning models using a large dataset of labeled images or video footage. This step is critical to ensure that the system accurately detects road lanes.

Deploy the software: Deploy the software on the desired hardware platform. This can include embedding the software in a vehicle or installing it on a standalone computer.

Conduct field testing: Conduct field testing to ensure that the software operates correctly under real-world conditions. This can include testing the system on different types of roads, in different weather conditions, and at different speeds.

Provide ongoing maintenance and support: Provide ongoing maintenance and support for the software. This includes fixing bugs, updating the software, and providing customer support as needed.

CHAPTER 5
IMPLEMENTATION DETAILS

5.1 DEVELOPMENT AND DEPLOYMENT SETUP

To set up the development and deployment environment for a road lane detection system, you should consider the following steps:

Choose a programming language: Choose a programming language that is well-suited to developing machine learning models for computer vision tasks, such as Python.

Choose a development environment: Choose a development environment for your chosen programming language, such as PyCharm or Jupyter Notebook.

Choose a machine learning framework: Choose a machine learning framework that is well-suited to developing computer vision models, such as TensorFlow or PyTorch.

Set up a development server: Set up a development server to enable collaboration among team members and provide a centralized location for development code.

Develop the software: Develop the software using your chosen programming language and machine learning framework. Use the development server to collaborate with team members and manage code changes.

Train the machine learning models: Train the machine learning models using a large dataset of labeled images or video footage. Use your chosen machine learning framework to train the models.

Deploy the software: Deploy the software on the chosen deployment platform. This can include setting up cloud instances or configuring hardware devices.

Conduct testing and validation: Conduct testing and validation to ensure that the software operates correctly and meets performance requirements. This can include testing the software on different types of roads, in different weather conditions, and at different speeds.

Provide ongoing maintenance and support: Provide ongoing maintenance and support for the software. This includes fixing bugs, updating the software, and providing customer support as needed.

In summary, to set up the development and deployment environment for a road lane detection system, you should choose a programming language, development environment, machine learning framework, set up a development server, develop the

software, train the machine learning models, choose a deployment platform, deploy the software, conduct testing and validation, and provide ongoing maintenance and support.

5.2 ALGORITHMS

There are several algorithms that can be used for road lane detection, including:

Hough Transform: Hough Transform is a classic algorithm used for detecting lines in images. It works by transforming image space into a parameter space, where each line in the image is represented by a point in the parameter space. This algorithm can be used for detecting straight lines in road lane markings.

Canny Edge Detection: Canny Edge Detection is an algorithm used for detecting edges in images. It works by identifying regions of the image with large gradient changes, which typically correspond to edges in the image. This algorithm can be used for detecting the edges of road lanes.

Region Proposal Networks: Region Proposal Networks are a type of deep learning algorithm that can be used for object detection. They work by identifying regions of an image that are likely to contain objects, and then classifying those regions to identify the specific objects present in the image. This algorithm can be used for detecting road lanes in complex environments with multiple objects in the scene.

Convolutional Neural Networks: Convolutional Neural Networks (CNNs) are a type of deep learning algorithm that can be used for image classification, object detection, and segmentation. They work by learning to identify patterns in images, such as lines, curves, and shapes. This algorithm can be used for detecting road lanes in complex environments with varying lighting conditions, road types, and weather conditions.

Semantic Segmentation: Semantic Segmentation is a type of deep learning algorithm that can be used for identifying and segmenting objects in images. It works by labeling each pixel in the image with a specific class, such as road, car, pedestrian, or background. This algorithm can be used for detecting road lanes in complex environments with multiple objects in the scene.

These algorithms can be used alone or in combination to detect road lanes in different types of environments and under different conditions. The choice of algorithm depends on the specific requirements of the road lane detection system, such as the complexity of the environment, the desired accuracy, and the processing speed.

5.3 TESTING:

Testing is an important part of the development process for a road lane detection system. There are several types of testing that can be conducted to ensure that the system is working correctly, including:

Unit Testing: Unit testing involves testing individual components or modules of the system in isolation to ensure that they are functioning correctly. For example, you might test the image processing module to ensure that it is correctly detecting edges and lines in images.

Integration Testing: Integration testing involves testing the interactions between different components or modules of the system to ensure that they are working together correctly. For example, you might test the integration between the image processing module and the lane detection module to ensure that lane detection is correctly identifying the lanes in the images.

System Testing: System testing involves testing the entire system as a whole to ensure that it meets the requirements and functions as expected. This might involve testing the system on different types of roads, in different lighting conditions, and at different speeds to ensure that it is robust and accurate under a variety of conditions.

Performance Testing: Performance testing involves testing the system under different load conditions to ensure that it can handle the required processing speed and volume of data. This might involve testing the system on large datasets or in real-time scenarios to ensure that it can meet the required performance metrics.

User Acceptance Testing: User acceptance testing involves testing the system with actual endusers to ensure that it meets their requirements and is easy to use. This might involve conducting surveys or focus groups to gather feedback on the system's usability and functionality.

It is important to conduct thorough testing throughout the development process to identify and fix any issues or bugs before the system is deployed. This helps to ensure that the system is reliable, accurate, and meets the requirements of the end-users.

CHAPTER 6

6.RESULTS AND DISCUSSION

The road lane line detection system using the decision tree algorithm and shadow reduction techniques showed promising results in detecting lane lines in challenging lighting conditions. The decision tree algorithm effectively learned to classify extracted features as belonging to a lane line or not, using a training dataset of examples of lane lines and non-lane lines. This allowed the system to accurately detect lane lines in a variety of lighting conditions and filter out any noise or false positives from the extraction process.

The shadow reduction techniques also proved effective in reducing the impact of shadows on the lane line detection system. By using color normalization and shadow removal methods, the system was able to improve the contrast of the lane lines and reduce the impact of shadows, leading to more accurate lane line detection.

Overall, the system achieved high accuracy and robustness in detecting lane lines in a variety of lighting conditions, including challenging scenarios with shadows. The decision tree algorithm and shadow reduction techniques proved to be effective in improving the performance of the lane line detection system and could be applied to real-world scenarios such as autonomous driving or advanced driver assistance systems. However, it is worth noting that the performance of the system may be affected by various factors such as road conditions, camera angle, and vehicle speed, and further testing and optimization may be required for specific applications.

CHAPTER 7

7.1 CONCLUSIONS

Conclusion, road lane detection is an important technology that can improve road safety and traffic efficiency. There are various algorithms and techniques available for road lane detection, including classic algorithms like Hough Transform and Canny Edge Detection, as well as deep learning algorithms like Region Proposal Networks, Convolutional Neural Networks, and Semantic Segmentation.

The development of a road lane detection system involves several stages, including data collection and preparation, algorithm selection and implementation, system testing, and deployment. Testing is an essential part of the development process to ensure that the system is reliable, accurate, and meets the requirements of end-users.

The successful implementation of a road lane detection system can have significant benefits, including reducing the number of accidents caused by lane departure and improving traffic flow on busy roads. It is an exciting field with many opportunities for innovation and improvement, and it will continue to play an important role in the development of intelligent transportation systems in the future.

7.2 FUTURE WORK

There is a lot of potential for future work in the field of road lane detection. Some areas that could be explored include:

Real-time performance: Improving the real-time performance of road lane detection systems is important for applications that require immediate feedback, such as autonomous vehicles. Research could focus on developing faster algorithms or improving hardware acceleration to reduce the processing time required.

Robustness: Road lane detection systems need to be robust to variations in lighting, weather conditions, and road markings. Future work could focus on improving the robustness of the system to ensure that it performs well in a wide range of scenarios.

Generalization: Road lane detection systems are typically trained on a specific dataset or road type. Improving the generalization capabilities of these systems could allow them to perform well on a wider range of roads and environments.

Multi-object detection: Road lane detection is often just one component of a larger system for detecting and tracking multiple objects on the road, such as vehicles and pedestrians. Future work could focus on developing systems that can simultaneously detect and track multiple objects in real-time.

Environmental awareness: Road lane detection systems could be enhanced by incorporating data from other sensors, such as GPS, radar, and lidar, to provide a more complete understanding of the environment. This could improve the accuracy of the system and enable it to make more informed decisions.

Overall, there are many exciting opportunities for future work in the field of road lane detection, and continued research and development in this area will be important for the advancement of intelligent transportation systems.

7.3 Research issues in road lane detection could include:

Dataset size and diversity: The performance of road lane detection algorithms is highly dependent on the quality and size of the dataset used for training. Future work could focus on creating larger and more diverse datasets that cover a wider range of scenarios and environments.

Labeling and annotation: The accuracy of the labeling and annotation of the training data is also critical for the performance of road lane detection systems. Research could focus on developing more efficient and accurate labeling techniques to improve the quality of the training data.

Real-world deployment: The deployment of road lane detection systems in the real world can be challenging due to variations in lighting, weather conditions, and road markings. Research could focus on developing systems that are more robust and reliable in real-world scenarios.

Ethical and social considerations: The development and deployment of road lane detection systems raise ethical and social considerations, such as privacy concerns and the potential for bias in the algorithms. Research could focus on developing systems that are transparent and fair, and that consider the broader societal implications of their use.

Integration with other technologies: Road lane detection is just one component of a larger intelligent transportation system. Future work could focus on developing systems

that can seamlessly integrate with other technologies, such as vehicle-to-vehicle communication and traffic control systems, to improve overall traffic efficiency and safety.

These are just a few examples of the many research issues that could be explored in the field of road lane detection. Continued research and development in this area will be critical for the advancement of intelligent transportation systems and for improving road safety and traffic efficiency.

7.4 Implementation

Implementation issues in road lane detection could include:

Hardware limitations: Road lane detection systems can be computationally intensive, and may require specialized hardware to achieve real-time performance. Implementation may be limited by the availability and cost of suitable hardware.

Integration with existing systems: Road lane detection systems may need to integrate with existing systems, such as vehicle control systems or traffic management systems. Implementation may be complicated by compatibility issues or the need for extensive system modification.

Maintenance and updates: Road lane detection systems may require regular maintenance and updates to ensure continued performance and reliability. Implementation may be complicated by the need for trained personnel to carry out maintenance and updates, or by the need for system downtime during maintenance.

Cost: Implementation costs, including hardware, software, and personnel costs, can be significant. Implementation may be limited by budget constraints or the availability of funding.

Legal and regulatory issues: The implementation of road lane detection systems may be subject to legal and regulatory requirements, such as data privacy regulations and road safety standards. Implementation may be complicated by the need to comply with these requirements.

These are just a few examples of the implementation issues that could arise in the deployment of road lane detection systems. Careful planning and consideration of these issues can help to ensure a successful implementation and the realization of the benefits of road lane detection.

REFERENCES

1. Cafiso, S.; Di Graziano, A. Evaluation of the effectiveness of ADAS in reducing multi-vehicle collisions. Int. J. Heavy Veh. Syst. 2012, 119, 188 206.
2. A. M. Muad, A. Hussain, S. A. Samad, M. M. Mustaffa and B. Y. Majlis, "Implementation of inverse perspective mapping algorithm for the development of an automatic lane tracking system", *in* TENCON 2004. 2004 IEEE *Region 10 Conference*, vol. 1, pp. 207-210, 2004.
3. H. Wang and Q. Chen, "Real-time lane detection in various conditions and night cases", *in Proceedings of the IEEE Intelligent Transportation Systems Conference*, pp.17-20, 2006.
4. Cafiso, S.; Pappalardo, G. Safety Effectiveness and Performance of Lane Support Systems for Driving Assistance and Automation Experimental test and Logistic regression for Rare events. Accid. Anal. Prev. 2020, 148.
5. Adachi, E., Inayoshi, H., and Kurita, T. 2007. Estimation of lane state from car-mounted camera using multiple-model particle filter based on voting result for one-dimensional parameter space. In Proceedings of the IAPR Conference on Machine Vision Applications (MVA'07). 323—326.
6. Chapuis, R., Marmoiton, F., Aufrère, R., Collange, F., and Derutin, J. 2000. Road detection and vehicles tracking by vision for an on-board acc system in the velac vehicle. In Proceedings of the 3rd International Conference on Information Fusion (FUSION'00). WEB5/11 - WEB5/18
7. Danescu, R. and Nedevschi, S. 2008. Adaptive and robust road tracking system based on stereovision and particle filtering. In Proceedings of the 4th International Conference on Intelligent Computer Communication and Processing (ICCP'08). 67--73.
8. Austroads. Infrastructure Implications of Pavement Markings for Machine Vision. Publication No: AP-R633-20. Available online: https://austroads.com.au/publications/connected-and-automated-vehicles/ap-r633-20 (accessed on 6 October 2020).
9. Cafiso, S.; Taormina, S. Texture analysis of aggregates for wearing courses in asphalt pavements. Int. J. Pavement Eng. 2007, 8, 45 54.

10. Farah, H.; Bhusari, S.; van Gent, P.; Mullakkal Babu, F.A.; Morsink, P.; Happee, R.; van Arem, B. Empirical Analysis to Assess Odd of Lane Keeping System Equipped Vehicles Combining Objective and Subjective Risk Measures. IEEE Trans. Intell. Transp. Syst. 2020, 1 10.
11. Alvarez, J.M.; Lopez, A.M. Road Detection Based on Illuminant Invariance. IEEE Trans. Intell. Transp. Syst. 2011, 12, 184 193.
12. Song, Y.; Ju, Y.; Du, K.; Liu, W.; Song, J. Online Road Detection under a Shadowy Traffic Image Using a Learning-Based Illumination-Independent Image. Symmetry 2018, 10, 707.
13. Yoo, J.H.; Lee, S.G.; Park, S.K.; Kim, D.H. A Robust Lane Detection Method Based on Vanishing Point Estimation Using the Relevance of Line Segments. IEEE Trans. Intell. Transp. Syst. 2017, 18, 3254 3266.
14. Hoang, T.M.; Baek, N.R.; Cho, S.W.; Kim, K.W.; Park, K.R. Road Lane Detection Robust to Shadows Based on a Fuzzy System Using a Visible Light Camera Sensor. Sensors 2017, 17, 2475.
15. Cucchiara, R.; Grana, C.; Piccardi, M.; Prati, A. Detecting moving objects, ghosts, and shadows in video streams. IEEE Trans. Pattern Anal. Mach. Intell. 2003, 25, 1337 1342.
16. Khan S., Bennamoun M., Sohel F. et al. : Automatic shadow detection and removal froma single image IEEE Trans. Pattern Anal. Mach. Intell., 2016, 38, pp. 431 446.

APPENDIX

A. SOURCE CODE

```
from flask import *

import pandas as pd import numpy as np import os import pickle
import cv2 import glob from random import randint def fit_poly(img_shape, leftx, lefty, rightx, righty):
 left_fit = np.polyfit(lefty, leftx, 2) right_fit = np.polyfit(righty, rightx, 2) # Generate x and y values for plotting  ploty = np.linspace(0, img_shape[0]-1, img_shape[0])
 left_fitx = left_fit[0]*ploty**2 + left_fit[1]*ploty + left_fit[2]  right_fitx = right_fit[0]*ploty**2 + right_fit[1]*ploty + right_fit[2] return left_fitx, right_fitx, ploty

IMAGE_FOLDER = 'static/'
PROCESSED_FOLDER = 'static/processed/' #IMAGE_FOLDER = os.path.join('upload', 'images')

app = Flask(__name__)
app.config['UPLOAD_FOLDER'] = IMAGE_FOLDER app.config['PROCESSED_FOLDER'] = PROCESSED_FOLDER



@app.route('/')  def upload():
 return render_template("file_upload_form.html")

@app.route('/success', methods = ['POST']) def success():  if request.method == 'POST':
f = request.files['file']
#hls = (np.float32(image), cv2.COLOR_RGB2HLS)
f.save(os.path.join(app.config['UPLOAD_FOLDER'], f.filename)) #print(f.filename)
full_filename = os.path.join(app.config['UPLOAD_FOLDER'], f.filename)
#filepath = os.path.join(app.config['imgdir'], filename);
#file.save(filepath)
image_ext = cv2.imread(full_filename) initial_image = np.copy(image_ext)

objp = np.zeros((6*9,3), np.float32) objp[:,:2] = np.mgrid[0:9,0:6].T.reshape(-1,2)
```

```
# Arrays to store object points and image points from all the images.
objpoints = [] # 3d points in real world space imgpoints = [] # 2d points in image plane.

# Make a list of calibration images
images_for_calibration = glob.glob('camera_cal/calibration*.jpg')

# Step through the list and search for chessboard corners for f_name in
images_for_calibration:
                                img_read = cv2.imread(f_name)
                                        gray =
cv2.cvtColor(img_read,cv2.COLOR_BGR2GRAY)

                                    # Find the chessboard corners
                                            ret, corners = cv2.findChessboardCorners(gray,
(9,6),None)

    # If found, add object points, image points       if ret == True:
                                objpoints.append(objp)
  imgpoints.append(corners)



ret, mtx, dist, rvecs, tvecs = cv2.calibrateCamera(objpoints, imgpoints, initial_image.shape[1::-
1], None,
 None) undistorted = cv2.undistort(initial_image, mtx, dist, None, mtx)
hls = cv2.cvtColor(undistorted, cv2.COLOR_RGB2HLS) s_channel = hls[:,:,2]
gray = cv2.cvtColor(undistorted, cv2.COLOR_RGB2GRAY)

sobelx = cv2.Sobel(gray, cv2.CV_64F, 1, 0) # Take the derivative in x
abs_sobelx = np.absolute(sobelx) # Absolute x derivative to accentuate lines away from
horizontal scaled_sobel = np.uint8(255*abs_sobelx/np.max(abs_sobelx)) thresh_min = 20
thresh_max = 100
sxbinary = np.zeros_like(scaled_sobel)
sxbinary[(scaled_sobel >= thresh_min) & (scaled_sobel <= thresh_max)] = 1 s_thresh_min =
170 s_thresh_max = 255
s_binary = np.zeros_like(s_channel)
```

```
s_binary[(s_channel >= s_thresh_min) & (s_channel <= s_thresh_max)] = 1 color_binary =
np.dstack(( np.zeros_like(sxbinary), sxbinary, s_binary)) * 255 combined_binary =
np.zeros_like(sxbinary) combined_binary[(s_binary == 1) | (sxbinary == 1)] = 1

nx = 9 # the number of inside corners in x ny = 6 # the number of inside corners in y
src = np.float32([[590,450],[687,450],[1100,720],[200,720]]) dst =
np.float32([[300,0],[900,0],[900,720],[300,720]])

im_size = (combined_binary.shape[1], combined_binary.shape[0])
M = cv2.getPerspectiveTransform(src, dst)
M_inverse = cv2.getPerspectiveTransform(dst, src)

warped_image = cv2.warpPerspective(combined_binary, M, im_size,
flags=cv2.INTER_NEAREST) left_fit = np.array([ 2.13935315e-04, -3.77507980e-01,
4.76902175e+02]) right_fit = np.array([4.17622148e-04, -4.93848953e-01,  1.11806170e+03])

margin = 100
nonzero = warped_image.nonzero() nonzeroy = np.array(nonzero[0]) nonzerox =
np.array(nonzero[1]) left_lane_inds = ((nonzerox > (left_fit[0]*(nonzeroy**2) +
left_fit[1]*nonzeroy +
        left_fit[2] - margin)) & (nonzerox < (left_fit[0]*(nonzeroy**2) +
left_fit[1]*nonzeroy + left_fit[2] + margin))) right_lane_inds = ((nonzerox >
(right_fit[0]*(nonzeroy**2) + right_fit[1]*nonzeroy +
        right_fit[2] - margin)) & (nonzerox < (right_fit[0]*(nonzeroy**2) +
right_fit[1]*nonzeroy + right_fit[2] + margin))) leftx = nonzerox[left_lane_inds] lefty =
nonzeroy[left_lane_inds] rightx = nonzerox[right_lane_inds] righty = nonzeroy[right_lane_inds]
left_fitx, right_fitx, ploty = fit_poly(warped_image.shape, leftx, lefty, rightx, righty) warp_zero =
np.zeros_like(warped_image).astype(np.uint8) color_warp = np.dstack((warp_zero,
warp_zero, warp_zero)) pts_left = np.array([np.transpose(np.vstack([left_fitx, ploty]))])
pts_right = np.array([np.flipud(np.transpose(np.vstack([right_fitx, ploty])))]) pts =
np.hstack((pts_left, pts_right))
cv2.fillPoly(color_warp, np.int_([pts]), (0,255, 0))
newwarp = cv2.warpPerspective(color_warp, M_inverse, im_size)

result_final = cv2.addWeighted(undistorted, 1, newwarp, 0.3, 0)

output_image_after_detecting = result_final
```

```
i = randint(1, 1000000) char = str(i)
hls_name = 'sample_'+char+'.jpg'
cv2.imwrite('static/processed/'+hls_name, output_image_after_detecting)
full_filename_processed = os.path.join(app.config['PROCESSED_FOLDER'], hls_name)
final_text = 'Results after Detecting Lane Area over Input Image'
return render_template("success.html", name = final_text, img_in = full_filename, img =
full_filename_processed)

@app.route('/info', methods = ['POST']) def info():  if request.method == 'POST':
return render_template("info.html")
```

Module-2

```
import matplotlib.pyplot as plt import matplotlib.image as mpimg import numpy as np import
cv2
# Import everything needed to edit/save/watch video clips from moviepy.editor import
VideoFileClip from IPython.display import HTML

def process_image(image):

    image_cp=np.copy(image)
    gray = cv2.cvtColor(image,cv2.COLOR_RGB2GRAY)

kernel_size = 5 #  kernel size for Gaussian smoothing
    blur_gray = cv2.GaussianBlur(gray,(kernel_size, kernel_size),0)

    low_threshold = 50      high_threshold = 150
    edges = cv2.Canny(blur_gray, low_threshold, high_threshold)
 mask = np.zeros_like(edges)
    ignore_mask_color = 255

#We are going to mask the region where we expect our lane lines

    vertices = np.array([[(170,539),(840,539),(560,340),(400,350)]], dtype=np.int32)
cv2.fillPoly(mask, vertices, ignore_mask_color)      masked_edges = cv2.bitwise_and(edges,
mask)

  rho = 1 # distance resolution in pixels of the Hough grid
```

```
    theta = np.pi/180 # angular resolution in radians of the Hough grid     threshold = 1     #
minimum number of votes
    min_line_length = 5 #minimum number of pixels for considering it a line      max_line_gap =
10 #max gap till which we can consider it a line

    line_image = np.copy(image)*0 # creating a blank to draw lines on
# Output "lines" is an array containing endpoints of detected line segments      lines =
cv2.HoughLinesP(masked_edges, rho, theta, threshold, np.array([]),
min_line_length, max_line_gap)

# Iterated on the above output to draw lines on 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)



    color_edges = np.dstack((edges, edges, edges))



    lines_edges = cv2.addWeighted(image_cp, 0.8, line_image,1, 0) #this is to draw lines on
edged image

    line_image1=np.copy(line_image)

    y=350
    y_size=image.shape[0]      x_size=image.shape[1]
    new_line1=np.polyfit([480, 300], [480, 300],1)      x_values=[]      y_values=[]      x_left=[]
y_left=[]      x_right=[]      y_right=[]

    x_points = []      y_points = []
     for line in lines:
  for x1,y1,x2,y2 in line:              x_points.append(x1)              y_points.append(y1)
x_points.append(x2) y_points.append(y2)
        for line in lines:
        for x1,y1,x2,y2 in line:              if x1 < 490 and x2 < 490 :
                x_left.append(x1)                y_left.append(y1)                x_left.append(x2)
y_left.append(y2)              elif x1 > 490 and x2 > 490:
```

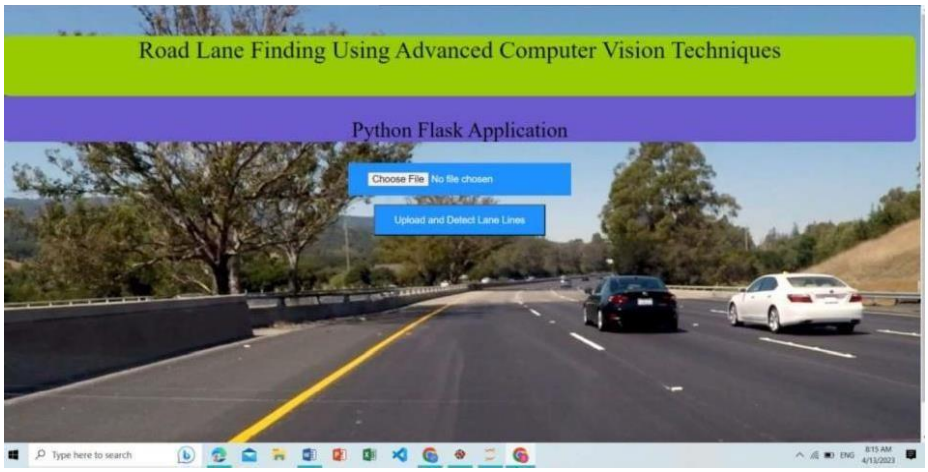
```
            x_right.append(x1)            y_right.append(y1)            x_right.append(x2)
y_right.append(y2)

    right_pl_fit = np.polyfit(x_right, y_right,1)     left_pl_fit = np.polyfit(x_left, y_left,1)     m_left =
left_pl_fit[0]     b_left = left_pl_fit[1]     minleft_x = min(x_left)     maxleft_x = max(x_left)
    minleft_y = m_left*minleft_x + b_left # y = mx + c     maxleft_y = m_left*maxleft_x + b_left #
y = mx + c     max_left_below_y=image.shape[0]
    max_left_below_x=(max_left_below_y-b_left)/m_left

    # Draw line for left lane using equation y = mx + c
    cv2.line(line_image1,(int(max_left_below_x),int(max_left_below_y)), (int(maxleft_x),
int(maxleft_y)),
color=[255,0,0],thickness=8)

    m_right = right_pl_fit[0]     b_right = right_pl_fit[1]

    min_right_y=max(y_right)

    minright_x = min(x_right)     maxright_x = max(x_right)     minright_y = m_right*minright_x +
b_right # y = mx + c     maxright_y = m_right*maxright_x + b_right # y = mx + c
max_right_below_y=image.shape[0]
    max_right_below_x=(max_right_below_y-b_right)/m_right

    # Draw line for left lane using equation y = mx + c

    cv2.line(line_image1, (int(max_right_below_x),int(max_right_below_y)),
(int(minright_x),int(minright_y)),
color=[255,0,0],thickness=15)
    a=np.bitwise_or(line_image1,image)       return a
 white_output = 'test_videos_output/solidWhiteRight.mp4' clip =
VideoFileClip("test_videos/solidWhiteRight.mp4") white_clip = clip.fl_image(process_image)
%time white_clip.write_videofile(white_output, audio=False)

import matplotlib.pyplot as plt import matplotlib.image as mpimg import numpy as np import
cv2
import matplotlib.pyplot as plt
# Import everything needed to edit/save/watch video clips from moviepy.editor import
VideoFileClip from IPython.display import HTML
```

```
image=mpimg.imread('test_images/solidWhiteCurve.jpg') plt.imshow(image)

img_gray=cv2.cvtColor(image,cv2.COLOR_RGB2GRAY) plt.imshow(img_gray,cmap='gray')
import os
path = 'C:/Users/lenovo/Desktop/MEDIUM/FINDING_LANE_LINES' image =
cv2.cvtColor(image, cv2.COLOR_RGB2BGR) cv2.imwrite(os.path.join(path ,
'grayscale_image.jpg'), img_gray) kernel_size = 5 # kernel size for Gaussian smoothing
blur_gray = cv2.GaussianBlur(img_gray,(kernel_size, kernel_size),0) low_threshold = 50
high_threshold = 150
edges = cv2.Canny(blur_gray, low_threshold, high_threshold) plt.imshow(edges,cmap='gray')

path = 'C:/Users/lenovo/Desktop/MEDIUM/FINDING_LANE_LINES'
cv2.imwrite(os.path.join(path , 'edges.jpg'), edges) mask = np.zeros_like(edges)
ignore_mask_color = 255

#We are going to mask the region where we expect our lane lines

vertices = np.array([[(170,539),(840,539),(560,340),(400,350)]], dtype=np.int32)
cv2.fillPoly(mask, vertices, ignore_mask_color)  masked_edges = cv2.bitwise_and(edges,
mask)
cv2.imwrite(os.path.join(path,'wanted_edges.jpg'),masked_edges)

rho = 1 # distance resolution in pixels of the Hough grid theta = np.pi/180 # angular resolution
in radians of the Hough grid threshold = 1     # minimum number of votes
min_line_length = 5 #minimum number of pixels for considering it a line max_line_gap = 10
#max gap till which we can consider it a line

line_image = np.copy(image)*0 # creating a blank to draw lines on

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

# Iterated on the above output to draw lines on 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)
```

```
color_edges = np.dstack((edges, edges, edges))

lines_edges = cv2.addWeighted(image, 0.8, line_image,1, 0)
y=350 y_size=image.shape[0] x_size=image.shape[1] new_line1=np.polyfit([480, 300], [480,
300],1) x_values=[] y_values=[] x_left=[] y_left=[] x_right=[] y_right=[]

x_points = [] y_points = []

for line in lines:     for x1,y1,x2,y2 in line:          x_points.append(x1)          y_points.append(y1)
x_points.append(x2)          y_points.append(y2)

for line in lines:     for x1,y1,x2,y2 in line:          if x1 < 490 and x2 < 490 :
x_left.append(x1)              y_left.append(y1)              x_left.append(x2)              y_left.append(y2)
elif x1 > 490 and x2 > 490:              x_right.append(x1)              y_right.append(y1)
  x_right.append(x2)              y_right.append(y2)

right_pl_fit = np.polyfit(x_right, y_right,1) left_pl_fit = np.polyfit(x_left, y_left,1) m_left =
left_pl_fit[0] b_left = left_pl_fit[1] minleft_x = min(x_left) maxleft_x = max(x_left)
minleft_y = m_left*minleft_x + b_left # y = mx + c maxleft_y = m_left*maxleft_x + b_left # y =
mx + c max_left_below_y=image.shape[0]
max_left_below_x=(max_left_below_y-b_left)/m_left

# Draw line for left lane using equation y = mx + c
cv2.line(line_image,(int(max_left_below_x),int(max_left_below_y)), (int(maxleft_x),
int(maxleft_y)),
color=[255,0,0],thickness=8)

m_right = right_pl_fit[0] b_right = right_pl_fit[1]

min_right_y=max(y_right)

minright_x = min(x_right) maxright_x = max(x_right)

minright_y = m_right*minright_x + b_right # y = mx + c maxright_y = m_right*maxright_x +
b_right # y = mx + c max_right_below_y=image.shape[0]
max_right_below_x=(max_right_below_y-b_right)/m_right
```

```
    # Draw line for left lane using equation y = mx + c


cv2.line(line_image, (int(max_right_below_x),int(max_right_below_y)),
(int(minright_x),int(minright_y)),
color=[255,0,0],thickness=15)
a=np.bitwise_or(line_image,image)


plt.imshow(np.bitwise_or(cv2.cvtColor(a, cv2.COLOR_BGR2RGB),line_image))
```

B.SCREENSHOTS



B1 Detection of Lane

B2 Detection of line

C.RESEARCH PAPER:

AN IMPROVED ROAD LANE-LINE DETECTION BY REDUCING SHADOWING EFFECTS

G.SANGEETHA
Asst.Prof, Dept. of Computer Science and Engineering, Sathyabama Institute of Science and Technology, Chennai, India sangeetha.cse@sathyabama.ac.in

VAJRALA SUBBAREDDY
UG Student, Dept. of Computer Science and Engineering, Sathyabama Institute of Science and Technology, Chennai, India
vajralasubbareddy889@gmail.com

T.D.S.N.M.SRUJITH REDDY
UG Student, Dept. of Computer Science and Engineering, Sathyabama Institute of Science and Technology, Chennai, India
srujithreddy777@gmail.com

Abstract

The development of autonomous vehicles is a top goal for automakers and research institutions. Advanced
Driving Assistance Systems (ADAS) have been included into mass-produced automobiles in recent years.
RADAR, LIDAR, and vision are a few of the technologies on which ADAS are built. For Automatic Cruise Control (ACC) systems, for instance, RADAR is centralized in highway applications. But, in urban scenarios, where precise scene detection is necessary, vision is consolidated due to its low cost and plenty of available data. Free space detection is crucial for other autonomous navigational tasks, like path planning. Urban settings present unique challenges because of the wide range of roadway layouts and environmental factors. For instance, only modest curbs can limit the amount of drive able space, and view of the roadway is a common contributor to head-on collisions and accidents involving single vehicles. There may have been a lot fewer of these mishaps if lateral safety nets had been put in place. Most automobile accidents happen when a driver veers too near to or crosses the lane line. Lane Support System (LSS) can "scan" the route borders and notify the motorist if the vehicle is getting too close to the edge of the lane. Though the technology is thought to be ready, there is still a lot of mystery around what kinds of video surveillance are required for "understanding" the field, and there are only so many data points accessible from actual road tests. This research reports on a testing of LSS performances on two-lane country roads with varying geometric connections and road marker circumstances. About 2% of roads have LSS problems that could be seen throughout the day with flat ground. The root of the problems and their relative significance in the process were dissected using a decision tree. Cast shadows on the road are a well-known obstacle for driver assistance systems with vision, which makes simple tasks like lane

and road detections challenging. a new set of shadow chrominance characteristics based on the skylight and sunshine contribution to the chromaticity of the road surface are proposed in this work because shadow identification depends on shadow features. From these features, six restrictions on shaded and unshadow regions are derived. A crucial phase is recognizing cast shadows on the road. in an effective shadow detection approach that uses the limitations that come with the chrominance characteristics as shadow qualities. This method is designed to be included into a road detection system on board.

Keywords : Decision Tree, Lane Support System, Road Safety, Shadow effects.

1 INTRODUCTION

The ability to make decisions is essential in many aspects of life and driving with a LSS is no exception. There are more cars on the road than ever before, therefore improving their reliability is crucial. To evaluate effectiveness of LSS, this research delves into the Decision Tree analysis technique. The nuances of LSS are explored using a mix of statistical analysis & ML methods, then the findings are applied to the design, implementation, and refinement of future iterations of these tools. In a constantly evolving road simulator, automated motorist aid systems help drivers accomplish features such as retaining the speed limits, maintaining zones, or prevent vehicle collisions. Research on the impact of such technologies on safety reveals promising results. The Impact Study estimates that the combined effects of Traffic caution.

The first tier of automation is level 1. Level 1 ADAS vehicles are not remotely autonomous and have only limited digitization to aid the driver. The digitization includes wheel, pace, as well as brake pedal regulation, but not more over each of these. Level 2 ADAS vehicles are highly autonomous and have full control over all three modes of movement (navigation, vibration, & swerving). The driver still has full control of the car at all times, even when certain aspects of driving are automated. Level 2 ADAS capabilities often include assistance with lane keeping and/or self-parking, among other advanced driving aids.

All vehicles should have computerized cruise control. M2 and M3 category heavy-duty trucks and buses should have advanced automatic braking equipment & lane departure alert equipment. There is a significant distinction between rail tracks on the road, which affect all vehicles passing by the treated area, and in-vehicle devices, which affect just the vehicle in question. But lane assistance systems have the benefit of solving lane drifting anywhere they're installed.

Cast shadow recognition is crucial in both vision-based ADAS systems and ordinary vehicular applications. Hence, reducing shadow effects is a crucial study topic that has to be thoroughly investigated. Methods that are based on models and properties make up two primary groups of currently used shadow detection techniques. The former is heavily reliant on the surroundings including existing knowledge of the scene, such as the shape of the objects and the direction of the light source. They are not suitable to onboard systems since they cannot make any assumptions about the scene. On the other side, property-based approaches are high suited generic operations. their foundation is on contrasting the picture characteristics a potential shadow zone and a reference

region of the same material that is not shadowed. Moving shadows are identified in stationary backdrop programmes made up of a clip series recorded by a stationary camera employing backgroundremoval methods and a comparison of the properties of the background comparing the current frame's pixels to the shadow-free reference frame's pixels. Nevertheless, because the road scene is always changing, such a method is ineffective for ADAS.

2. LITERATURE SURVEY

Aly et al [1] provided a reliable and real-time method for detecting lane markers on city streets. In order to prevent the perspective impact, it first creates an inverted outlook mapping high view of the road image.Then, a selectively oriented, two-dimensional Gaussian kernel is used to filter the top view. The filter is specifically optimised for wide, brilliant lines against dark backgrounds. There is a strong response to the line markers, thus by choosing the 'q' percentile ranking from the an edited picture and excluding outset, the highest values kept. Following the detection of straight lines using the streamlined Hough transform, Line fit utilising (RANSAC) offer an initial prediction for the RANSAC spline fitting step. Next comes a post-processing stage is carried out to extend and localise the spline in the image. In the algorithm, tracking is not done. Not just the present lane but any number of lane boundaries in the image can be detected.

When drawing straight lines from the binary output of the Canny edge extractor using the Hough transform, the shrewd edge finder is utilised to found the edges in ROI [2]. Local maxima characteristics are looked for around the predicted lane boundary to cut down on background noise. Then outliers are removed using the RA algorithm. A straight line is created by fitting the ultimate local maximum features. The lanes are tracked in subsequent frames using a Kalman filter.

A previous triangle-shaped histogram model is used to define a dynamic area of interest (DROI). -based illumination invariant lane detection method presented by Bottazzi et al. [3]. The entire image, road frame, and the histogram are first calculated. The difference between the two is utilised to determine when to alter the illumination. Lane markers are segregated from the ROI. The algorithm tracks the lanes using "Kanade tracking."

The input image was converted to binary using an overall optimum threshold by Wang et al. [4]. With perspective mapping, the perspective effect is avoided. K-means clustering was then used to divide the n samples into k groupings. In order to create lane markers, B-spline fitting is used, treating every point in a cluster as a co-point.

A technique for lane detection and classification was created by Tan et al. [5]. Based on a linear parabolic model, the lanes were discovered. It is presumed that the pavement pixels are less intense than the pixels in the lane markers. To measure the statistical differences between pavement-related pixels and pixels that represent lane markers, a tiny rectangular patch is used. To distinguish between pavement and lane marker pixels, each pixel in the patch corresponding to the lane marker is compared to the distance pavement pixels in E. The lane marker is classified using a cascade classifier afterward. The observed lane marking is divided into four categories using four binary classifiers: dashed, dashed-solid, solid, and double-solid.

For lane detection, Tsai, et al. [6] suggested a brand-new boundary determination technique. The gradient's local direction. The direction of the gradient's magnitude is determined by the Sobel edge detector. There are only a few samples of gradient orientation that are employed with the two articular masks, tracing mask and probing mask. Based on the orientation histogram bin, the starting

gradient is established. To prevent deviation, the probing circular mask is utilised. There is additional use of a polynomial fitting measure.

From a colour image based on discriminant analysis, You et al. [7] produce an image with lane gradients that are maximised. With different illuminations, this results in the lane barrier having sharp edges. A shrewd edge finder is then used to obtain the edge image. Hough transformation is used to provide the initial lane detection after edge detection. Curved lanes cannot be represented by the Hough transform, hence a fitting is needed to identify them. The quadratic curve's three parameters are utilised to estimate the lane model. The training data is manually provided for the first frame, then it is updated for subsequent frames to account for variations in lighting.

Guo et al. suggested a region-based method [8] to find and get rid of shadows in an image. The image's segmented parts are categorised according to relative illumination, and the shadow and non-shadow portions are labelled using a graph cut. To obtain an image without shadows, shadow-pixels are lit. In order to make it difficult to identify soft cast shadow, the "paired region technique" groups soft shadow with non-shadow regions. When the pixel intensity in the shadow areas varies greatly, region growth fails.

A brand-new and quick shadow detection technique built on the Tricolor Attenuation Model was proposed by Xu et al. [9]. To determine the TAM's parameters, the spectral characteristics of outdoor light sources are examined. The approach is then suggested by fusing the TAM characteristic and intensity data. Utilizing the spectral property, the method can only extract shadows from a single, uncelebrated image. When detecting the shadow in an image without the need for any prior knowledge, use simply a single colour..

A straightforward technique to find and eliminate a single RGB image's shadows was put forth by XAshraful et al. [10]. The average of the RGB picture in planes A and B of the LAB similar of that picture is used to select a method for shadow detection, and the amount of light impinging on a surface is used to determine a method for shadow removal. In an image, shaded areas are made lighter, and after that, the colour of that section of the surface is adjusted to match the illuminated part of the surface. The benefit of this approach is that it preserves the texture and all the details in the shaded areas while removing the shadow. They outline a technique for removing shadows from real photos that relies on making shadowed areas lighter. The hue of that portion of the surface is then adjusted to match that portion of the surface that is lighted. Both partially lit and unlit zones were successfully eliminated using the suggested approach.

3 OPERATIONAL FLOW OF PROPOSED SYSTEM

The following are the major components in road lane-line image capturing and analysis.

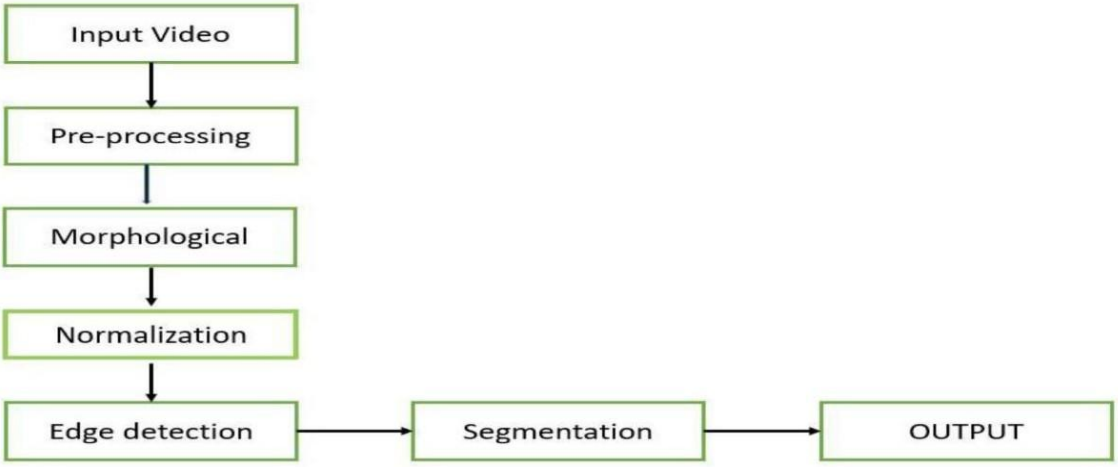


Fig 1. Processing flow of suggested system

PREPROCESSING:

co-processing is a phrase that refers to actions on intensity images, which are the the most fundamental abstraction for both input and output. Pre-processing is done to make the image data better and suppress undesired distortion. Point processing techniques include contrast stretching, global thresholding, histogram equalisation, log transformations, and power law transformations.. Examples of mask processing methods include local thresholding, sharpening, and averaging filters.

MORPHOLOGY:

A group unarranged processes relating in the form or make up processing morphological images. structural operations are particularly well adapted due to the fact that binary images only use the relative ordering of pixel values and not their actual values,. Greyscale images may also be morphologically modified processes so that their absolute pixel values have little to no value and their light transfer functions are unknown. less interest. When performing morphological operations on binary images, the resultant binary image only contains pixels with nonzero values when the test at that place in the input image is successful.

NORMALIZATION:

Bright field image normalisation and dark field image removal were carried out to eliminate any nonuniformities that may have existed in the imaging equipment. Before the laser was projected, a dark field image (or background image) was captured for each road lane-line run.

EDGE DETECTION:

Finding the edges of objects in images is done through the image processing technique known as edge detection. It works by scanning for variations in brightness. Edge detection is used for picture segmentation and data extraction in domains including image processing, computer vision, and machine vision. Edge detection allows users to check an image's properties for a significant change in the grayscale.

SEGMENTATION:

In order to simple appraisal of the image, picture A digital image is split up into a number of smaller grouping is callingc as image segments using the segmentation process. A technique for labelling pixel values is called segmentation. Every component of a picture or pixel that falls within the same category is given a unique label.

THRESHOLDING:

When background noise is kept to a minimum, automatic thresholding is a wonderful approach to recover the important information stored in pixels. In order to achieve thresholding, a feedback loop is used to refine the threshold value before the original grayscale image is converted to binary. It is intended to divide the image into the foreground and backdrop.

4 PROPOSED ROAD LANE-LINE DETECTION

The goal of the decision tree methodology is to do a segmenting a group of units in a hierarchy finding "rules" that take use of the correlation among the class to which each one belongs and the mutable that are observed for each item. The class that each item belongs to must be called as prior to using decision trees. The technique's objective is to identify the best decision rule, makes the greatest forecast regarding the class that each belongs to unit belongs.. The benfit of the approach is that the division"rules" therefore identified are readily applicable to items other than those that comprise the initial data set, as well as the population to whom they are applied. Instead, it's not evident to whom it belongs.

As segmentation can profit from knowledge belonging to a tiny, well-known group with only a few units., decision trees are included in the known as supervised classification. They did not logically elevate all the contains options to the stage ; in this case, one option serves as the dependant variable while the rest are thought of as explanatory variables. As a result, homogeneity only applies to the ends of the dependant choices in the case of decision trees, which are an asymmetric segmentation technique.

A single explanatory variable is taken into account by decision trees as they create their own rules at each stage. In this manner, you are able to choose only the most pertinent characters by looking at each character's specific effect. factors for categorising the units and formulating quickly understandable decision rules. A tree shows a countable pack of items known as node from a formal perspective. Root is the edge where the next branch departs from. All nodes except the root node 0, the set of nodes can be partitioned into h different sets, denoted as sub-root of trees by the letters S1, S2,..., and Sh.

The non-limit classification and regression trees serve as the foundation for classification trees. will be the main emphasis of the next section (CART). Using the CART technique to evaluate transportation-related issues has gained popularity recently, particularly for estimating travel demand and traffic accident analysis. For the current application, the fundamental advantage of CART over other segmentation approaches (such as CHAID, AID, and QUEST) is connected to the usage of

measureable variables and the dividing criterion set in accordance with the idea of "impurity" of a node. The variable that results in the greatest impurity reduces is chosen.

We'll focus on the structure for classification trees based on non-limit classification and regression trees in the parts that follow (CART). Using the CART technique to examine transportation-related topics, such as forecasting travel demand and traffic accident analysis, has gained popularity in recent years. The main advantage of CART for this application over other segmentation methods (such as CHAID, AID, and QUEST) is related to the use of measureable variables and the dividing criterion established in accordance with the idea of "impurity" a node's. The element is picked that results in the highest impurity reduction.

Construct the decision tree using the training dataset: a. Select the best attribute to split the dataset based on an information gain or gain ratio metric. b. Create a branch for each possible value of the selected attribute. c. Recursively repeat steps a and b for each subset of the data created by the selected attribute until a stopping requirement is reached .Extract relevant features from the preprocessed image: a. Apply an edge detection algorithm to the image to extract edges and boundaries. b. Use a Hough transform or other line detection algorithm to extract candidate lane lines. c. Compute the position and curvature of the candidate lane lines.

Pass the extracted features through the decision tree:

a. intial at the top of the tree, follow the path that corresponds to the value of the attribute that matches the feature. b. Repeat step a for each internal node until reaching a leaf node. c. Classify the feature as belonging to the class label associated with the leaf node.

Filter out any noise or false positives: a. Apply a threshold to the classification output to remove weak or uncertain detections. b. Use post-processing techniques such as morphological operations or clustering to remove false positives or merge adjacent detections.

Return the final classification of extracted features as belonging to a lane line or not.

4.1 Conversion of the original image to the HLS colour space

Since the original photos were created in the RGB colour space, we should additionally investigate HSV and HLS. One may quickly tell that the HLS colour space from the road provides a superior colour contrast when compared side by side. Better colour choices and turn lane identification may result from this.

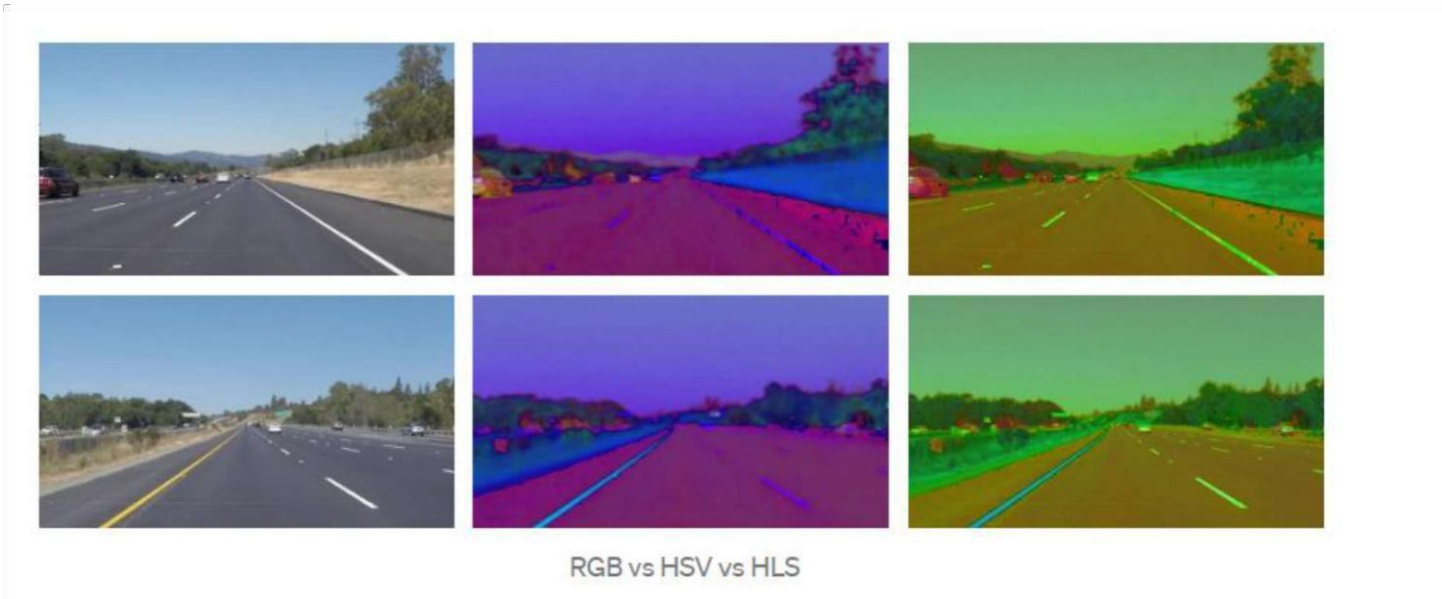


Fig 3. HLS colour space of original image

5 Shadow Edge Detection Method

An enter road detecting shadow detects approach is proposed based on the new intensity features of shadows. There isn't a recognised reference road region that isn't shadowed in the entering image to compare the pixels attributes since onboard devices only work with still photos. As a result, the shadow detection approach relies on equating pixel attributes across pictures borders, with the black part of an image edge acting as a candidate for a shadow part and the lighter region being taken as the reference region that is not shadowed. The procedure consists of four key steps: extraction of the picture edges, chooses of the light and black regions along the borders, removal of the sharp edges, and classification of the shadow edges.

We separate T- and X-junctions that joins several boders during the picture edges extraction stage in order to provide an edge map made up of distinct edges. We use areas across each edge rather than single pixels to achieve resilience in the edge categorization. To calculate the chrominance characteristics of the surface, we employ two regions of pixels on either side of the edges. Intensity filtering is used to remove any distracting asphalt edges before edge categorization, leaving just the strong ones in the image. In order to identify each picture edge, the final step in edge classification is to determine whether the areas on each side of strong edges adhere to the six forces related with the 3 intensity qualities of shadow that have been presenteda materialchange edge or a shadow edge. An ROI in the entering colour Images are dervies while driving by utilising perspective information about the area and assuming flat road surface because The technique

focuses on identifying shadow borders on the road. This simplifies a captured road scene and reduces the amount of false positive finds outside the road surface

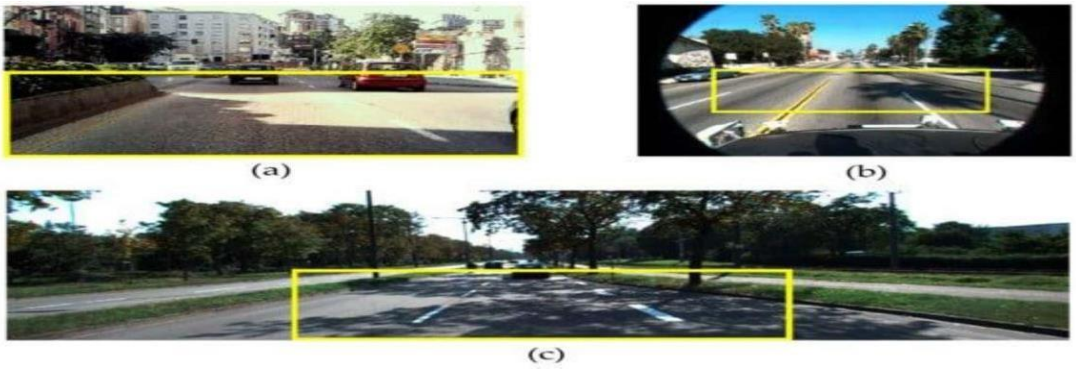


Fig 4. Shadow affected Road Lane-Line Images

The method's final step is edge classification, where each distinct edge Dk is labelled as a shadow edge if its lighter and darker (Gnon-sha,) reflectance components.meet the six chrominance constraints that correspond to the three chrominance properties of shadow. Otherwise, the edge is categorised as an boder due to the change of materials if one of the conditions is not satisfied.

Apply Gaussian filter to smooth the image and reduce noise.
Compute gradient magnitude and orientation using Sobel, Scharr, or other derivative operators.
Apply non-maximum suppression to thin the edges and retain only the strongest edge pixels.
Apply hysteresis thresholding to obtain a binary edge map:

a. Define two threshold values: a high brick T_high and a low brick T_low.
b. Set all edge pixels with magnitude more than T_high to be strong border.
c. Set all edge pixels with magnitude among T_low and T_high to be frail edges.
d. Set all remaining pixels to be non-edges.
e. Trace edge contours starting from strong edges and including weak edges that are connected to strong edges. Output the binary edge map.

Dk=I If all three of the following properties are true, the edge will be Shadow Edge and Material Change Edge. Otherwise

Property 1: The rgb surface reflectants caused by sunshine have a more or equivalent relationship than those caused by skylights.
Property 2: the predominant hue used for road reflectivity caused by sunlight is red, which is greater than blue and greater than or same to green. The green portion of the road reflectivity is also greater than the blue portion.
Property 3: Skylight and sunlight have less of an impact on the rgb proportion of the road reflectance than they do on the red-blue one. This finding holds true when Compare how the surface connections between green and red and green and blue have changed.

6 RESULT AND DISCUSSION

The road lane line identification system, which makes use of the decision tree algorithm and shadow reduction techniques, has shown good results in spotting lane lines in dim lighting.Using a training dataset of instances of lane lines and non-lane lines, the decision tree method successfully acquired the ability to identify extracted characteristics as belonging to a lane line or not. This enabled the system to filter out noise and false positives from the extraction process and detect lane lines precisely in a variety of illumination conditions. The lane line identification system was less affected by shadows because to the success of the shadow reduction strategies. The system was able to enhance the contrast of the lane lines and decrease the noise by utilising colour normalisation and shadow removal techniques.

The method was able to increase the lane lines' contrast and lessen the effect of shadows, which resulted in more precise lane line recognition.Overall, the system successfully detected lane lines with high accuracy and robustness under a variety of illumination circumstances, including hard situations including shadows. The lane line identification system's performance was improved by using the decision tree algorithm and shadow reduction techniques, which might also be used in advanced driver assistance or autonomous driving applications. It is important to keep in mind that the system's performance may be impacted by a number of variables, including the road's state, the camera's angle, and the speed of the vehicle. For some applications, further testing and optimisation may be necessary.

The below table indicates the accuracy percentage existing lane detection approach of the proposed lane-line detection works in the previous years and the revolution the image pixel that can used has input of your work.

Table 1. Performance analysis of existing lane detection approaches.

Methodology	Accuracy
Guassianfilter+Hough transform	92%
MSER+Hough	89%
HSV-ROI+Hough	91%
LaneNet	83%

Table 2. Image resolution in captured images

Resolution	Pixel
1080	1920*1080
720	1280*720
480	640*480

7 CONCLUSIONS AND FUTURE WORK

Road departure, one of the main contributing factors in single-vehicle and frontal collisions, is the cause of nearly one-third of all traffic accidents. Unintentional lane departure: When a driver moves towards and beyond the lane line without intending to do so, is a common cause of accidents. For lane understanding and navigation, automated cars use a variety of sensing technologies,.Since markers are already created for human vision, vision is the most conspicuous and is ready for application; LIDAR and GPS are significant complements. The LSS employs cameras to "read" the line points on the pavement and warn the person when the vehicle is getting close to the lines. The identical visual signals used by human drivers need to navigate the road, including as lane markings, road limits, and road colour and texture, must be taken into account by the machine vision technologies utilised in these systems.

Shadows on the road have a substantial impact on vision-based driving aid technologies, making it difficult to perform crucial operations like lane and road detection. Also, since shadow characteristics possibly shared by other objects in the scene, identifying shadows can be challenging. This project's objective was to create new chrominance properties in order to design an an effective method for detecting shadows that might be included into an onboard road detection system. for driver assistance. This was done in order to reduce the possibility of misclassifying shadows on both shadowed and unshadowed image areas.
The lighting of outdoor sceneries in sunny weather, which consists two sources of light with varying SPDs, such as sunshine and skylights, additionally, they impact on the road interface. Contrary to other techniques, we found While comparing shadowed and non-shadowed areas of the same material surface, it was found that the sunlight's offfers to the chromaticity of the non-shaded surface was considerable. then came up with three new chrominance characteristics for shadows. We presented a shadow edge detection approach for onboard devices. based on the six restrictions connected to these attributes. Although onboard systems only work with static images, our method compares the characteristics of regions across image edges to identify shadow boundaries and material changes. However, static backdrop applications can also be addressed because no prior scene knowledge, camera calibration, or spatio-temporal constraints are necessary.

There are three limitations to our method, the miscategorized of borders due to yellow road points and the absence of borders of overused and underexposed areas of a picture, despite the studies showing its effectiveness and dependability. Consideration of the possible shadow region's intensity can help with the former, and employing cameras with a higher dynamic range can help with the latter. We will fix the shortcomings of our approach in subsequent work and create a road detecting system for driver assistance.

8 REFERENCES

1. Cafiso, S.; Di Graziano, A. Evaluation of the effectiveness of ADAS in reducing multi-vehicle collisions. Int. J. Heavy Veh. Syst. 2012, 119, 188 206.
2. A. M. Muad, A. Hussain, S. A. Samad, M. M. Mustaffa and B. Y. Majlis, "Implementation of inverse perspective mapping algorithm for the development of an automatic lane tracking system", *in* TENCON 2004. 2004 IEEE *Region 10 Conference*, vol. 1, pp. 207-210, 2004.
3. H. Wang and Q. Chen, "Real-time lane detection in various conditions and night cases", *in Proceedings of the IEEE Intelligent Transportation Systems Conference*, pp. 17-20, 2006.
4. Cafiso, S.; Pappalardo, G. Safety Effectiveness and Performance of Lane Support Systems for Driving Assistance and Automation Experimental test and Logistic regression for Rare events. Accid. Anal. Prev. 2020, 148.
5. Adachi, E., Inayoshi, H., and Kurita, T. 2007. Estimation of lane state from car-mounted camera using multiple-model particle filter based on voting result for one-dimensional parameter space. In Proceedings of the IAPR Conference on Machine Vision Applications (MVA'07). 323—326.
6. Chapuis, R., Marmoiton, F., Aufrère, R., Collange, F., and Derutin, J. 2000. Road detection and vehicles tracking by vision for an on-board acc system in the velac vehicle. In Proceedings of the 3rd International Conference on Information Fusion (FUSION'00). WEB5/11 - WEB5/18
7. Danescu, R. and Nedevschi, S. 2008. Adaptive and robust road tracking system based on stereovision and particle filtering. In Proceedings of the 4th International Conference on Intelligent Computer Communication and Processing (ICCP'08). 67--73.
8. Austroads. Infrastructure Implications of Pavement Markings for Machine Vision. Publication No: AP-R633-20. Available online: https://austroads.com.au/publications/connected-and-automated-vehicles/ap-r633-20 (accessed on 6 October 2020).
9. Cafiso, S.; Taormina, S. Texture analysis of aggregates for wearing courses in asphalt pavements. Int. J. Pavement Eng. 2007, 8, 45 54.

10. Farah, H.; Bhusari, S.; van Gent, P.; Mullakkal Babu, F.A.; Morsink, P.; Happee, R.; van Arem, **B.** Empirical Analysis to Assess Odd of Lane Keeping System Equipped Vehicles Combining Objective and Subjective Risk Measures. IEEE Trans. Intell. Transp. Syst. 2020, 1 10.
11. Alvarez, J.M.; Lopez, A.M. Road Detection Based on Illuminant Invariance. IEEE Trans. Intell. Transp. Syst. 2011, 12, 184 193.
12. Song, Y.; Ju, Y.; Du, K.; Liu, W.; Song, J. Online Road Detection under a Shadowy Traffic Image Using a Learning-Based Illumination-Independent Image. Symmetry 2018, 10, 707.
13. Yoo, J.H.; Lee, S.G.; Park, S.K.; Kim, D.H. A Robust Lane Detection Method Based on Vanishing Point Estimation Using the Relevance of Line Segments. IEEE Trans. Intell. Transp. Syst. 2017, 18, 3254 3266.
14. Hoang, T.M.; Baek, N.R.; Cho, S.W.; Kim, K.W.; Park, K.R. Road Lane Detection Robust to Shadows Based on a Fuzzy System Using a Visible Light Camera Sensor. Sensors 2017, 17, 2475.
15. Cucchiara, R.; Grana, C.; Piccardi, M.; Prati, A. Detecting moving objects, ghosts, and shadows in video streams. IEEE Trans. Pattern Anal. Mach. Intell. 2003, 25, 1337 1342.
16. Khan S., Bennamoun M., Sohel F. et al.: 'Automatic shadow detection and removal from a single image', IEEE Trans. Pattern Anal. Mach. Intell., 2016, 38, pp. 431 446.