

Case Study of Self Driving Car
Lane Detection using Digital Image Processing

Project report submitted in partial fulfilment of the requirement for the degree of Bachelor of
Technology

in

Computer Science and Engineering/Information Technology

By

Aryan Bathla 191434

Under the supervision of

Dr. Himanshu Jindal, Assistant Professor (SG)

To



Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology Waknaghat,

Solan 173234, Himachal Pradesh

Certificate

Candidate's Declaration

I hereby declare that the work presented in this report entitled "**Case Study of Self Driving Car : Lane Detection using Digital Image Processing**" in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering/Information Technology** submitted in the department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from January 2023 to May 2023 under the supervision of **Dr Himanshu Jindal, Assistant Professor (SG)**, Department of Computer Science and Engineering.

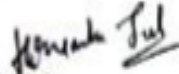
I also authenticate that I have carried out the above mentioned project work under the proficiency stream Data Science.

The matter embodied in the report has not been submitted for the award of any other degree or diploma.



Aryan Bathia 191434

This is to certify that the above statement made by the candidate is true to the best of my knowledge.


Dr. Himanshu Jindal

Assistant Professor (SG)

Department of Computer Science and Engineering

Dated: 27th April, 2023

PLAGIARISM CERTIFICATE

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT
PLAGIARISM VERIFICATION REPORT

Date: 1st May 2023

Type of Document (Tick): Ph.D Thesis M.Tech Dissertation/ Report B.Tech Project Report Paper

Name: Ashwin Bhatla Department: Computer Science Enrolment No 191434

Contact No. 9219935033 E-mail: ashwinbhatla2@gmail.com

Name of the Supervisor: Dr. Vinod Kumar Jindal


Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): CASE STUDY OF SELF DRIVING CARS: LANE DETECTION USING DIGITAL IMAGE PROCESSING

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

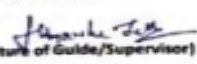
Complete Thesis/Report Pages Detail:

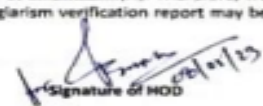
- Total No. of Pages = 50
- Total No. of Preliminary pages = 11
- Total No. of pages accommodate bibliography/references =


(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found Similarity Index at 19 (%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.


(Signature of Guide/Supervisor)


(Signature of HOD)

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	Excluded	Similarity Index (%)	Generated Plagiarism Report Details (Title, Abstract & Chapters)	
Report Generated on	<ul style="list-style-type: none"> • All Preliminary Pages • Bibliography/Images/Quotes • 14 Words String 		Word Counts	
			Character Counts	
		Submission ID	Total Pages Scanned	
			File Size	

Checked by _____ Librarian

Name & Signature

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.jutb@gmail.com

ACKNOWLEDGEMENT

Firstly, I express my heartiest thanks and gratefulness to almighty God for His divine blessing makes it possible to complete the project work successfully.

I am really grateful and wish my profound indebtedness to Supervisor Dr. Himanshu Jindal, Associate Professor (SG), Department of CSE Jaypee University of Information Technology, Wakhnaghat. Deep Knowledge & keen interest of my supervisor in the field of “Computer Science” to carry out this project. Her endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts and correcting them at all stages have made it possible to complete this project.

I would like to express my heartiest gratitude to Dr. Himanshu Jindal, Department of CSE, for his kind help to finish my project.

I would also generously welcome each one of those individuals who have helped me straightforwardly or in a roundabout way in making this project a win. In this unique situation, I might want to thank the various staff individuals, both educating and non-instructing, which have developed their convenient help and facilitated my undertaking.

Finally, I must acknowledge with due respect the constant support and patients of my parents.

Group No. - 54

Aryan Bathla

191434

TABLE OF CONTENT

Content	Page No.
Declaration by Candidate	I
Certificate by Supervisor	II
Acknowledgment	III
Abstract	IV
Chapter 1: INTRODUCTION	
1.1 General Introduction	1
1.2 Problem Statement	2
1.3 Objectives	3
1.4 Methodology.....	4

1.5 Organisation.....13

Chapter 2: LITERATURE SURVEY

2.1 Elaborative literature review.....14

2.2 Table of Comparison.....17

Chapter 3: SYSTEM DEVELOPMENT

3.1 Analytical20

3.2 Computational.....21

3.3 Mathematical.....22

Chapter 4 : Experiments & Result Analysis

4.1 Analysis24

4.2 Methods.....26

Chapter-5 Conclusions

LIST OF FIGURES

Figure No.	Description	Page No.
1.	Gaussian Kernel	4
2.	Sobel Kernel	5
3.	Non Maximum Suppression	5
4.	Hysteresis Thresholding	6
5.	Triangular Mask	7
6.	Block Diagram	9
7.	Block Diagram	10
8.	Research Objective's Breakdown Structure	11
9.	Minimalistic approach-based lane detection	13
10.	Pixel Summation Output	14
11.	Output	14
12.	Boundary Detection using Feature-Based Methods	15
13.	Flowchart	15
14.	Architecture of Proposed Lane-GAN	16

15.	Training dataset	21
16.	Input image and Gaussian Distribution	22
17.	Gradient Vector Convolutional Results	23
18.	Edge Detection	23
19.	Non-Maxima Points	24
20.	Hysteresis Thresholding	24
21.	Image Space	25
22.	Hough Space	25
23.	Image Space	26
24.	Hough Space	26
25.	Image Space	26
26.	Hough Space	26
27.	Hough Space	27
28.	Image Space	27
29.	Hough Space	27
30.	Image Space	27
31.	Hough Transformation	28
32.	Polar System	28
33.	Sinusoidal Representation	29
34.	Sinusoidal Representation for values of X and Y	29
35.	Preprocessed Image (Grey-Scale)	31
36.	Gaussian Blur	32

37.	Canny Edge Output	33
38.	Sampling of an Image	38
39.	Quantization of an Image	39
40.	Experimental Canny edge Output	44
41.	Lines by Canny Edge and Hough Transform	45
42.	Final Output/Result	45
43.	Sobel and Canny Edge Detection	45

List of Tables

S.No	Description	Page No.
1	Literature Survey	19
2	Sampling and Quantization	38

ABSTRACT

Self-driving vehicles have expanded dramatically over the last few years. The introduction of autonomous vehicles will alter human existence. Large datasets and powerful computers are needed for such systems. An emulator exists that makes it simple to produce the desired amount of photos of a moving vehicle. On the basis of only the available photographs, the challenge tried to anticipate and detect traffic lanes.

We use our eyes to choose where to go when we are driving. The lines on the road show us where the lanes are, and we can guide the car using them as a constant reference. Naturally, one of the first features we would like to build into a self-driving car is the ability to detect and recognise lane lines using an algorithm. The problem of lane detecting is difficult to tackle. It has long caught the interest of the computer vision community..

Lane detection, which has proven to be challenging for computer vision and machine learning algorithms to tackle, is fundamentally a multi-feature detection problem. We offer a technique based on image processing that uses Canny Edge Detection and region masking.

A growing technology used in cars to enable autonomous navigation is lane detecting. The majority of lane-detection systems are built for properly designed roads and rely on the presence of markings. The main drawback of these methods is that they can yield incorrect results or fail to function at all when there are indistinct markings or none at all. This paper reviews one such method for spotting lanes on an unmarked road before moving on to a better method. Both methods only use data from vision or cameras and are based on digital image processing techniques. The primary goal is to acquire a real-time curve value that will let the driver or autonomous vehicle make necessary turns and stay on the road.

CHAPTER 1

INTRODUCTION

1.1 GENERAL INTRODUCTION

With the expansion of knowledge and communication technology as well as vehicle design, the competition to develop and commercialize intelligent and autonomous cars has become increasingly fierce. The comfort of the driver, the steadiness of the automobile, and improved traffic efficiency are all features of these vehicles. Through a human-machine interface, the advanced driving assistance system (ADAS) assists drivers by using the lane departure warning system, lane keeping assistance system (LKAS), front collision warning system, and smart parking assistant system (SPAS). In order to keep the car on track and close to the centre of the lane when the system notices it straying from its lane, LKAS continuously provides a modest amount of counter-steering force [1]. SPAS provides services that simplify parking. to assist a driver in getting into the right position before starting Automatic steering assistance along a predefined path is offered to help a driver get to the appropriate starting position for beginning to reverse into a parking space [2]. An autonomous vehicle can reach its destination without the driver needing to focus on the road in front of them. The route tracking strategy and development of a lane detection system using techniques utilized in autonomous and intelligent automobiles are explained in this paper. The most popular lane line detectors are the Hough transform and convolution-based techniques. Lane detection is the course of detecting lane markers on the road and thereafter presenting these locations to an intelligent system. Intelligent vehicles cooperate with the infrastructure to achieve a safer environment and better traffic conditions in intelligent transportation systems. The use of an automated lane detecting system can range from as simple as indicating out lane positions to the user on an external screen to more complex responsibilities like forecasting a lane change in real time to avoid collisions with other vehicles.

1.2 PROBLEM STATEMENT

Finding sufficient real-world data to be fed into the potent deep learning algorithms that are needed to carry out tasks like lane detection is one of the key challenges in edge detection in self-driving automobiles. For it to develop the algorithms for self-driving vehicle applications, a significant amount of data must be gathered and labeled. Collecting and labeling real-world data takes time and money, and it is impractical to test every conceivable event in reality, such as a car smashing at high speeds into a brick wall.

For self-driving automobiles, the issue of road lane detection and signal detection is to automatically identify lanes and traffic signs. The ability to recognize traffic signs and road tracks from video frames throughout the self-driving process is entirely owing to advances in image processing and deep learning. In this study, the vehicle incorporates YOLO version 1 for object detection, a polynomial regression model with thresholding for lane guidance, and a controller to coordinate data across the systems. The suggested methods can be applied to steering suggestion, object identification, object location detection (left, front, or right), and road lane guidance.

A crucial challenge is the detection and identification of objects in real time. A notable instance of a security failure involves the 2016 Tesla autopilot accident, in which the vehicle's sensors were mixed by sunlight and the system did not recognize the truck approaching from the right, resulting in the crash. Real-time identification and detection of objects is a critical task. The Tesla auto-pilot disaster in 2016 is a well-known example of a security failure. In that incident, the vehicle's sensors were masked by the sun and the algorithm failed to detect a truck approaching from the right, which resulted in the collision.

1.3 OBJECTIVES

Self-driving vehicles are not only a reality today, but their technology also provides a preview of what complicated technology can look like in the future. To produce the greatest autonomous vehicles possible, a large number of specialists from diverse fields have collaborated. Lane detection is already a standard function in vehicles, and training models for it needs a lot of real-world data. This information must take into account all possible weather conditions, road topographies, driver actions, and drivers who are vulnerable in order to construct completely autonomous vehicles. With the eventual goal of using this system to create data for training models in self-driving cars, the idea for the system is to build a system for recognising road lanes using Python and OpenCV. To enable the model to focus on recognising lanes, the algorithm will employ Canny edge detection to recognize the margins of the lane and a masking function to hide undesirable elements in images like trees, rocks, and electrical lines. The suggested method accurately determines lanes in real-time by using the Hough Transform to detect and draw lanes.

The following objectives are intended to achieve through this research:

1. To utilize computer vision algorithms i.e., canny edge detection algorithm for detecting lane edges in real-time.
2. To develop a system which would detect lanes and develop a large volume of datasets for building a lane detection system in self-driving cars.
3. To contribute to minimizing the time complexity in terms of collecting real life dataset for developing lane detection system in self-driving cars

1.4 METHODOLOGY

1.4.1 Hough Transform

The threshold is automatically selected for threshold processing based on with edge information once the Canny operator has located the edge of the image's region of interest. Three limitations from the angles and lane width are suggested to enhance the Hough transform's recognition of lane lines. As a result, rectilinear regression is used to fit the proper lane lines.

1. Noise Reduction

Noise may be an important issue which frequently results in erroneous detection, as is the case with all edge detection techniques. The image is convolved (smoothed) using a 5x5 Gaussian filter to lessen the detector's sensitivity to noise. In order to do this, a kernel of normally distributed values is used to run across the entire image, adjusting each pixel's value to the weighted mean of its nearby pixels. In this case, the kernel is 5x5.

$$\mathbf{B} = \frac{1}{159} \begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} * \mathbf{A}$$

Fig 1: 5x5 Gaussian Kernel. The asterisk(*) denotes convolution operation.

2. Intensity Gradient

The smoothed image is then used using a Sobel, Roberts, or Prewitt kernel (Sobel is used in OpenCV) to identify if the borders are horizontal, vertical, or diagonal.

$$\text{Edge_Gradient } (G) = \sqrt{G_x^2 + G_y^2}$$
$$\text{Angle } (\theta) = \tan^{-1} \left(\frac{G_y}{G_x} \right)$$

Fig 2: Sobel kernel for calculating the primary derivative of horizontal and vertical directions

3. Non-maximum suppression

"Thin" is subjected to non-maximum suppression, which successfully sharpens the edges. The value for each pixel is examined to see if it is the maximum locally in the gradient's estimated direction. A is positioned vertically on the edge. The values of the pixels of B and C are contrasted to pixel values of A to see if A is a local maximum since the gradient is perpendicular to the edge direction. If A is the local maximum, the next point is examined for non-maximum suppression. If not, A is suppressed and its pixel value is set to zero.

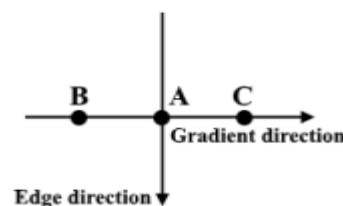


Fig 3 : Non-maximum suppression on three-point

4. Hysteresis Thresholding

Strong pixels are proven to be present in the final representation of edges following non-maximum suppression. To identify whether weak pixels represent an edge or noise, further analysis is necessary. Applying two preset threshold values, minVal and maxVal , we decide which pixels are edges and which pixels aren't edges and should be eliminated. Edges are any pixels with gradients in intensity bigger than maxVal . Pixels having an intensity gradient within minVal and maxVal are only considered edges if they are linked to a pixel that has an intensity gradient beyond maxVal .

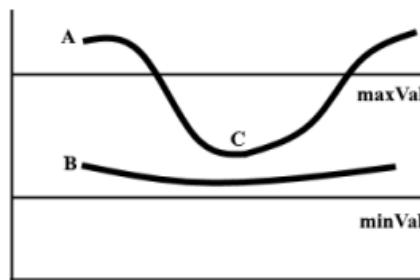


Fig 4: Hysteresis Thresholding on two lines

5. Segmenting line area

To increase the efficiency of our later stages, we will manually create a triangular mask to segment that roadway area and eliminate the unnecessary portions of the frame. The three coordinates, denoted by the green circles, will be used to define the triangular mask.

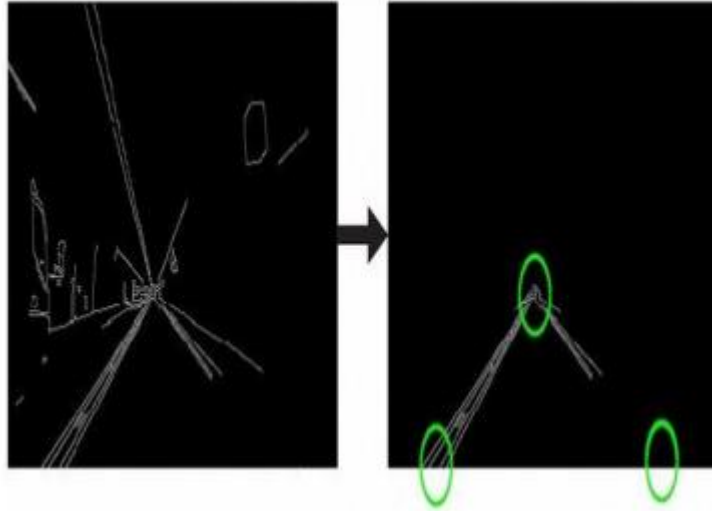


Fig 5 : The triangular mask is going to be defined by three coordinates, indicated by the green circles.

1.5 ORGANISATION

We have explained the fundamental principles of self-driving cars in this project report, as well as how lane detection, a crucial element of self-driving cars, operates. Additionally, in order to acquire the finest computing capabilities for the same, we will be experimenting with various methods and methodologies.

The fundamental concept of self-driving automobiles was covered in part one. Section two, or the literature review, is divided into two sections. Section three, or the system development portion, is divided into two sections. The performance analysis and comparisons are presented in Section 4. The conclusion/final solution will be presented in Section 5.

CHAPTER 2

LITERATURE SURVEY

Tullimalli Sarsha Sree and Sandeep Kumar Sathapathy

In order to locate the lane lines along the road, we're employing a software programme that we are currently working on. The development of algorithms for self-driving cars depends on their capacity to distinguish and follow lanes. Here, we'll estimate the price of developing a software pipeline for tracking traffic lanes using computer vision techniques. We'll approach this task using two different approaches. They are the hough transform technique and convolutional neural networks (CNN). However, for safe driving, lane markings may be a vital point of reference. This work suggests a modified Hough transform-supported lane line recognition algorithm to increase the accuracy and speed of lane line recognition.

The threshold is automatically selected for threshold processing based on with edge information once the Canny operator has located the edge of the image's region of interest. Three limitations from the angles and lane width are suggested to enhance the Hough transform's recognition of lane lines. As a result, rectilinear regression is used to fit the proper lane lines.

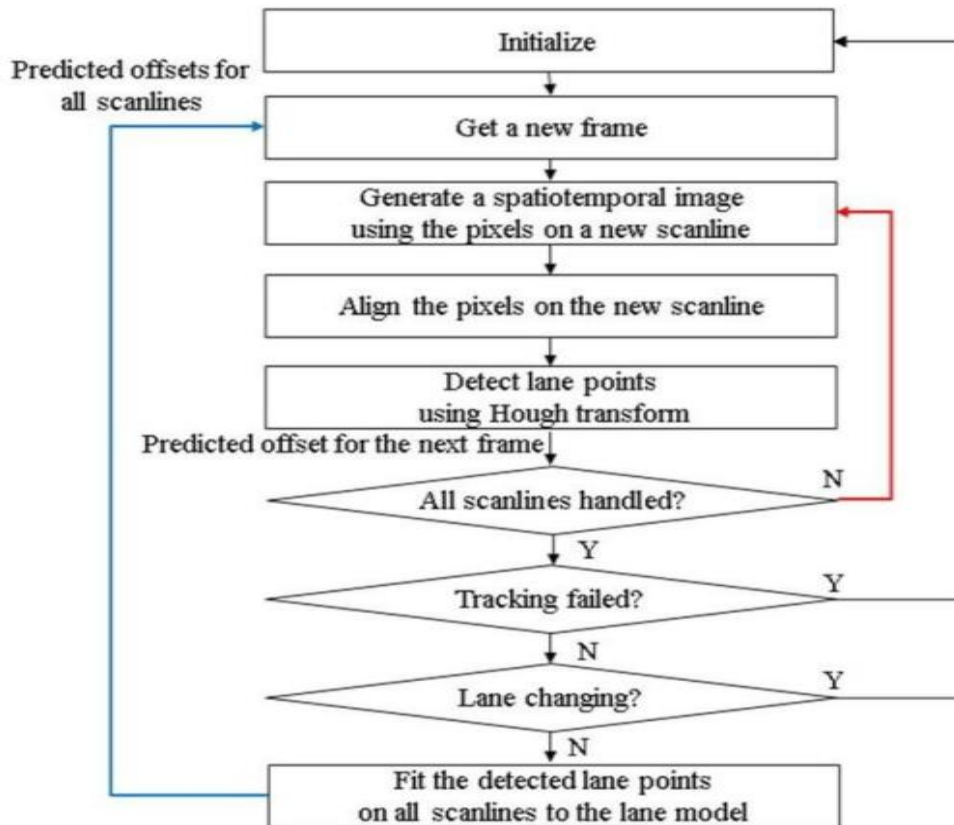


Fig 6 : Block Diagram

Nidhi Lakhani, Ritika Karande, Deep Manek, Vivek Ramakrishnan

The rapid societal development has led to the rise of the automobile as a mode of mobility. On the congested road, there are a rising number of different kinds of automobiles. Intelligent car systems have made use of lane detection, a hot topic in the fields of computer vision and machine learning. It is a new field that has applications in the business sector. The vehicle's position and trajectory with respect to the lane are reliably approximated by the lane detection system, which uses lane markers in a complex environment. The lane exit warning system heavily relies on lane detection at the same time. The two fundamental components of lane detection are detection of edges and line detection. In the method of lane detection, line detection is just as crucial as edge detection.

The Hough transform and convolution-based methods are the most often used lane line detectors. The process of recognizing lane markings on the road and then presenting the positions to an intelligent system is known as lane detection. In intelligent transportation systems, intelligent cars work together with the infrastructure to create a more secure atmosphere and better traffic conditions. The use of a lane detection system can range from straightforward jobs like pointing out lane positions to the user on an external display through more complex ones like anticipating a lane change in an instant to avoid colliding with other vehicles.

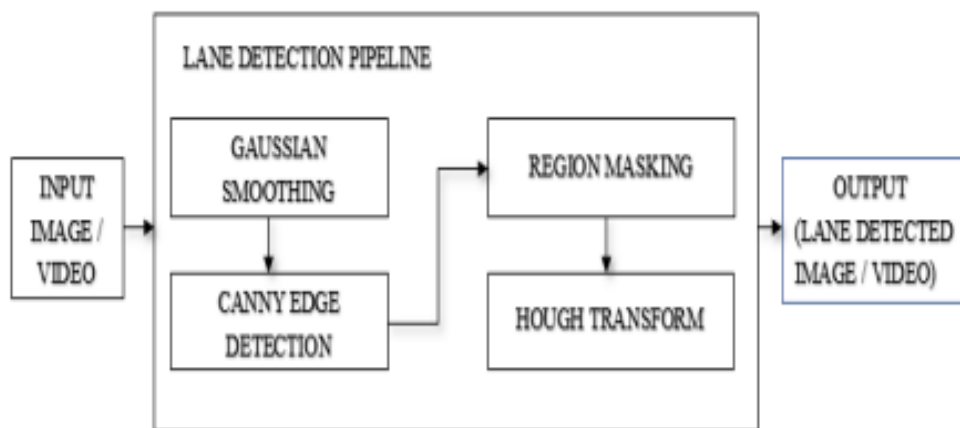


Fig 7: Block diagram

Iftikhar Ahmad , Jin Ho Lee , and Soon Ki Jung

Every day, humans make a variety of decisions, many of which are influenced by the sensory data we gather from our environment. The majority of this perception when related to driving is visual. Autonomous vehicles, commonly referred to as self-driving automobiles, are built to be able to recognise items in their environment and decide how to react to them quickly. This creates a number of computer vision issues for autonomous vehicles, including identifying pedestrians, other vehicles, lanes, and traffic signs. This study specifically addresses lane detection, which is a vital component of a vehicle's movement planning.

In this approach, a lane detection method suggested for the Asphalt 8: Airborne real-time racing game utilising the Canny edge detection technique. The programming language's Python Imaging Library (PIL)'s ImageGrab module is used to access the game's screen. OpenCV is used to determine the lanes' boundaries using Canny edge detection, a popular technique in computational image processing, and a masking algorithm was employed to remove obstructions including trees, rocks, and cables. The game's lanes were marked and drawn using the Hough Transform. The gaming environment features real physics, graphics, and a range of settings, including lane-keeping aids like abrupt corners, slopes, and various weather conditions.

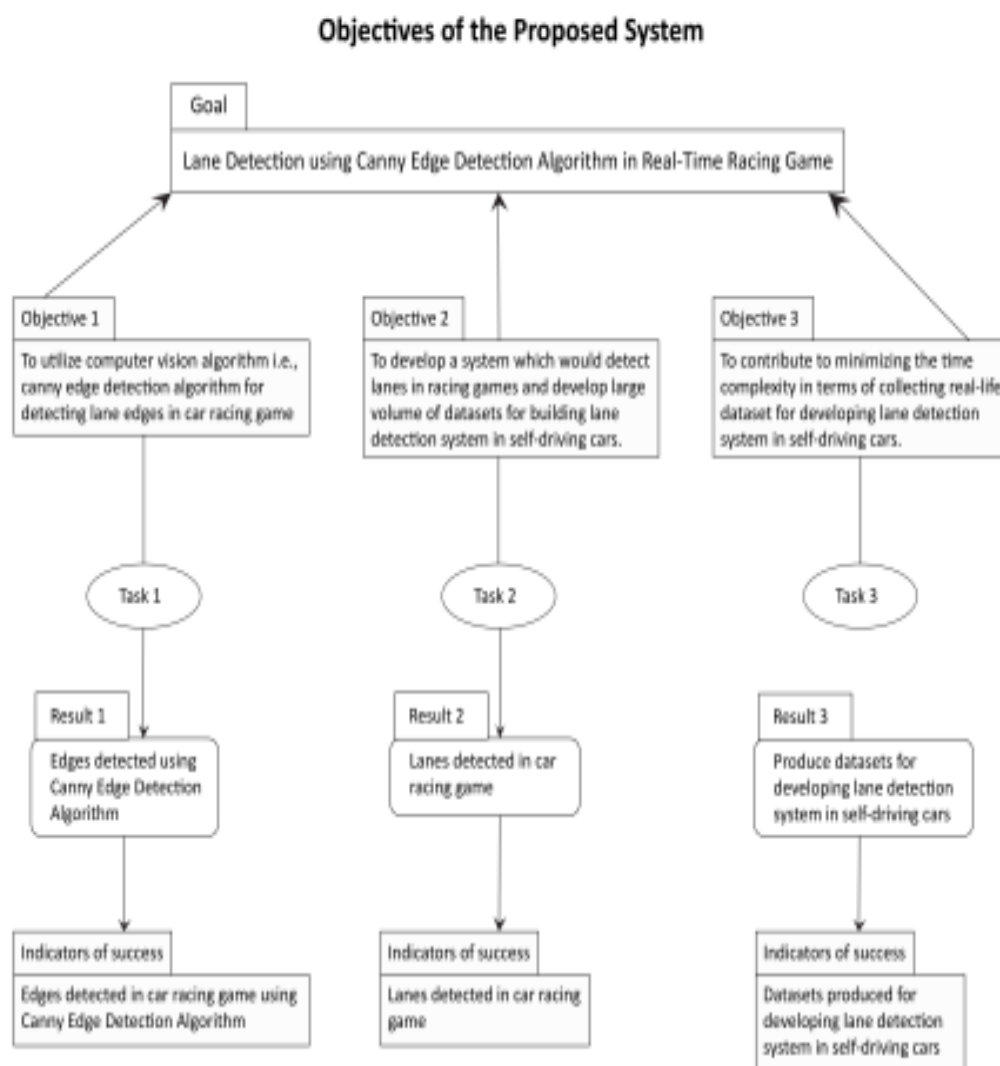


Fig 8 : Research Objective's Breakdown Structure (RBS) of our study

Raja Muthalagu, Anudeep Sekhar Bolimera, Dhruv Duseja, Shaun Fernandes

In the modern world, people spend a lot of time driving or dealing with other automobiles while walking on the streets. 1.35 million people worldwide lose their lives in traffic accidents every year, and every day, up to 3,700 people die in collisions involving buses, lorries, cars, motorbikes, bicycles, or people (Singh, 2015). Worldwide, the number of cars on the road is also rising quickly (Bellis et al., 2008). Researchers have found that the majority of traffic accidents are caused by human error.

A fully autonomous self-driving automobile is being developed as a result of the Advanced Driver Assistance Systems (ADAS) that have been created in recent years to improve passenger safety and comfort (Lu et al., 2005; Bengler et al., 2014; Liyong et al., 2020). Perception, planning, and control are the three main components that make up self-driving car technology. Researchers have put in a lot of effort to create new methods for increasing driving safety and lowering traffic accidents. The majority of ADAS systems use a range of sensors to detect lanes and objects on the highway. The detection of the lanes and objects is proposed using a variety of camera vision techniques. Our contributions to this work are outlined as follows:

1. To identify the straight lane lines, a simple lane identification method is suggested.
2. A CNNs-based model that is capable of learning to drive a vehicle using the driver's driving data has been developed. This method teaches the car how to drive by mimicking the actions of its owner.
3. The basic remote-controlled automobile has front-facing cameras that collect video, and it employs the suggested delayed detection approach to identify lanes and objects.

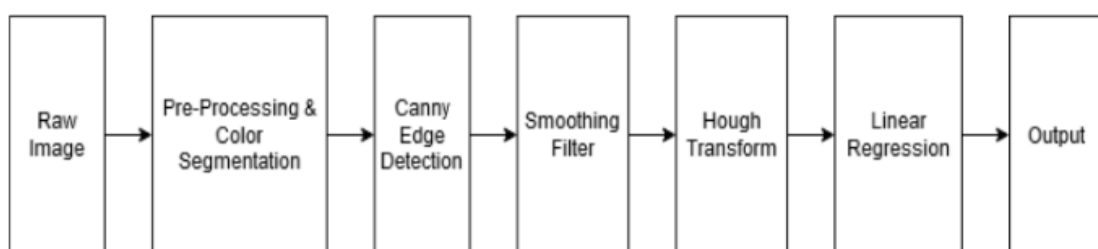


Fig 9: Minimalistic approach-based lane detection

Vighnesh Devane, Ganesh Sahane, Hritish Khairmode, Gaurav Datkhile

For lane curve fitting, the method [4] proposes an effective variation of the sliding window algorithm. This enhanced variant has the benefit of allowing sliding windows to be used even on uneven lane markers. In the second way, the road borders from the binary image are used to apply this technique. The curve value and the car's offset are then calculated using the geographic coordinates of the roadway boundaries.

The overall goal of the project is to make lane detection possible on roads with limited visibility or worn-out lane markings. With the aid of image warping, thresholding, and techniques like pixel summation and sliding window algorithm, this research attempts to build lane detection for an efficient autonomous car. Finally, the benefits and drawbacks of the aforementioned techniques as well as the most appropriate application scenarios have been discussed.

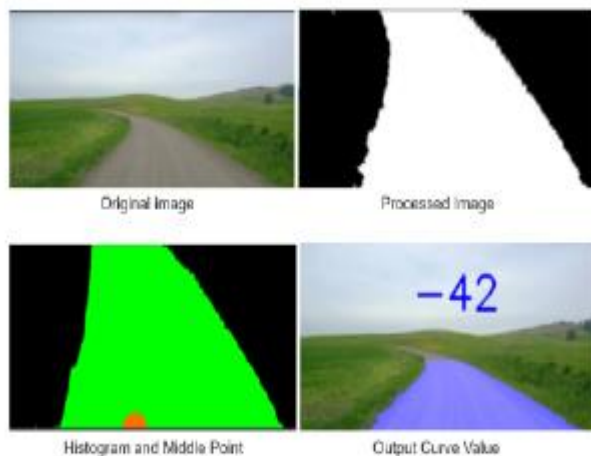


Fig 10: Pixel summation output



Fig 11: Output

Jamel Baili, Mehrez Marzougui, Ameer Sboui, Samer Lahouar, Mounir Hergli

Lane detection has been the subject of numerous studies in image processing. A summary of

these studies reveals that they can be divided into two main groups. In the first, markers for lanes in an input picture from a rear-view camera are recognised using the bird's-eye view transform. The second group makes advantage of the front-mounted camera. In the latter case, various image processing methods have been developed, such as the Probability of Picture Shape (LOIS) algorithm, the B-Snake technique, and others. These algorithms employ a variety of ways to extract features from an image, such as edges, using a feature-based approach.

The suggested method has performed well, but it has to be tweaked to accommodate inclement weather (such as rain and snow) and poor lighting (nighttime). We intend to develop an LDW System in an integrated processor in future work and assure a robust monitoring mode using information from TLC calculation.

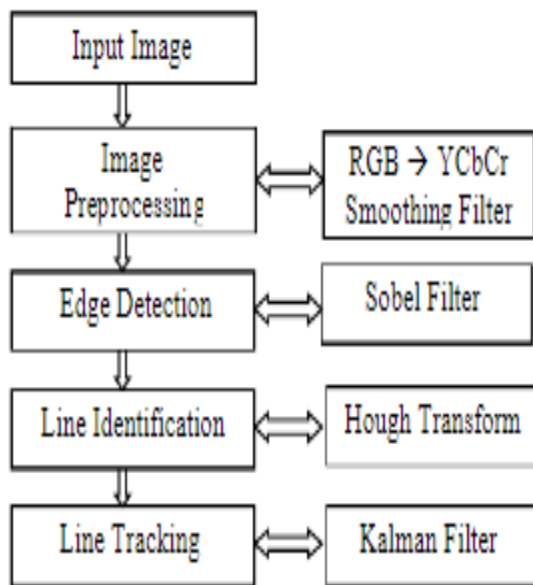


Fig 12 : Flowchart of lane boundary detection using feature-based method

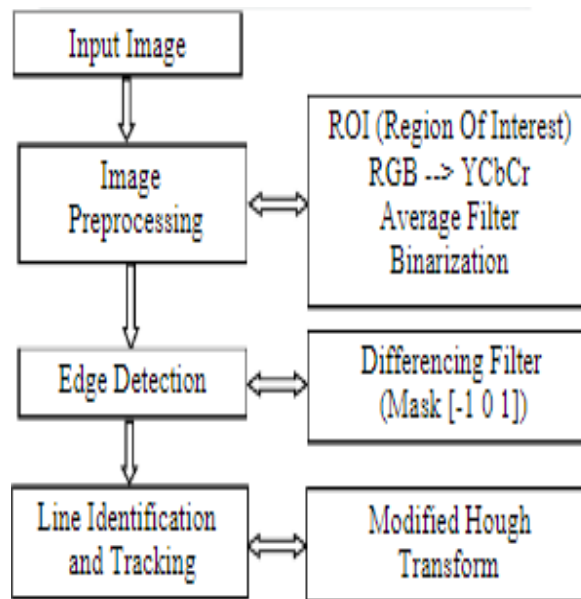


Fig 13 : Flowchart of our methodology

Yan Liu , Jingwen Wang , Yujie Li , Canlin Li

This paper proposes a lane detection system to address the challenge of lane detection in hazy settings. The following list highlights the main contributions made by this study.

1. This page includes a dataset of blurred lane lines.

2. To improve the characteristics of the lanes & increase the effectiveness of blurring lane recognition in complicated road settings, an upgraded GAN is utilised.
3. The proposed method outperforms existing state-of-the-art detectors in high speed and difficult road conditions (line curves, dirty lane line, illumination change, occlusions), resulting in a significant improvement over the current state-of-the-art detectors.

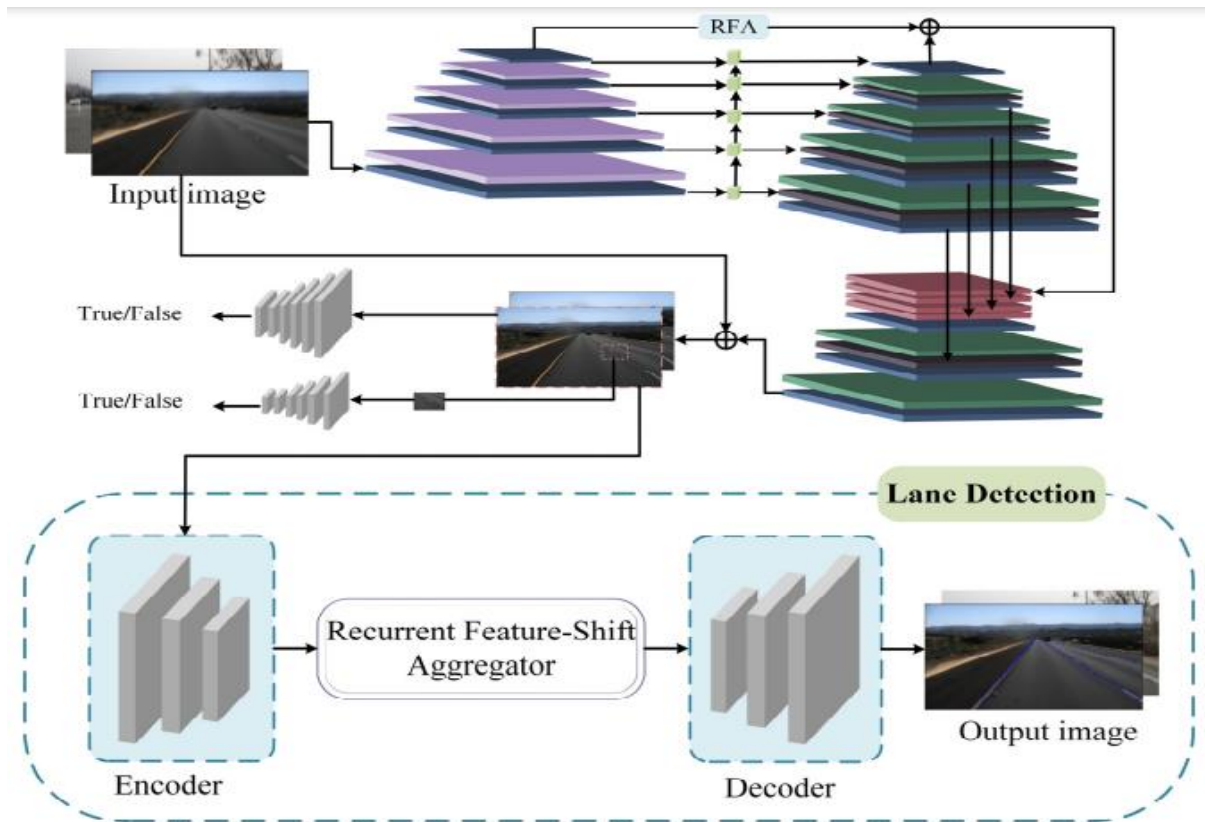


Fig 14: Architecture of Proposed Lane-GAN

Zequn Qin, Huanyu Wang, and Xi Li

Applying global image features-based row-based selecting to lane detection. In other words, our approach involves leveraging the global features to choose the appropriate lanes for each predefined row. Lanes are modeled in our formulation as row anchors are a sequence of horizontal points in predetermined rows. The initial stage in representing places is gridding. The location is separated into several cells on each row anchor. This makes it possible to compare the identification of lanes to the selection of specific cells over predetermined row anchors.

According to the suggested definition, lane detection is a row-based selection problem that needs to be solved using global features. The issue of speed and no-visual-clue can be

solved in this way. It is also suggested to employ structural loss to explicitly model lane prior knowledge. Both the qualitative and quantitative testing support the value of our approach and the structural loss. Particularly, our model with the Resnet-34 backbone could reach the highest levels of accuracy and speed. Even at the same resolution, a lightweight Resnet-18 version of our approach was capable of 322.5 FPS with a competitive performance.

Ling Ding, Huyin Zhang, Jinsheng Xiao, Cheng Shuang Shejie Lu

The approach has its roots in a roadway segmentation and lane detection technique and incorporates the cavity convolution used in DeepLabv1 and LMD algorithms as well as the discriminant loss function used in LaneNet. To increase the receptive field, void convolution is used in place of the extracted feature part's common convolutional layer. The discriminant loss function stands out due to its simplicity of integration into various network structures and the fact that instance segmentation is accomplished through post-processing. Additionally, some works on denoising images initial processing have been discussed in earlier articles.

The correctness and robustness in the suggested method are confirmed after being tested on different data sets and lanes in various weather conditions.

The algorithm has a significantly slower detection speed but a much higher detection accuracy, especially when it comes to the recognition of corners and false actual lane lines. In comparison, the deep learning algorithm can produce precise detection and has none of these issues when it comes to effective detection.

Y. Wang, E. K. Teoh and D. Shen

The lane model is essential for lane detecting. The lane modelling must make a few assumptions about the real structure of the road in order to fully recover 3D data from the 2D static image. As we focus on creating the 2D lane model in this work, both sides of the road borders are taken to be in line on the ground plane.

A fresh B-Snake-inspired lane model has been developed to describe the viewpoint effect that results from general lane borders (or markings). It can depict a wider range of lane structures than other lane models, like straight and parabolic models. Here, the challenge of finding the middle of the lane and the challenge of detecting the two ends of the lane.

Gonzalez, J.P., Ozguner

Here, a vision for a system of intelligent cars is suggested. To find lane markers, the algorithm makes use of the properties of the road's grey level histogram. The relationships between each lane marker are then studied using a decision tree, and structures specifying the lane boundaries are created. Additionally, the system produces images that can be utilised in pre-processing stages in algorithms for lane detection, lane tracking, or obstacle detection. The system operates in real-time at roughly 30 Hz speeds.

The two suggested approaches to the lane detecting problem turned out to be reliable solutions. While requiring very little computational effort, the histogram-based segmentation outperformed many conventional methods. The methodology enables us to take into account image attributes that other approaches would have to forgo due to processing restrictions. The technique can be modified to be used for applications like off-road navigation or terrain classification. The decrease in processing time acquired and the decrease in data to be processed by the following step (we are analysing just under half of the elements in the image) should both be taken into consideration. A combined approach employing edge detection and histogram-based segmentation might produce an incredibly trustworthy result if additional processing capacity were available.

Table 1: Literature Survey

S.No	Authors	Advantages	Disadvantages
1	Tullimalli Sarsha Sree and Sandeep Kumar Sathapathy	This study proposes a lane line identification algorithm supported by modified Hough transform to improve the precision and real-time efficiency of lane line detection.	More Image Focused than Video graphic Input

2	Nidhi Lakhani, Ritika Karande, Deep Manek, Vivek Ramakrishnan	It is a new field that has applications in the business sector. The vehicle's position and trajectory with respect to the lane are reliably approximated by the lane detection system, which uses lane markers in a complex environment.	Time Consuming in video graphic inputs
3	Raja Muthalagu, Anudeep Sekhar Bolimera, Dhruv Duseja, Shaun Fernandes	Accurate Results	To combat overfitting, initially the use of Dropout was made. However, the performance was much worse.
4	Vighnesh Devane, Ganesh Sahane, Hritish Khairmode	Overall purpose of this project is to enable lane detection in poor road conditions	More Image Focused than Video graphic Input
5	Jamel Baili, Mehrez Marzougui, Ameer Sboui, Samer Lahouar, Mounir Hergli	Different algorithms for image processing have been created in this latter situation, including the Likelihood of Picture Shape algorithm, the B-Snake algorithm etc.	It has to be tweaked to accommodate inclement weather (such as rain and snow) and poor lighting (nighttime).
6	Yan Liu , Jingwen Wang , Yujie Li , Canlin Li	Aiming at the difficulty of lane detection in blurred scenarios, a lane detection network a blurred image is proposed in this article.	Time Consuming in video graphic inputs
7	Zequn Qin, Huanyu Wang, and Xi Li	Feature aggregation method for high-level semantics and low-level visual information is also depicted.	proposed formulation regards lane detection as a problem of row-based selecting

8	Ling Ding, Huyin Zhang, Jinsheng Xiao, Cheng Shuand Shejie Lu	To increase the receptive field, void convolution is used in place of the extracted feature part's common convolution layer.	the algorithm is much slower in detection speed
9	Y. Wang, E. K. Teoh and D. Shen	A novel B-Snake inspired lane model has been devised that represents the viewpoint effect of parallel lines.	More Image Focused than Video graphic Input
10	Gonzalez, J.P., Ozguner	The histogram-based segmentation outperformed many conventional methods.	In light to medium traffic scenes, the high level classifier performed quite well; but, in heavy traffic scenes, it performed less well.

CHAPTER 3

SYSTEM DEVELOPMENT

3.1 Analytical

Three cameras, in particular, are positioned behind the data-acquisition vehicle's glass. The steering angle that the human driver utilised is captured in the time-stamped video from the cameras. The vehicle's Controller Area Network (CAN) bus is used to get this steering command. To make the technology independent of the car's design, they express the vehicle's driving instruction as $1/r$, with r is the turning radii measured in meters. To avoid a singularity when going straight, they use $1/r$ instead of r . While travelling straight, the turning radius is infinite; as $1/r$ approaches 0, left turns (positive values) effortlessly shift to tight turns (positive values). The corresponding steering instruction ($1/r$) consists of a few isolated video stills combined with the training data. 101397 frames with related angular position, torque, and speed descriptors make up the training data set. We then used an 80/20 approach to divide this set of statistics into training and validation. There is also a test set of 5615 frames available. The picture has a 640x480 original resolution.

5 distinct driving films were used to create the training images:

1. 221 seconds, bright sunlight, and several lighting variations. Good twists at the start, errant shoulder lines, a lane merge in the conclusion, and a split highway
2. There are no shoulder lines, the road merges lanes after 791 seconds of a divided highway, there are many shadows, there is a green traffic signal, Direct sunshine, numerous extremely narrow corners where the central camera is unable to see much of the road, and swift elevation changes that provide substantial advantages and disadvantages over the top. Around the 350s, the roadway makes a U-turn and rapidly changes back to 2 lanes.
3. 99 second roundtrip travel over the peak part on a divided highway

4. 212 seconds, a guardrail, and a two-lane road; training may be difficult initially owing to shadows, but everything will return to normal at the end.

5. A split multi-lane motorway with moderate traffic, 371 seconds



Fig 15: Training dataset

3.2 Computational

In the field of computer vision & image processing, the Canny edge detector is likely the most well-known and often used edge detector. Even though it's not really "trivial" to comprehend the Canny edge detector, we'll split down the steps into manageable chunks so we can see what's really going on. Fortunately for us, OpenCV has already included the Canny edge detector for us in the cv2.Canny function due to how frequently it is used in virtually all computer vision applications.

You'll probably find a call to the Canny edge detector inside in the source code of many image processing projects. The Canny edge detector is frequently used as a crucial preprocessing step, whether we are determining the distance between our camera and an item, creating a document scanner, or seeing a Game Boy screen in an image.

It requires following the procedures listed below when spotting edges in an image.

1. Using a Gaussian filter, noise in the input image is removed.
2. Calculating the Gaussian filter's derivative to determine the gradient of an image's pixels and to determine its magnitude along the x and y dimensions.
3. Reduce the non-max edge contributing pixel points by taking into account a group of neighbours for any curve extending in a direction orthogonal to the specified edge.
4. Utilising the Hysteresis Thresholding method, ignore pixels that are less than the low threshold value and maintain those that are more than the gradient magnitude.

3.2.1 Noise Removal or Image Smoothing:

The pixel might not even be close to resembling its neighbours when noise is present. This could lead to the identification of edges being inaccurate or incorrect. The desired edges in the output images are prevented by the Gaussian filter, that's organised with the picture and removes noise, in order to prevent the same.

In the example below, we are convolving an image I with a gaussian filter, or kernel, $g(x,y)$. We employ the matrix $[1 \ 1 \ 1]$ to retain the closeness between pixels and eliminate noise in this case because we want to ensure that any particular pixel must be equivalent to its neighbouring pixels in the output.

$$S = I * g(x, y) = g(x, y) * I$$

$$g(x, y) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Fig 16: I is Input Image and $g(x,y)$ is Gaussian Distribution

To get the magnitude of the gradient along the dimensions, compute the filter's derivative with respect to the X and Y axes and combine it with I . The tangential component of the

angle formed by the two dimensions can also be used to determine the image's direction.

$$\nabla S = \nabla(g * I) = (\nabla g) * I$$

$$\nabla S = \begin{bmatrix} g_x \\ g_y \end{bmatrix} * I = \begin{bmatrix} g_x * I \\ g_y * I \end{bmatrix}$$

$$\nabla g = \begin{bmatrix} \frac{\partial g}{\partial x} \\ \frac{\partial g}{\partial y} \end{bmatrix} = \begin{bmatrix} g_x \\ g_y \end{bmatrix}$$

Fig 17: convolution results in a gradient vector that has magnitude and direction.

Here is an illustration of how Gaussian Derivatives contribute to the edges in the final images.

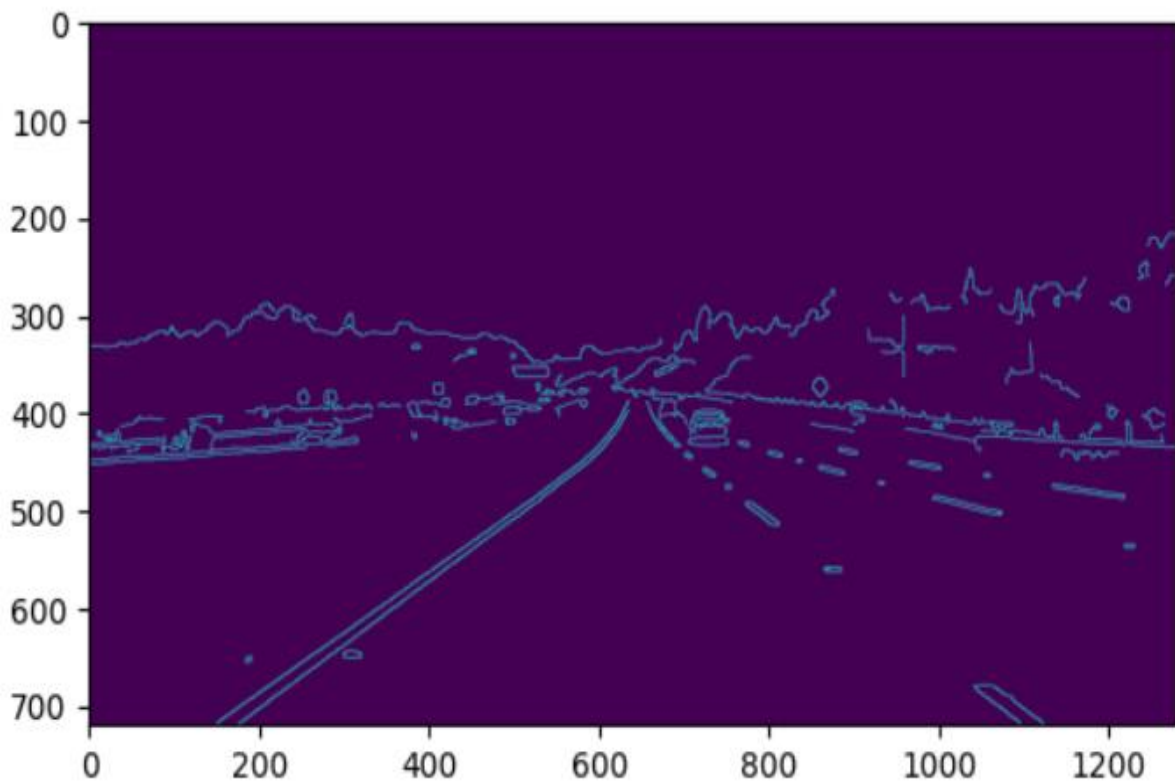
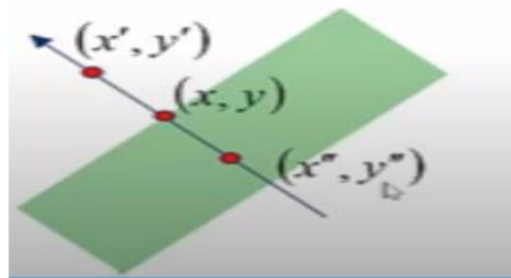


Fig 18: Edge Detection

3.2.2 Non-Max Suppression

It is typically noticed that only a few spots along an edge increase edge visibility. Therefore, we can disregard edge locations that don't significantly increase feature visibility. We employ the Non Maximum Suppression approach to do the same. Here, we indicate the locations along the edge's curve wherever the magnitude is greatest. This can be discovered by searching for a maximum and a slice that is perpendicular to the curve.

Take a look at the edge in the figure below, which contains three edge points. Assume that point (x,y) has the highest gradient of an edge. Look for edge points that are parallel to the edges and check to see if the slope is lower than (x,y) .



$$M(x, y) = \begin{cases} |\nabla S|(x, y) & \text{if } |\nabla S|(x, y) > |\Delta S|(x', y') \\ & \& |\Delta S|(x, y) > |\Delta S|(x'', y'') \\ 0 & \text{otherwise} \end{cases}$$

Fig 19: Non-Maxima Points along the curve

3.2.3 Hysteresis Thresholding

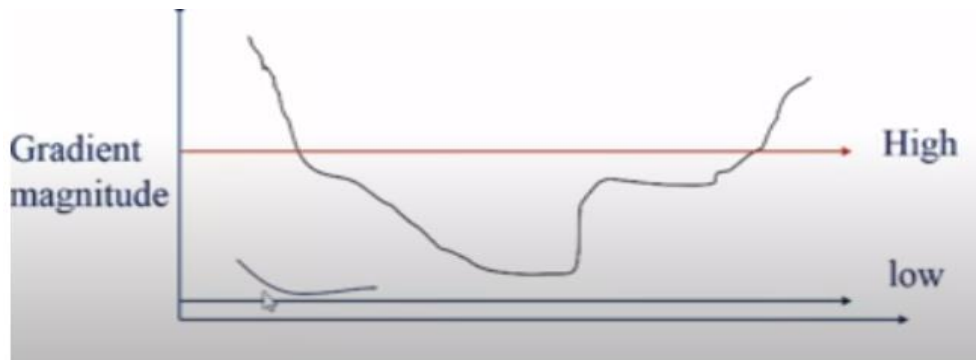


Fig 20: Hysteresis Thresholding

If a pixel's gradient is as follows:

- Identify it as an "edge pixel" above "High."
- "Low" describes it as a "non-edge pixel" below.
- In the range of "low" and "high."

If a pixel is connected to another "edge pixel" or through pixels between "low" and "high," it is first declared a "edge pixel" by repeatedly taking into account its neighbours.

A feature extraction technique used in the analysis of images is the **Hough transform**. Any regular curve, such as lines, circles, ellipses, etc., can have its features isolated using the Hough transform. In its most basic version, the Hough transform can be applied to find lines that are straight in an image. Applications where a straightforward analytic representation of features is impossible can benefit from the use of a generalised Hough transform. People typically avoid utilising the technique due to its computational difficulty. Additionally, learning-based algorithms are capable of extracting intricate elements from a picture or footage that are better suited to the issue at hand.

The most basic boundary identified in the image is a straight line. A border can be far more complicated than just a single straight line. The picture space is converted into a rough space. This changes a line in picture space into a point in high-dimensional space.

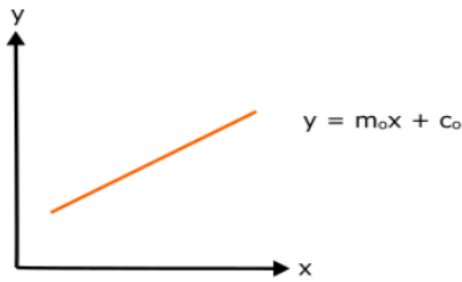


Fig 21: Image Space

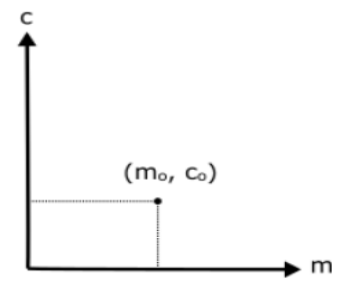


Fig 22: Hough Space

The equation for the line in the image space is $y = mx + c$, m stands for the slope and c for the y-intercept. A point of the form (m, c) will be created from this line in the Hughes space. But for vertical lines in this representation, m is infinite. Let's instead use polar coordinates.

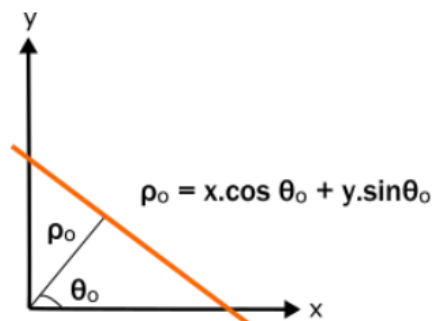


Fig 23: Image Space

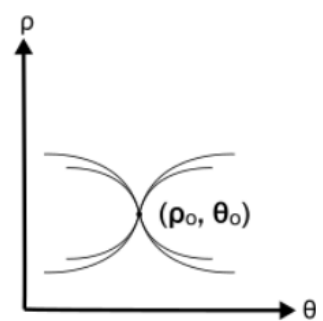


Fig 24: Hough Space

The length of the segment ρ and the angle θ it creates with the x-axis serve as representations for the line. The line in question will be changed into a point through hough space with the form (ρ, θ) .

The parameter space is represented by a histogram array created by the Hough transform, which is a $M \times N$ matrix with M different radius ρ values and N different angle values θ . The total amount of non-zero pixels from the image being input that would be near to the corresponding line is then determined for every parameter combination, and, and the array is properly increased at position (ρ, θ) .

3.2.4 Intuition for line detection

The intersection of several lines in picture space corresponds to the intersection of several points in three-dimensional space.

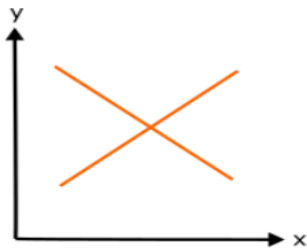


Fig 25: Image Space

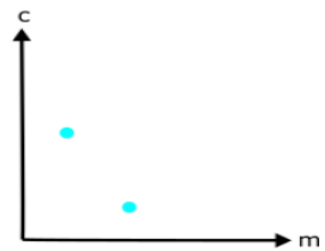


Fig 26: Hough Space

Lines crossing at a point (m, c) in hough spaces can be translated to the line $y = mx + c$ in image space in the same way.

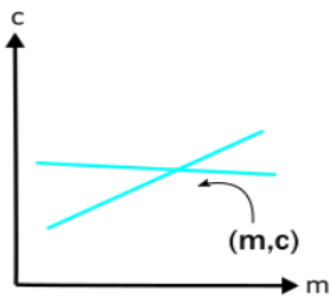


Fig 27: Hough Space

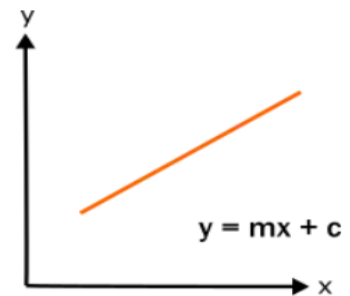


Fig 28: Image Space

When a line in the picture space is composed of numerous segments or points that are near the same line equation, the result is a large number of intersecting lines in the high-dimensional space.

So let's take a look at a line in the picture space that has a few small discontinuities and is an edge detected line. We may convert this discontinuous line from image space to hough space and search for intersecting points in hough space to identify the continuous line in an image. The continuous line in image space will be represented by this intersection point in rough space.

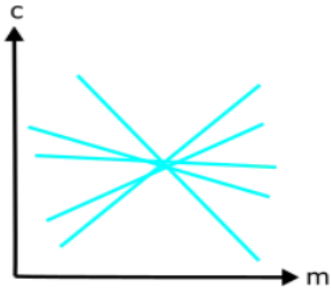


Fig 29: Hough Space

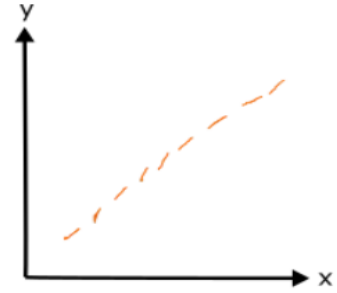


Fig 30: Image Space

3.3 Mathematical

Two variables can be used to express a line in image space. As an example,:

Cartesian coordinate system: Parameters: (m,b)

Polar coordinate system: Parameters: (r,θ)

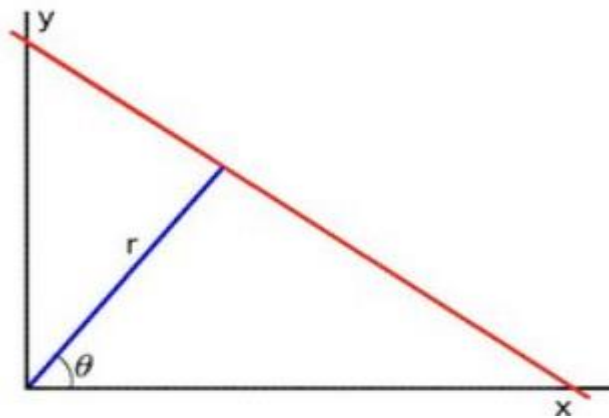


Fig 31: Hough Transformation

$$y = \left(-\frac{\cos \theta}{\sin \theta} \right) x + \left(\frac{r}{\sin \theta} \right)$$

Fig 32: Polar System

Arranging the terms: $r = x\cos\theta + y\sin\theta$

1. In general for each point (x_0, y_0) , The family of lines such passes through that location can be defined as follows:

$$r\theta = x_0 \cdot \cos\theta + y_0 \cdot \sin\theta$$

Meaning that each pair $(r\theta, \theta)$ represents each line that passes by (x_0, y_0) .

2. If for given (x_0, y_0) we plot the family of lines that goes through it, we get a sinusoid. For in-stance, for $x_0 = 8$ and $y_0 = 6$ we get the following plot (in a plane $\theta - r$):

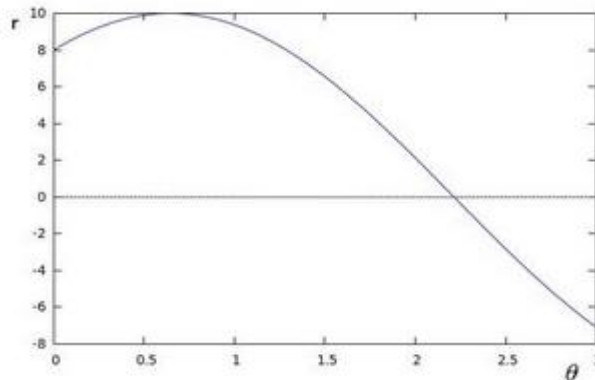


Fig 33: Sinusoidal Representation

We consider only points such that $r > 0$ and $0 < \theta < 2\pi$.

3. We can also repeat the previous technique for all of the points that exist in that particular image. If the curves of two different points intersect in the plane $(\theta - r)$, That is, both points are on the same line. For instance, following with the example above and drawing the plot for two more points: $x_1=4, y_1=9$ and $x_2=12, y_2=3$, we get:

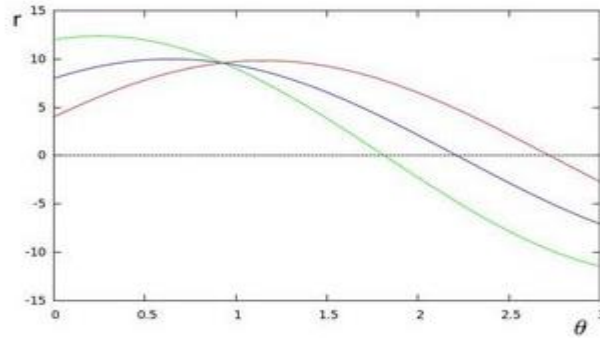


Fig 34: Sinusoidal Representation for different values of x and y

The three plots intersect in 1 single point (0.925,9.6), these coordinates are the parameters (θ, r) or the line on which (x_0, y_0) , (x_1, y_1) and (x_2, y_2) lay respectively.

4. What does all the above statements or calculations mean? It means that in general, a line can be detected by locating the number of intersections between curves. The more curve intersecting means that the line represented by that intersection have more points. In general, we can define a threshold of the min. number of intersections that are needed to detect a line.

5. The Hough Line Transform performs this. It maintains note of where the curves of each point in the picture cross. If the amount of intersections exceeds a certain threshold, it is declared as a line with the intersection point parameters (r). Standard and Probabilistic Hough Line Transform

OpenCV implementations has two kind of Hough Line Transforms:

a. The Standard Hough Transform

- It basically comprises of what we just stated in the previous part. It gives you as result a vector of couples (θ, r)
 - It is implemented in OpenCV by the function `HoughLine()`.
- b. The Probabilistic Hough Line Transform

b. Probabilistic Hough Transformation

- A more efficient Hough Line Transform implementation. It gives as output the extremity of the detected lines (x0,y0,x1,y1)
- **HoughLinesP()** is used in OpenCV library.

3.4 Experimental

We must convert our picture into grey scale because the Canny Edge detector requires such images. Red, Green, and Blue pixels are being combined into one channel with a pixel's value that is between [0,255].

```
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt
```

After importing libraries, we upload the image.

```
from google.colab.patches import cv2_imshow
from google.colab import files
files = files.upload()
```

Converting the uploaded image into grey scale using these lines of code:

```
img = cv.imread("um_000063.png")
gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
plt.imshow(gray)
plt.show()
```

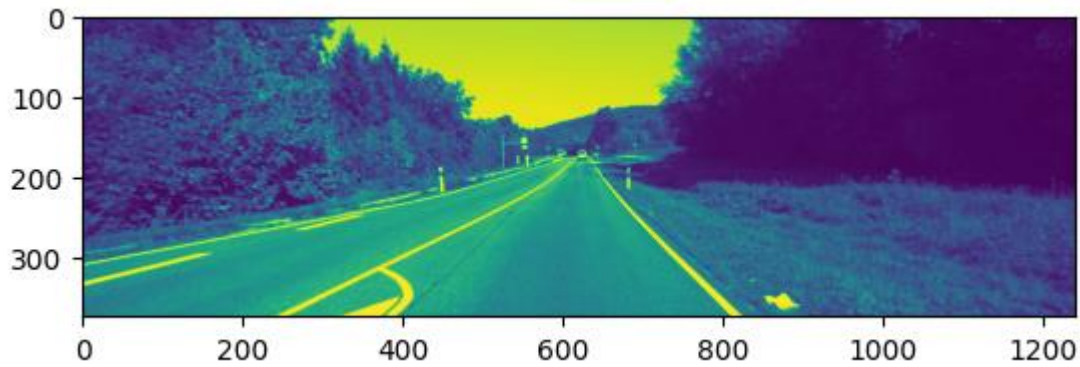


Fig 35: Preprocessed image (Grey scale)

Adding a Gaussian blur to our grayscale image is optional because the clever detector will take care of this step for us.

```
blur = cv.GaussianBlur(gray, (10, 10), 0)
plt.imshow(blur, cmap="grey")
plt.title("GaussianBlurr"), plt.xticks([]), plt.yticks([])
plt.show()
```



Fig 36: Gaussian Blur

The gaussian filter's goal is to minimise image noise. We do this as Canny's gradients are extremely noise-sensitive, so we need to get rid of as much noise as we can.

```
cv.GaussianBlur(image, ksize, sigma)
```

1. image – Image that is to be processed

2. `ksize` - dimension of the kernel which we convolute over the image.
3. `Sigma` - defines the standard deviation along x axis.

The basic notion is to identify edges by detecting abrupt changes in luminance, such as a transition from white to black or from black to white. The parameters are 3.

- The `img` option specifies the image on which we will identify edges.
- The `threshold-1` variable filters out all gradients that are less than this value (they are not considered edges).
- The `threshold-2` parameter specifies the value at which an edge is considered valid.
- Any gradient in between the two thresholds will be considered if it is attached to another gradient who is above *threshold-2*.

```
edges = cv.Canny(blur, 50, 150)
plt.imshow(edges)
plt.show()
```

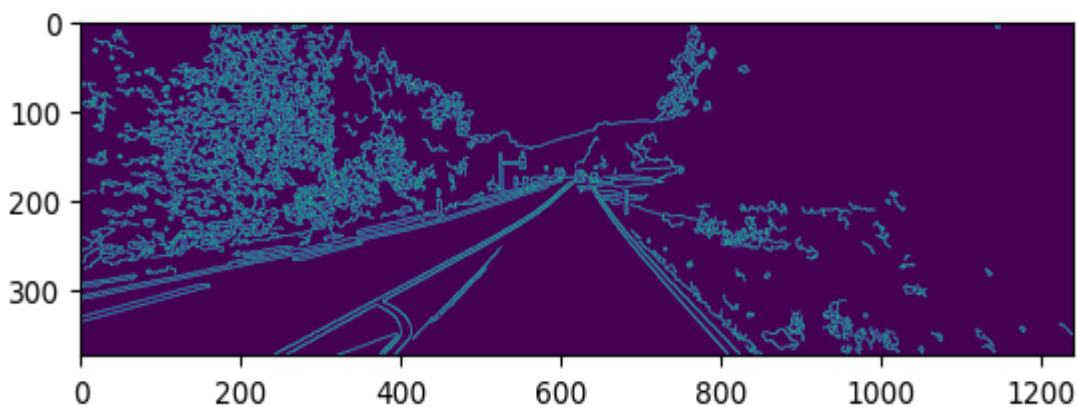


Fig 37: Canny edges image output

```
lines = cv.HoughLinesP(masked_edges, rho=6, theta=np.pi/60, threshold=160,
minLineLength=40, maxLineGap=25)
```

The algorithm's main logic is contained in just one line of code. The component that converts those collections of white pixels from our remote area into actual lines is known

as the Hough Transform.

Parameter 1:

Isolated gradients

Parameter 2 and 3:

Defining the size of the bin, 2 is the value for rho and $\text{np.pi}/180$ theta's value

Parameter 4:

Minimum amount of intersections needed per bin to be considered a line (here its 100 intersections)

Parameter 5: Array's Placeholder

Parameter 6: Min. Line length

Parameter 7: Max. Line gap

```
line_image = np.zeros_like(img)
```

```
for line in lines:
```

```
    x1, y1, x2, y2 = line[0]
```

```
    cv.line(line_image, (x1, y1), (x2, y2), (0, 255, 0), 5)
```

```
plt.imshow(line_image)
```

```
plt.show()
```

The lines generated by the `cv2.HoughLinesP` function are averaged out using this function. Two solid lines having the mean slope & y-intercept of every line segment positioned to the left and right, respectively, will be the outcome.

Each line segment has two coordinates in the final result of the `cv2.HoughLinesP` function: one indicates the line's beginning, and the other its conclusion. We will determine the slopes and the y intercepts of every line segment using these coordinates.

CHAPTER 4

EXPERIMENT AND RESULT

DIGITAL IMAGE PROCESSING TOOLS

4.1 GOOGLE COLABORATORY

A product of Google Research is Colaboratory, or "Colab" for short. Colab is especially well-suited to machine learning, data analysis, and education. It enables anyone to create and execute arbitrary Python code using the browser. Technically speaking, Colab is a service for hosted Jupyter notebooks that offers free access to computer resources, including GPUs, and requires no initial setup to use and is absolutely free to use.

The resources available to Colab are neither limitless nor assured, and the consumption limits occasionally shift. This is required for Colab to provide resources without cost. Colab prioritises its resources for active use cases. Actions related to mass computing, those that have a harmful effect on others, and those that circumvent our policies are all prohibited. The following are disallowed from Colab runtimes:

- file hosting, media serving, or other web service offerings not related to interactive compute with Colab
- downloading torrents or engaging in peer-to-peer file-sharing
- remote control such as SSH shells, remote desktops, remote UIs
- connecting to remote proxies
- mining crypto currency
- running denial-of-service attacks
- password cracking
- using multiple accounts to work around access or resource usage restrictions
- creating deep fakes

Colab notebooks can be loaded through GitHub or stored in Google Drive. Similar to how

you would share Google Docs or Sheets, Colab notebooks can also be shared.

A virtual machine that is exclusive to your account runs code. Virtual machines have the maximum lifetime imposed according to the Colab service and are removed after being inactive for a while. Colab places a high premium on interactive computing. Inactivity will cause runtimes to expire.

The no-cost edition of Colab's notebooks can run free up to 12 hours based on supply and usage patterns. Colab Pro, Pro+, and Pay As You Go provide you more compute availability based on the remaining amount of your compute units. According to availability and usage patterns, notebooks usually have an optimal operating time of twelve hours. Backend termination is what you may anticipate if you use up all of your allotted compute sessions on a Pro, Pro+, or Pay As You Go plan.

If you possess enough processing power, Colab Pro+ provides continuous code execution for a maximum of 24 hours. Idle timeouts are only effective following code execution.

You can use Colab's free edition to access virtual machines with a typical system memory profile. You may use computers with a large memory system profile in Colab's paid editions, subject to capacity and your computing unit level. The term "memory" refers to the computer's memory. The memory profile is the same across all GPU chips.

When you are through working in Colab, think about closing your open tabs. Try to avoid using extra memory or GPUs unless they are absolutely necessary. You will be less likely to encounter use caps within Colab as a result of this. If you reach your limits, you can always use Pay As You Go to buy more compute.

4.2 DIGITAL IMAGE PROCESSING

The practise of applying different techniques to the image in order to enhance it or extract useful information from it is known as image processing. A photo is used as the input in this particular type of signal processing, and the output may include another picture or features or properties associated with the original image. One of the innovations that is

evolving swiftly is image processing. It is a focus of research in the domains of engineering and information technology.

Basically, image processing involves these three steps:

- using image acquisition tools for image importation;
- analysing and editing the imported image;
- output, the output that can be a summary based on an image analysis or a changed image.

Image processing techniques are classified into two types: conventional and digital. Analogue image processing can be used on hard copies such as printouts and photographs. Image analysts use various interpretational fundamentals while using these visual techniques. The usage of digital image processing technologies allows for computer-based digital image editing. All types of data must go through three general phases when using digital techniques: pre-processing, augmentation, and presentation, and information extraction.

4.2.1 Sampling and quantization

An image function $f(x,y)$ has to be digitised both spatial and in amplitude to be appropriate for digital processing. The analogue video stream is typically sampled and quantized using an image grabber or digitizer. Therefore, we must convert continuous data into digital format in order to build a digital image. These require two steps to complete it:

- Sampling
- Quantization

The number of grey levels in the digitised image is determined by the quantization level, while the sampling rate defines the spatial resolution of the digitised image. In image processing, the size of the sampled image is represented as a digital value. Quantization is the process of converting an image function's continuous values into their digital counterparts.

For humans to be able to perceive the fine shading features in the image, the quantization level count should be high enough. The fundamental issue with an image that is being quantized with inadequate brightness levels is the appearance of spurious outlines.

Sampling	Quantization
Digitization of co-ordinate values	Digitization of amplitude values.
x-axis(time) – discretized and y-axis(amplitude) – continuous.	x-axis(time) – continuous and y-axis(amplitude) – discretized.
Before to the quantization procedure, sampling is carried out.	Following the sample procedure is quantification.
It establishes the digitised images' spatial resolution.	It establishes how many grey levels there are in the digitised photos.
It reduces c.c. to a series of tent poles over a time.	It turns c.c. into a never-ending flight of stairs.

Table 2

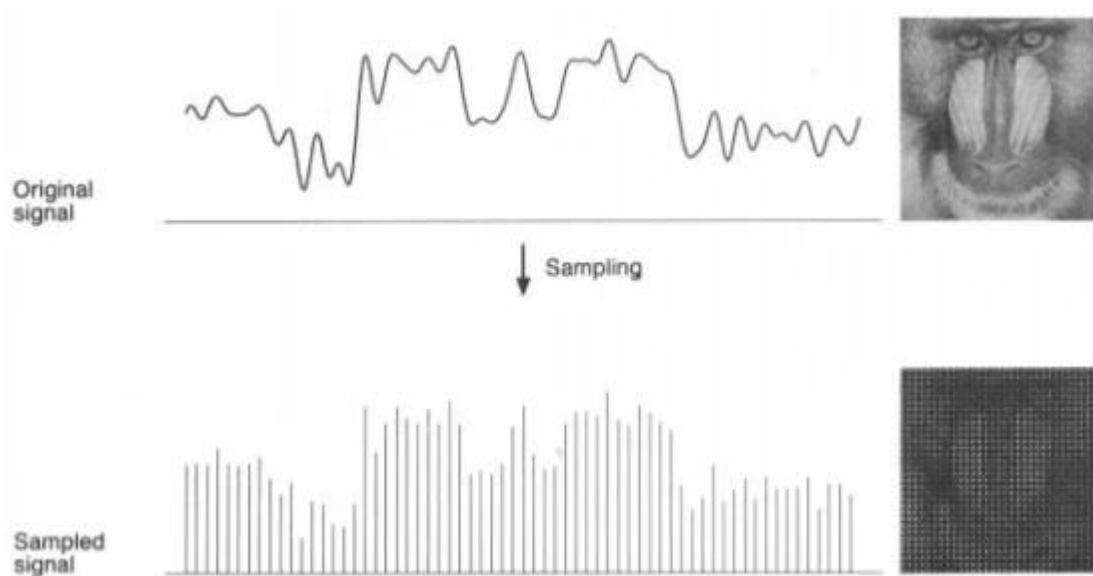


Fig 38: Sampling of an Image

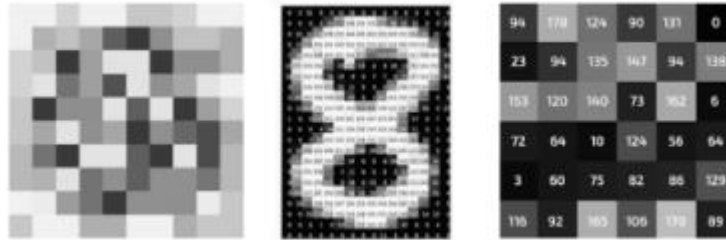


Fig 39: Quantization of an image of level 256

4.2.2. Resizing the Image

picture interpolation is the process of scaling or distorting a picture from one pixel grid to another. Remapping is possible when correcting for lens distortion or rotating an image, however image scaling is necessary when changing the total number of pixels. Zooming is the process that increases the amount of pixels in an image in order to see more detail.

Interpolation makes use of known data for approximating values at unknowable points. By using the numerical values of nearby pixels as a base, image interpolation aims to get the most accurate estimate of a pixel's intensity. It has two directions of operation. Adaptive and non-adaptive interpolation techniques are the two main categories.

4.3 PYTHON FOR DIP

Many libraries are available in Python for image processing, like

- **OpenCV** is a real-time computer vision-focused image processing library that finds utility in a variety of fields, including object identification, mobile robots, 2D and 3D features toolkits, facial and gesture recognition, and human-computer interaction.
- For processing and manipulating images, use the **Numpy** and **Scipy** libraries.
- Numerous image processing algorithms are offered by **Scikit**.

- **Python Imaging Library (PIL)** – To carry out fundamental operations on images, such as thumbnail creation, resizing, rotation, and file format conversion.

A 2D operation A picture can be represented by $F(x,y)$, where x and y are spatial coordinates. The peak to trough of F at that specific value of x,y is used to determine the brightness of an image at a given location. If the x , y , and amplitude values are all finite, we refer to the object as a digital image. In an array, pixels are arranged in rows and columns. Pixels are the elements of an image that contain data about colour and intensity. In the three-dimensional form of images, X , Y , and Z are changed into spatial coordinates. Pixels are arranged in a matrix-like pattern. An RGB picture is what is meant by the term.

Images come in a variety of forms:

- Red, Green, and Blue bands make up the three layers of this two-dimensional RGB image.
- Images in the grayscale format only have one channel and various degrees of black and white.

Classic Image Processing algorithms include:

4.3.1 Morphological Image Processing

Because binary regions created by straightforward thresholding can be damaged by noise, morphological image processing attempts to clean up the flaws in the binary images. Utilising opening and closing processes, it also aids in bringing the image into focus.

Grayscale pictures can be used as an extension for morphological operations. It includes non-linear processes that are connected to the organisation of an image's characteristics. It depends not only on the numerical values of the pixels but also on their related ordering. This method compares the related neighbourhood pixels with a small template called a structuring element that is placed in various potential positions throughout the image.

4.3.2 Gaussian Image Processing

The outcome of blurring a picture with a Gaussian function is gaussian blur, commonly referred to as gaussian smoothing. It is employed to lessen details and visual noise. This kind of blurring creates a similar visual impression to seeing a picture through a translucent screen. It can be used as a way to augment data in deep learning or for image improvement at various scales in computer vision.

4.3.3 Fourier Transform in image processing

An image is broken down into cosine and sine components using the Fourier transform. It can be used for many different things, including image filtering, image compression, and image reconstruction. We shall consider the discrete Fourier transform as we are discussing images.

Let's think about a sinusoid, which consists of three elements:

- Magnitude and contrast-related terms
- brightness-related spatial frequency
- Phase: connected to information about colour

4.3.4 Edge Detection in image processing

Edge detection is a method of image processing that locates the edges of objects in pictures. It operates by looking for changes in brightness. Since the majority of the shape's information is included in the edges, this could be quite helpful in obtaining valuable information from the image. Traditional edge detection techniques find brightness discontinuities. When detecting the changes in grey levels in a picture, it can react quickly if a certain amount of noise is found. Edges are referred to as the local gradient maxima.

4.4 IMAGE PROCESSING LIBRARIES

4.4.1 OpenCV

Open Source Computer Vision Library is what it stands for. This collection contains more

than 2000 optimised algorithms that can be used for machine learning and computer vision. Image processing techniques uses OpenCV in a variety of ways, some of which are given below:

- Converting images between colour spaces, such as between BGR and grayscale, BGR and HSV, etc.
- Applying thresholding techniques to picture data, such as basic thresholding and adaptive thresholding.
- Blurring and using custom filters to photos are examples of image smoothing.
- Morphological manipulations of pictures.
- Building pyramidal images.
- Utilising the GrabCut algorithm to extract foreground from photos.
- Watershed algorithm image segmentation.

4.4.2 Scikit-image

It is an image preparation library that is open-source. With just a few built-in functions, it can execute complicated manipulations on images using machine learning.

Even for individuals who are brand-new to Python, this module is pretty straightforward and works with numpy arrays. Among the operations that scikit image may perform are:

- Use the `try_all_threshold()` method on the picture to implement thresholding operations. Seven global thresholding techniques will be used. The `filters` module contains this.
- Utilise the `sobel()` technique within the `filters` module to accomplish edge detection. We must first convert an image to grayscale because this method demands a 2D grayscale picture as an input.
- Use the `filters` module's `gaussian()` function to achieve gaussian smoothing.
- Use the `exposure` module to apply histogram equalisation, the `equalize_hist()` method to apply conventional histogram equalisation onto the original image, and the `equalize_adapthist()` method to apply adaptive equalisation.
- Use the `rotate()` function found in the `transform` module to rotate an image.
- Use the `rescale()` method in the `transform` module to rescale the image.

- Use the `binary_erosion()` and `binary_dilation()` functions in the morphology module to perform morphological operations.

4.4.3 NumPy

You may also use this library to carry out basic picture operations like flipping, feature extraction, and analysis.

Numpy multi-dimensional arrays can be used to represent images, hence their type is `NdArrays`. A three-dimensional numpy array is a colour image. The RGB channels of the multidimensional array can be divided.

The image can be subjected to the following operations using NumPy (the image is loaded into a variable called `test_img` using `imread`).

- Use `np.flipud(test_img)` to flip the image vertically.
- Use `np.fliplr(test_img)` to flip the picture in a horizontal manner.
- Use `test_img[::-1]` to flip the image (the image is named `img_name` after being stored as a numpy array).

4.5 RESULTS

The following lines of code provided the desired output of this project:

- Importing the python libraries on google colab


```
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt
```
- Uploading the Image:


```
from google.colab.patches import cv2_imshow
from google.colab import files
files = files.upload()
```
- A png image file with name `um_000063.png` is uploaded and preprocessed by converting it into grayscale for image processing


```
img = cv.imread("um_000063.png")
```

```

gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
plt.imshow(gray)
plt.show()

```

- Gaussian Blur

```

blur = cv.GaussianBlur(gray, (5, 5), 0)
plt.imshow(blur, cmap="gray")
plt.title("GaussianBlurr"), plt.xticks([]), plt.yticks([])
plt.show()

```

- Canny Edge

```

edges = cv.Canny(blur, 50, 150)
plt.imshow(edges)
plt.show()

```

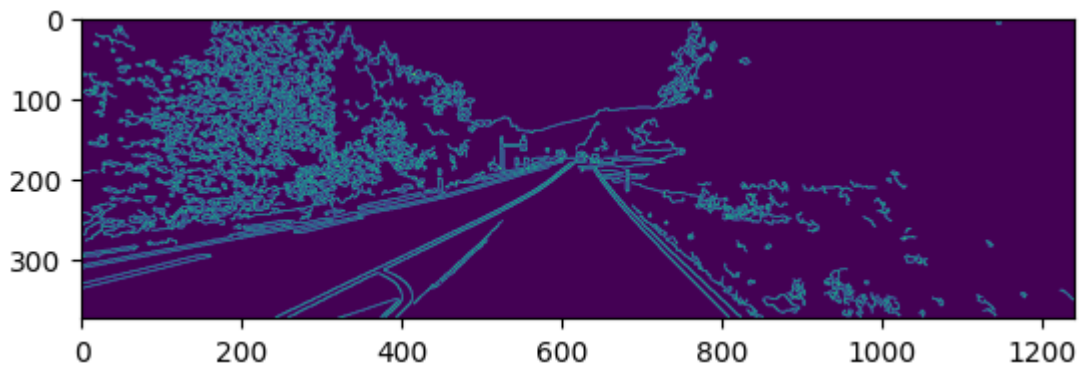


Fig 40: Canny Edge detection output

- Region of Interest

```

mask = np.zeros_like(edges)
height, width = img.shape[:2]
roi_vertices = [(0, height), (width/2, height/2), (width, height)]
mask_color = 255
cv.fillPoly(mask, np.array([roi_vertices], dtype=np.int32), mask_color)
masked_edges = cv.bitwise_and(edges, mask)

```

- Hough Transformation

```

lines = cv.HoughLinesP(masked_edges, rho=6, theta=np.pi/60, threshold=160,
minLineLength=40, maxLineGap=25)

```


- Drawing Lines

```
line_image = np.zeros_like(img)
for line in lines:
    x1, y1, x2, y2 = line[0]
    cv.line(line_image, (x1, y1), (x2, y2), (0, 255, 0), 5)
plt.imshow(line_image)
plt.show()
```

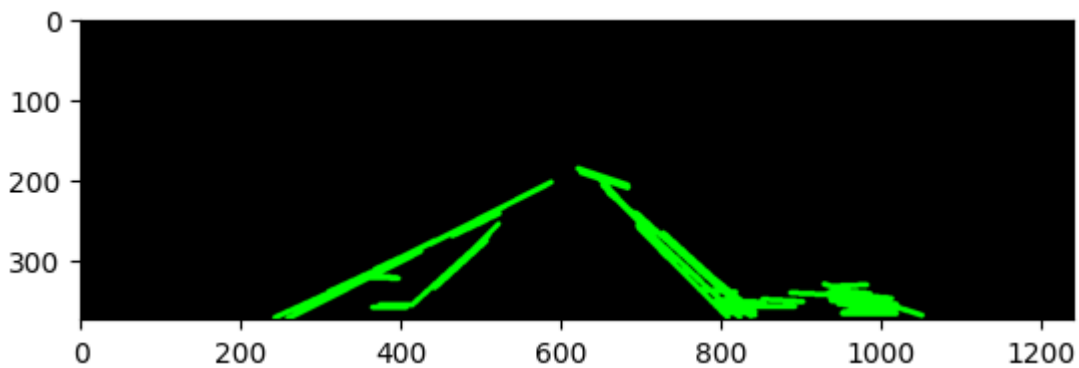


Fig 41: Lines created by Canny Edge Detection and Hough Transformation

- Overlaying lines on the original image and the display the final output

```
final_image = cv.addWeighted(img, 0.8, line_image, 1, 0)
plt.imshow(final_image)
plt.show()
```

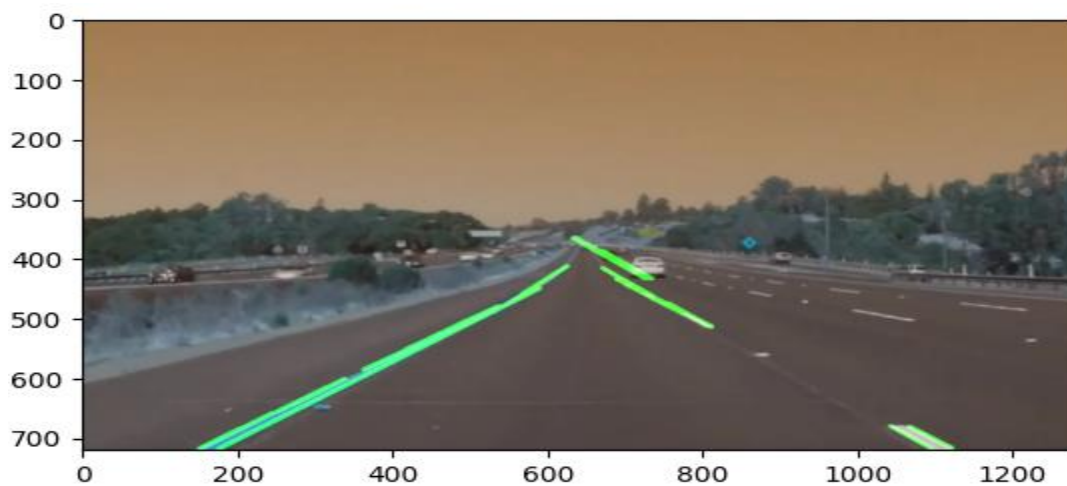


Fig 42: Final Output/Result

The Sobel operator's key benefits are that it is less complicated and takes less time.

The edges, though, are jagged. On the other side, non-maxima suppressing and thresholding are used in the Canny techniques, which results in edges that are more smooth. The Canny algorithm has the drawback of being more intricate and time-consuming than Sobel. Having an understanding of these variations might help you choose the best strategy or method for a given application.



Left: input image. middle: Sobel . right: Canny

Fig 43: Comparison of Sobel and Canny Edge Detection Methods

V

CONCLUSION

When we are driving, we make decisions based on our vision. Our constant point of reference concerning where to direct the car is the lines on the roadway that the model has identified as the lane markings. Additionally, this steering is automatic. Naturally, automatically detecting lane lines with an algorithm is one of the initial tasks we would like to achieve when creating a self-driving car. The region of interest (ROI) for road detection must be adaptable.

The horizon will shift when moving upward or downward along a steep incline and will no longer be determined by the frame's proportions. Additionally, this is something to take into account when there are tight corners and heavy traffic.

In-depth research, designing, and planning resulted in the development of a lane detection system that autonomous driving systems can use to identify lanes, ensuring safer travel for the passengers in the car as well as for pedestrians and other vehicles in the area. The system preprocesses the image frame using the clever edge detection method before applying the Hough transform algorithm to highlight the lanes for the user's convenience.

This method's key benefit is that it can anticipate the lane configurations on the road without the need for a trained model. Instead, it actively receives the video frames as input and actively recognizes the frame boundaries, which are subsequently returned to the user as lanes. As a result, no effort is wasted creating, training, and testing a dataset. With the help of this technology, users can swiftly integrate lane detection into their autonomous driving systems.

Highway cameras can also use this technique to identify and ticket vehicles who are driving outside of their lanes, endangering other drivers on the same route.

The standardization of associated image processing technology shall be done in

accordance with how artificial intelligence is now advancing the field of algorithm research in digital image processing. Image enhancing methods based on deep convolution neural networks are being gradually proposed, influenced by the advancement of artificial intelligence and deep learning. This kind of approach can provide the speed and accuracy requirements for digital image processing. This further demonstrates the development of intelligence in underwater image processing.

It is possible to perform specific operations on an image in order to improve it or extract valuable information from it. It is a kind of signal processing where a picture serves as the input and the output can either be another image or attributes related to the input image. One of the technologies with the quickest growth rate nowadays is image processing. It also opens up a significant area for research in computer science and engineering.

However, ease and miniaturization are also challenges which digital image processing techniques must solve in order to expand the audience.

In summary, the technology for processing images has a significant social impact and is employed in almost every aspect of people's lives. Digital image processing continues to have a lot of new areas to investigate and will continue to advance and develop in a more positive manner as long as human requirements continue to expand

REFERENCES

- [1] Tullimalli Sarsha Sree and Sandeep Kumar Sathapathy, Volume: 07 Issue: 06 | June 2020, 'Lane Line Detection using Hough Transform and Convolutional neural Networks'
- [2] Nidhi Lakhani, Ritika Karande, Deep Manek, Vivek Ramakrishnan, Volume: 09 Issue: 04 | Apr 2022, 'Lane Detection using Image Processing in Python'
- [3] Raja Muthalagu, Anudeep Sekhar Bolimera, Dhruv Duseja, Shaun Fernandes, Volume: 22, no.4, 2021 , 'Object and Lane Detection Techniques for Autonomous Car using Machine Learning approaches'
- [4] Vighnesh Devane, Ganesh Sahane, Hritish Khairmode, Gaurav Datkhile, ITM Web of Conferences 40, 03011 (2021), 'Lane Detection Techniques using Image Processing'
- [5] Jamel Baili, Mehrez Marzougui, Ameer Sboui, Samer Lahouar, Mounir Hergli, 2017 Second International Conference on Anti-Cyber Crimes (ICACC), 'Lane Departure Detection using Image Processing Techniques'
- [6] Yan Liu , Jingwen Wang , Yujie Li , Canlin Li, Accepted: 28 April 2022 | Published: 30th April 2022 'Lane-GAN: A Robust Lane Detection Network for Driver Assistance System in High Speed and Complex Road Conditions'
- [7] Nushaine Ferdinand, May 5, 2020, 'A Deep Dive into Lane Detection with Hough Transform'
- [8] Zequn Qin, Huanyu Wang, and Xi Li, August 5, 2020, 'Ultra Fast Structure-aware Deep Lane Detection'
- [9] Ling Ding, Huyin Zhang, Jinsheng Xiao, Cheng Shuang Shejie Lu, CMES, vol.122, no.3, pp.1039-1053, 2020, 'A Lane Detection Method Based on Semantic Segmentation'

- [10] Y. Wang, E. K. Teoh and D. Shen, 22, P 269-280, 2004, 'Lane detection and tracking using B-snake, Image and Vision Computing'
- [11] Qiu, D., Weng, M., Yang, H., Yu, W. and Liu, K., 2019, June. Research on Lane Line Detection Method Based on Improved Hough Transform. In 2019 Chinese Control And Decision Conference (CCDC) (pp. 5686-5690). IEEE.
- [12] Wu, P.C., Chang, C.Y. and Lin, C.H., 2014. Lane-mark extraction for automobiles under complex conditions. *Pattern Recognition*, 47(8), pp.2756-2767.
- [13] de Paula, M.B. and Jung, C.R., 2013, August. Real-time detection and classification of road lane markings. In 2013 XXVI Conference on Graphics, Patterns and Images (pp. 83-90). IEEE.
- [14] Kunz, N. Chase. (2017) Vision-Based Control of a Full-Size Car by Lane Detection. All Graduate Theses and Dissertations. 6534. Utah State University, United States.
- [15] Open-Source Autonomous Driving Dataset. (2017)
- [16] Eskandarian A. (2012) Fundamentals of Driver Assistance. In: Eskandarian A. (eds) Handbook of Intelligent Vehicles. Springer, London.
- [17] Zhao, Z.Q., Zheng, P., Xu, S.T. & Wu, X. (2018) Object detection with deep learning: A review, arXiv e-prints, arXiv:1807.05511.
- [18] Bellis, Elizabeth, & Jim. (2008) National motor vehicle crash causation survey (NMVCCS) .(SAS analytical user's manual. No. HS-811 053.)
- [19] Hernández, D.C., Hoang, V.D. and Jo, K.H., 2013, July. Vanishing point based image segmentation and clustering for omnidirectional images. In the International Conference on Intelligent Computing (pp. 541-550). Springer, Berlin, Heidelberg.