**A MINI PROJECT**

**on**

**LANE-LINE DETECTION SYSTEM IN PYTHON USING OPEN CV**

Submitted to

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY, HYDERABAD

In partial fulfilment of the requirement for the award of the degree of

**BACHELOR OF TECHNOLOGY**

**in**

**ELECTRONICS AND COMMUNICATION ENGINEERING**

**By**

|  |  |
| --- | --- |
| **ANOUSHKA KONDOJ** | **187Y1A04C3** |
|  |  |
|  |  |

**Under the Guidance of**

B.N.SRINIVAS

Associate professor



**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING**

****

**November, 2021**

****

**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING**

Date: 1-11-2021

**CERTIFICATE**

This is to certify that the project work entitled “**LANE-LINE DETECTION SYSTEM IN PYTHON USING OPEN-CV** ” work done by **ANOUSHK KONDOJ(187Y1A04C3)** student of Department of Electronics and Communication Engineering, is a record of bonafide work carried out by the member during a period from July, 2021 to November, 2021 under the supervision of **B.N.SRINIVAS**. This project is done as a fulfilment of obtaining Bachelor of Technology Degree to be awarded by Jawaharlal Nehru Technological University Hyderabad, Hyderabad.

The matter embodied in this project report has not been submitted by me to any other university for the award of any other degree.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **ANOUSHKA KONDOJ** |

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

Date: (B.N.SRINIVAS)

The Viva-Voce Examination of above student, has been held on………………………

|  |  |
| --- | --- |
| Head of the Department | External Examiner |
|  |  |
| Principal | |

**ACKNOWLEDGEMENTS**

I would like to express my sincere gratitude to my guide **B.N.Srinivas**, **Associate Professor**, Department of Electronics and Communication Engineering, for his excellent guidance and invaluable support, which helped me accomplish the B.Tech (ECE) degree and prepared me to achieve more life goals in the future. His total support of my dissertation and countless contributions to my technical and professional development made for a truly enjoyable and fruitful experience. Special thanks are dedicated for the discussions we had on almost every working day during my project period and for reviewing my dissertation.

I am very much grateful to my Project Coordinator**, Dr. G. Amarnath**, **Associate Professor,** Department of ECE, MLRITM, Dundigal, Hyderabad, who has not only shown utmost patience, but was fertile in suggestions, vigilant in directions of error and has been infinitely helpful.

I am extremely grateful to **Dr. Srinivas Bachu**, **Associate Professor & HOD-ECE**, MLRITM, Dundigal, Hyderabad, for the moral support and encouragement given in completing my project work.

I wish to express deepest gratitude and thanks to **Dr. K. Venkateswara Reddy, Principal,** MLRITM, for his constant support and encouragement in providing all the facilities in the college to do the project work.

I would also like to thank all our faculties, administrative staff and management of MLRITM, who helped me to completing the mini project.

On a more personal note, I thank my **beloved parents and friends** for their moral support during the course of our project.

**TABLE OF CONTENTS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | | | **Page No.** |
| *Certificate* | | | | *ii* |
| *Acknowledgements* | | | | *iii* |
| *Table of Contents* | | | | *iv* |
| *List of Figures* | | | | *vi* |
| *List of Abbreviations* | | | | *vii* |
| *Abstract* | | | | *viii* |
| **Chapter 1: Introduction** | | | | 1-13 |
|  | 1.1 |  | Introduction | 1-2 |
|  | 1.2 |  | Overview if the Proposed System | 2-3 |
|  | 1.3 |  | Proposed Methods | 3-5 |
|  | 1.4 |  | Preprocessing | 5-6 |
|  |  | 1.4.1 | Cropped Image | 6 |
|  | 1.5 |  | Thresholding | 7-8 |
|  |  | 1.5.1 | HLS thresholding | 7-8 |
|  | 1.6 |  | Sliding Window Search | 8-9 |
|  | 1.7 |  | Illustrate Lane | 10-11 |
|  | 1.8 |  | Related Works | 11-13 |
| **Chapter 2: Literature Survey** | | | | 14-16 |
|  | 2.1 |  | Introduction | 14 |
|  | 2.2 |  | Stereo Vision | 14 |
|  |  | 2.2.1 | Passive Stereo | 14 |
|  |  | 2.2.2 | Active Stereo | 14-15 |
|  | 2.3 |  | Mapping | 15 |
|  |  | 2.3.1 | Point Clouds | 15 |
|  |  | 2.3.2 | Occupancy Grids | 15-16 |
| **Chapter 3: Methodology** | | | | 17-24 |
|  | 3.1 |  | Preprocessing | 17 |
|  | 3.2 |  | Colour Transform | 17-18 |
|  | 3.3 |  | Basic Preprocessing | 18-19 |
|  | 3.4 |  | Adding Colour Extraction in Preprocessing | 19-20 |
|  | 3.5 |  | Adding Edge Detection in Preprocessing | 20-23 |
|  |  | 3.5.1 | ROI Selection | 21 |
|  |  | 3.5.2 | Lane Detection | 22 |
|  |  | 3.5.3 | Edge Detection | 22-23 |
|  | 3.6 |  | Lane Tracking Using Extended Kalman Filter | 23-24 |
| **Chapter 4: Software tools required** | | | | 25-46 |
|  | 4.1 |  | Tools | 25 |
|  |  | 4.1.1 | Python | 25 |
|  |  | 4.1.2 | Graphical Processing Unit | 25-26 |
|  |  | 4.1.3 | OpenCV | 26 |
|  |  | 4.1.4 | NumPy | 27-28 |
|  |  | 4.1.5 | SciPy | 28-29 |
|  | 4.2 |  | Introduction to CNN | 30-33 |
|  | 4.3 |  | Layers of CNN | 34 |
|  | 4.4 |  | Pooling/subsampling layers | 35 |
|  | 4.5 |  | Non-linear layers | 35-37 |
|  | 4.6 |  | Why CNN? | 37-42 |
|  | 4.7 |  | The Future of CNN | 43 |
|  | 4.8 |  | Comparison of various lane detection and tracking algorithm | 43-46 |
| **Chapter 5: Result and Conclusions** | | | | 47-52 |
| ***References*** | | | | 53 |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **Figure No.** | **Name of the Figure** | **Page No.** |
| Figure 1.2 | Block Diagram | 3 |
| Figure 1.3 | Polar co-ordinates of hough transform | 4 |
| Figure 1.3.1 | Block diagram of preprocessing | 5 |
| Figure 1.4 | Undistorted image | 6 |
| Figure 1.4.1 | Cropped image | 7 |
| Figure 1.5 | Threshold image | 8 |
| Figure 1.6 | Sliding window searched images | 9 |
| Figure 1.7 | Ilustrated lanes | 11 |
| Figure 3.2 | Colour transformed image | 18 |
| Figure 3.3 | Grey scaled, Blurred images | 19 |
| Figure 3.4 | Adding Colour Extraction in preparing | 20 |
| Figure 3.5 | Adding edge detection in preprocessing | 20 |
| Figure 3.5.1 | ROI selection images | 21 |
| Figure 3.5.2 | Edge detected image | 22 |
| Figure 3.5.3 | Lane detected image | 23 |
| Figure 3.6 | Lane Tracking using extended kalman fiter | 24 |
| Figure 4.2.1 | Training of neural network | 30 |
| Figure 4.2.1 | Biological neron versus artificial neuron network | 31 |
| Figure 4.2.3 | Typical block diagram of CNN | 32 |
| Figure 4.2.4 | Vision algorithm pipeline | 33 |
| Figure 4.3 | Pictorial representation of convolution process | 34 |
| Figure 4.4 | Pictorial representation of max pooling and average pooling | 35 |
| Figure 4.5.1 | Pictorial representation of ReLU functionality | 36 |
| Figure 4.5.2 | Processing of a fully connected layer | 37 |
| Figure 4.6.1 | GTSRB ideal traffic signs | 39 |
| Figure 4.6.2 | Performance vs Complexity | 40 |
| Figure 4.8 | The optical flow vector of a moving object in a video sequence | 42 |
| Figure 5.1 | HSV colour conversion | 47 |
| Figure 5.2 | ROI selection | 48 |
| Figure 5.3 | Edge and Lane detection | 49 |
| Figure 5.4 | Hough transform | 50 |
| Figure 5.5 | Canny edge detection | 50 |
| Figure 5.6 | Output | 51 |

**LIST OF ABBREVIATIONS**

|  |  |
| --- | --- |
| EKF | Extended Kalman Filter |
| ACC | Adaptive Cruise Control |
| LDW | Lane Departure Warning |
| DBSCAN | Density based spatial application noise clustering |
| LPF | Low Pass Filter |
| HPF | High Pass Filter |
| MSER | Masters of Science in Economic Research |
| IPM | Integrated Pest Management |
| PVP | Player versus player |
| RNN | Recurrent Neural Network |
| CNN | Cable News Network |
| HLS | Hyper Text Transfer Protocol Live Streaming |

**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 multifeature 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 preprocessing 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 preprocessing stage and then select ROI on the basis of the proposed preprocessing. This new preprocessing 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.

**CHAPTER 1**

**INTRODUCTION**

**1.1 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) [4] 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 [6]. 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 reliably . 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. These errors will affect the subsequent lane 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.

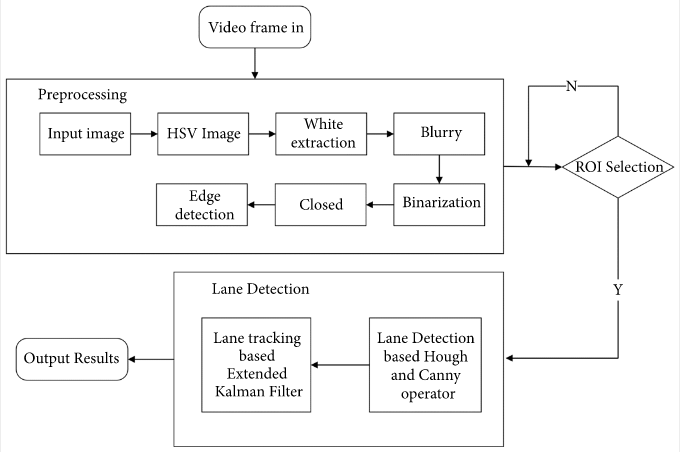
Line detection is as important as edge detection in lane detection. With regard to line detection, we usually have two methods which include feather-based method and model-based methods. Niu et al. used a modified Hough transform to extract segments of the lane profile and used DBSCAN (density based spatial application noise clustering) clustering algorithm for clustering. In 2016, Mammeri et al. used progressive probabilistic Hough transform combined with maximum stable extreme area (MSER) technology to identify and detect lane lines and utilized Kalman filter to achieve continuous tracking. However, the algorithm does not work well at night.

In this paper, 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 preprocessed 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. In this paper, experiments show that the proposed method is significantly better than the existing preprocessing method and ROI selection method in lane detection.

**1.2 Overview of the Proposed System:**

This paper presents an advanced 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.

Figure 1 shows 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 color feature extraction and edge feature extraction. In order to reduce the influence of noise in the process of motion and tracking, after extracting the color 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.



**Figure 1.2 Block diagram**

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 (EKF) to detect and track lane line in real time.

**1.3 Proposed Methods**

In this paper, based on the previous preprocessing, we firstly extract the colour features based on the white colour and then extract the edge features based on the straight lane. Because

the high-speed section is the traffic accident-prone section, the high-speed road section mostly is the straight line lane. Therefore, in order to obtain a very high recognition rate, we successively carry on colour detection and edge detection to the lane. This paper combines colour features extraction and edge features extraction, and the experiment proves that the recognition rate and accuracy of lane detection are greatly improved.

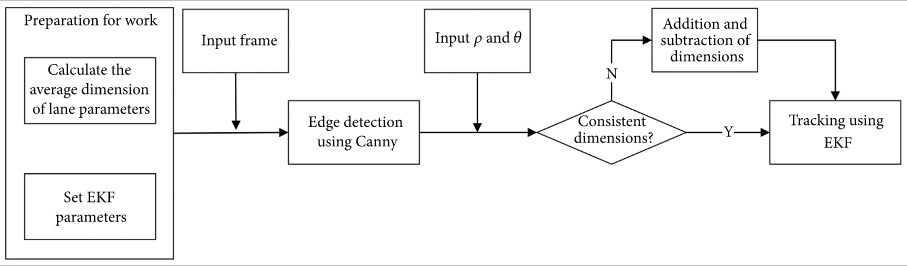
Our main contribution in this paper is to do a lot of work in the preprocessing stage. We proposed to perform colour transform of HSV in the preprocessing stage, then extract white, and then perform conventional preprocessing operations in sequence. Moreover, we selected an improved method proposed in the area of interest (ROI). In this paper, based on the proposed preprocessing method (after HSV colour transform, white feature extraction, and basic preprocessing), one-half part of the processed image is selected as the area of interest (ROI). In addition, we performed twice edge detection. The first is in the preprocessing stage, and the second is in the lane detection stage after the ROI is selected. The purpose of performing twice edge detection is to enhance the lane recognition rate.

In this paper, Hough transform is used for the straight line detection. Figure [2](https://www.hindawi.com/journals/am/2018/8320207/fig2/) shows the basic principles of the Hough transform. In Figure [2](https://www.hindawi.com/journals/am/2018/8320207/fig2/)(a), each point on the straight line crossing the point and the point corresponds to a straight line and a straight line on the parameter space map in Figure [2](https://www.hindawi.com/journals/am/2018/8320207/fig2/)(a) after Hough transformation; two lines intersect at the point , where and are the parameters of the line determined by the point and point . On the contrary, the straight line and the straight line where the parameter space intersects at the same point and the collinear points in Figure [2](https://www.hindawi.com/journals/am/2018/8320207/fig2/)(a) are correspondence . According to this characteristic, given some specific points in Figure [2](https://www.hindawi.com/journals/am/2018/8320207/fig2/)(a), the line equations connecting these points in Figure [2](https://www.hindawi.com/journals/am/2018/8320207/fig2/)(b) can be calculated by Hough transform.



**Figure 1.3 Polar co-ordinates of hough transform**

Different from Figure [2](https://www.hindawi.com/journals/am/2018/8320207/fig2/)(b), when Figure [3](https://www.hindawi.com/journals/am/2018/8320207/fig3/)(b) is expressed in polar coordinates, the collinear point and point mapped to the parameter space in the original image intersect at the point .The Kalman filtering algorithm is used to track lane lines in real time. In this paper, we use Extended Kalman Filter (EKF) to track the lane in real time. After the parameters and of the lane based on the straight line model are obtained from the Hough transforms of Figures [2](https://www.hindawi.com/journals/am/2018/8320207/fig2/) and [3](https://www.hindawi.com/journals/am/2018/8320207/fig3/), the lane line can be tracked using the EKF. The EKF tracking algorithm is described in Table [1,](https://www.hindawi.com/journals/am/2018/8320207/tab1/) the initial value of the parameter and the initial value of the covariance are set as the unit matrix, and the predicted value of the current state is the tracking result of the previous state. The real value of the current state is the sequence frame of the current reading; thus the tracking value of the current state (the optimal estimation result) can be obtained. This value is also used as the prediction value of the next state to realize the cyclic estimation of the lane parameters, that is, the tracking. Table [1](https://www.hindawi.com/journals/am/2018/8320207/tab1/) shows Extended Kalman Filter algorithm module.As shown in Figure [4](https://www.hindawi.com/journals/am/2018/8320207/fig4/), before inputting the image frame, we made preparations such as calculating the average value of vehicle parameter dimensions and setting EKF parameters. The input image frame is detected by the Canny edge operator and the resulting edge image is obtained. Then, we add the parameters and of the lane line based on the straight line model that have been obtained by the Hough transform and determine whether the lane parameters and dimensions detected by the Hough transform are the same for all input frame images. If they are equal, use EKF for lane tracking, or enter the dimension addition and subtraction module to adjust the parameter dimension.

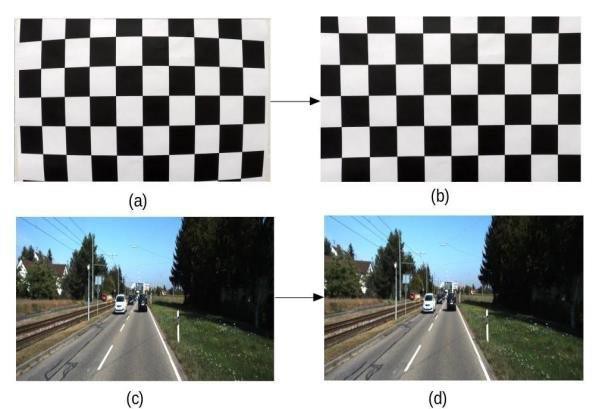


**Figure 1.3.1 Block diagram of preprocessing**

**1.4 Preprocessing**:

Preprocessing stage has a significant characteristic of the lane lines marking. The main purpose of preprocessing is to increase the contrast, eliminate the noise and generate an edge image for the corresponding input image. In this stage, the images are undistorted so that it will restore the straightness of lines, helping to identify lane lines. The variation between the

distorted (original) and undistorted images is clear. The curved lines are now straight. The camera matrix and distortion coefficients using chessboard images are calculated in OpenCV. Here, it can be accomplished by gaining the inside corners within an image and utilizing that information to undistort the image. The foregoing concerns to coordinates in our 2D mapping while the contemporary represents the real-world coordinates of those image points in 3D space (with the z-axis, or depth = 0 for our chessboard images). Those mappings facilitate to attain out how to accurately eliminate distortion on images. The preprocessing steps with input image for the lane detection system are illustrated in Fig. From the Fig. it can be seen that the undistorted image is brighter than the original image as it is noise free.

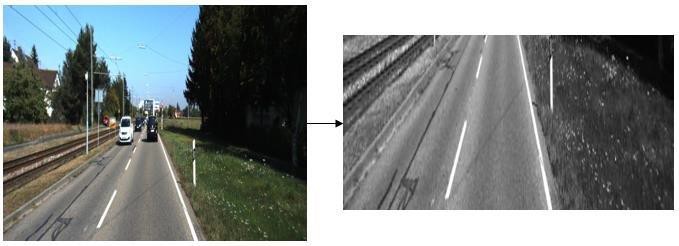


**Figure 1.4 Undistorted image**

**1.4.1 Cropped Image**

Cropping is a procedure that is used to eliminate the unwanted region from a particular image. During the determination of lane lines, we only need to focus on the regions where we are likely to see the lanes. For this reason, the cropping operation is done and performing the further image processing only in the particular areas of the image. The cropped image to focus the particular region of lanes is illustrated in Fig. 3. We have resized the image to smaller dimensions. This supports with doing the image processing pipeline faster.

**Figure 1.4.1 Cropped image**



**1.5 Thresholding**:

Thresholding is generally used for image segmentation. This method is a kind of image segmentation that separates objects by altering grayscale images into binary images. Image thresholding technique is the most appropriate in images with high stages of contrast.

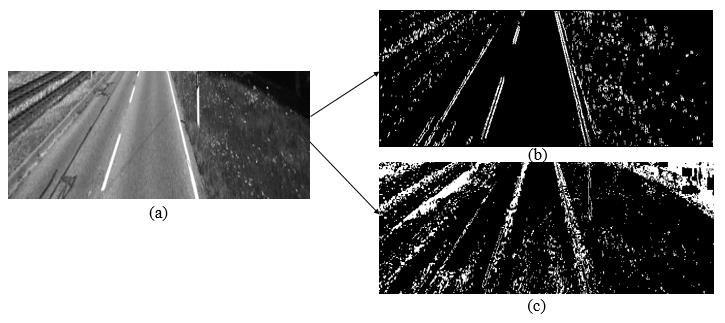
**1.5.1 Gradient Thresholding:**

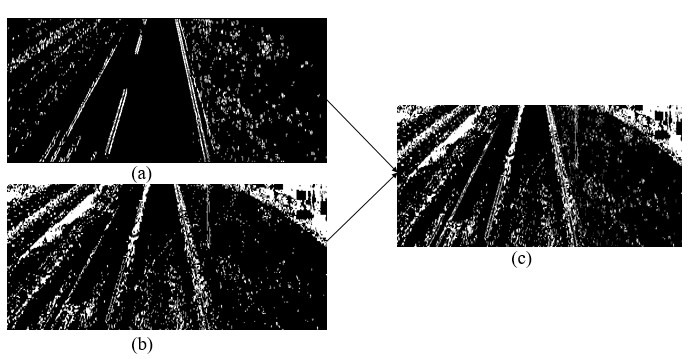
Sobel is a kernel for gradient thresholding in both and x and y-axis. Since the lane lines are probably going to be vertical, the more weight on the inclination in a y-axis is given. For proper scaling, total slope esteems and standardized is taken into consideration. The outcome after applying gradient thresholding is illustrated in Fig. 4 (b).

**1.5.2 HLS thresholding:**

(Hue Saturation Lightness) color channel is used to handle cases when the road color is too bright or too light. L (lightness) channel threshold diminishes edges formed from shadows in the frame. S (saturation) channel threshold expands white or yellow lanes. H (hue) toward the line colors. The outcome after applying HLS thresholding is depicted in Fig. 4(c). We have combined both of the Gradient and HLS (color) thresholding into one for the final thresholding binary image that improves the overall results of the lane detection process.

The perspective transformation is used to convert 3d world image into a 2d image. While undistorting and thresholding help to cover the crucial information, we can additionally divide that information by taking a drake at the part of the image of the road surface. To center in around the road part of the image, we move our point of view to a best down perspective of the street. While we don’t obtain any more information from this step, it’s enormously easier to isolate lane lines and measure things like curvature from this perspective.



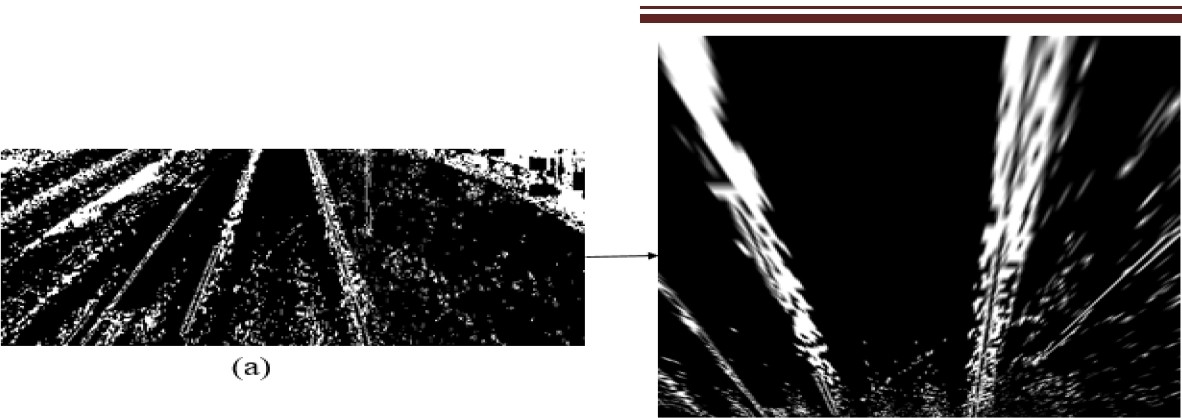


**Figure 1.5 Thresholded image**

**1.6 Sliding Window Search**:

As the lane lines already detected in an earlier frame, the information is used in a sliding window, placed around the line centers, to detect and track lane lines from bottom to the top of the image. The result of the sliding window search is demonstrated in Fig. 7. This permits us to do a highly qualified search and saves a lot of processing time.

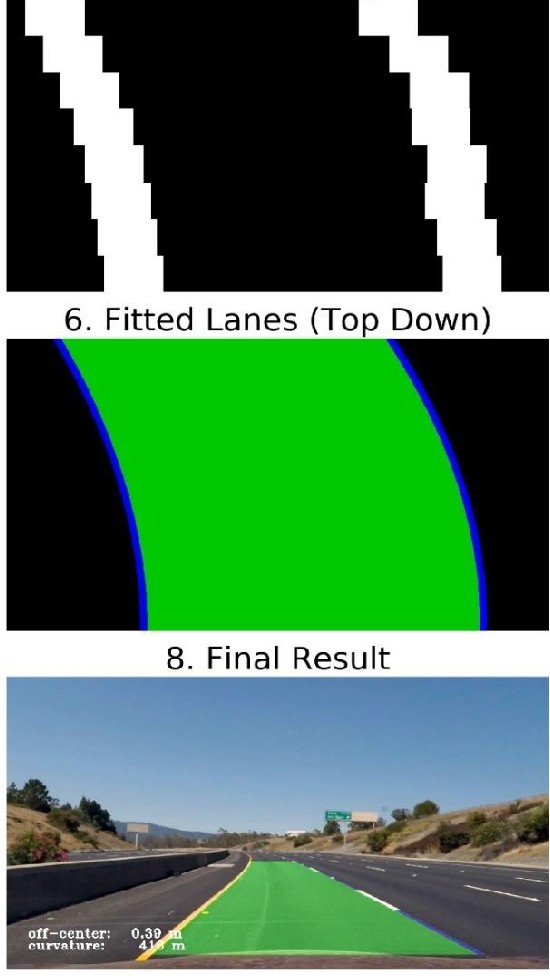
(b)



1. Undistorted Blurred

5. Masked Points



4. Masks for Left *I* Right Lanes

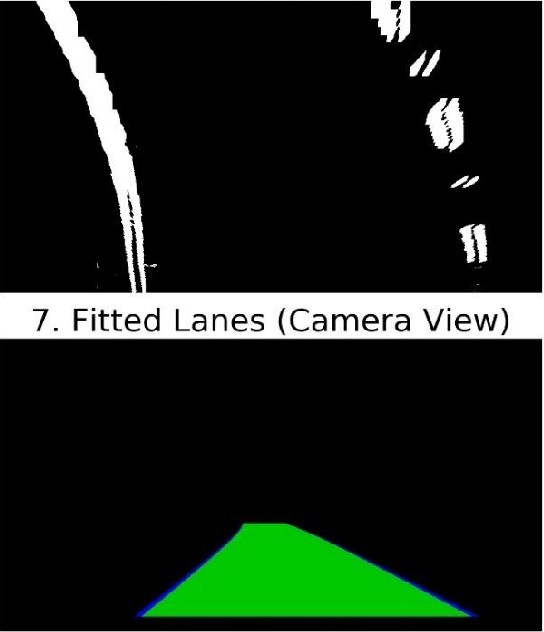


Figure 1.6 Sliding window searched images

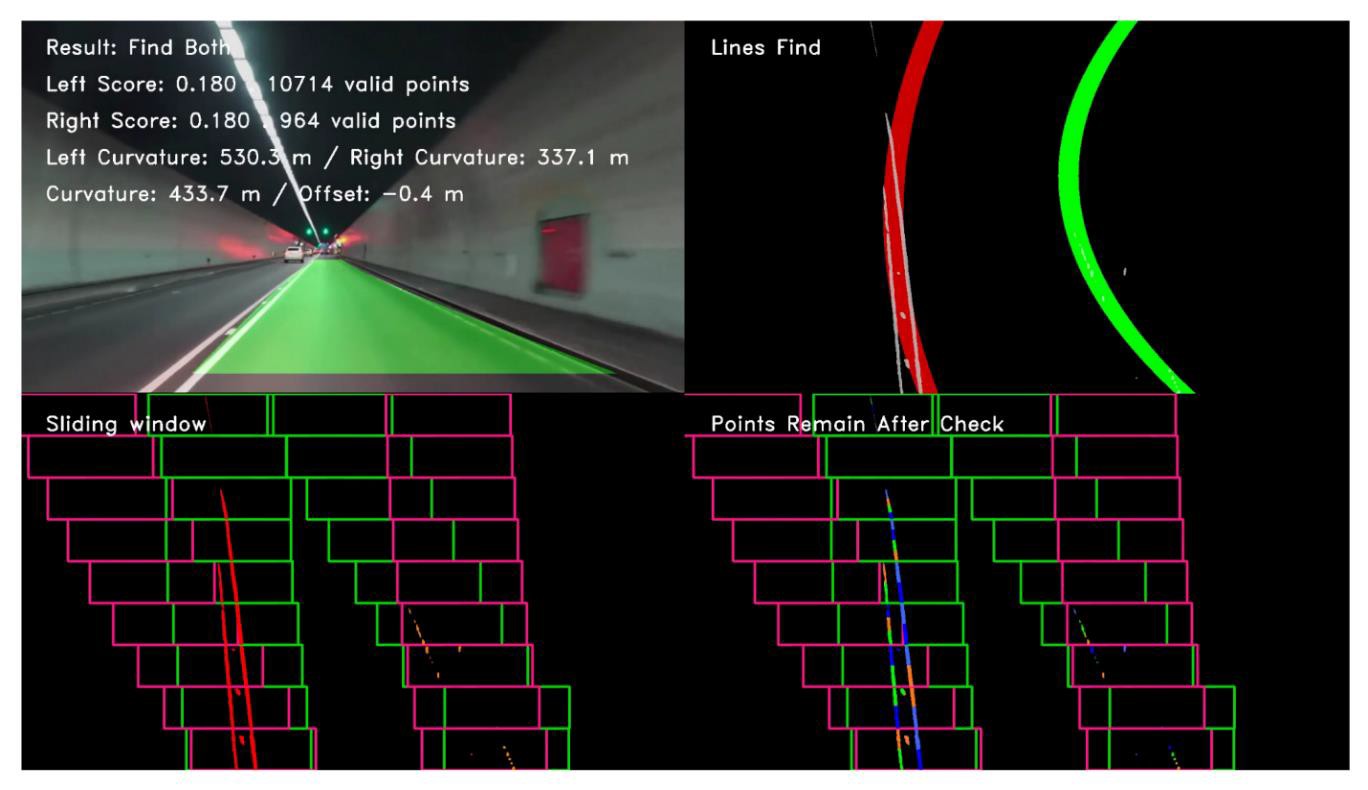
**1.7 Illustrate Lane**:

Starting from the top point of view, the lane lines are easily recognized. A sliding window search distinguishes the lane lines. The green boxes express to the windows where the lane lines are colored. Since the windows search higher, they re-center to the standard pixel position so that they resemble the lines. The shaded lines will be stepped back onto the original image. The result of illustrated lanes from the sliding window search image is shown in Fig. 8.

After illustrating the lane lines, the warp and crop operations are performed for proper visualization of the image. To warp, a 3x3 transformation matrix operation performed. Straight lines will continue straight still after the transformation. To observe this transformation matrix,

4 points on the input image and the corresponding points on the output image are needed. Among those 4 points, 3 of them should not be collinear. Then the images are cropped because, during the determination of recognizing lane lines, we only need to focus on the regions where we are likely to see the lanes.

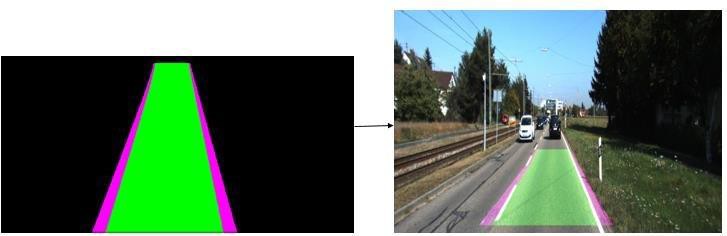
Conclusively, we get all this information and draw the results back onto the original image. The two pink lines which are recognized above are stated as lane lines, and the space between them is colored green to determine the road surface. While the pipeline prepares for a single image, it can easily be applied to processing many images to detect the lane line on the road surface. The final result of the proposed lane detection system is shown in Fig. 10.



(a)

(b)





(c)

**Figure 1.7 Ilustrated lanes**

**1.8 RELATED WORKS:**

Several studies in computer vision have been proposed in the most recent years associated with lane detection. Many institutions have been working for a long time to propose efficient lane detection system. The researches related to this area are drawn briefly as follows. Lee and Moon proposed a real-time lane detection algorithm with a Region of Interest (ROI) that is able to work with the high noise level and response in a shorter time. The system used the Kalman filter and a least square approximation of linear movement for lane tracking operation. The system detects the lane as well as track the lane also. Song et al. presented a system that can detect the lane as well as classify them using the concept of stereo vision for

the essence of Advanced Driver Assistance Systems (ADAS). They proposed a model to detect the lane using the idea of Region of Interest (ROI), and for the classification task, they used Convolutional Neural Network (CNN) structure that is trained with the KITTI dataset to classify the right or left lane. However, the system failed to detect the lanes as the disparity image was noisy. Wu et al. designed a lane detection and departure warning scheme by determining the region of interest (ROI) in the region near to the automobile. The ROI is divided into non- overlapping chunks and to get the chunk gradients and chunk angles, two basic masks are developed that decrease the computational complexity. The driving situations are classified into four classes, and the departure system is developed with respect to lane detection outcomes. From the experimental outcomes, it is shown that the average lane detection rate is 96.12% and the departure warning rate is 98.60%. However, it takes comparatively high processing time due to computing the vertical and horizontal gradients. Yoo et al. presented a lane detection technique based on the vanishing point estimation. The system used the probabilistic voting technique to detect the vanishing points of the lane segments at first. The actual lane segments were determined by setting the threshold of the vanishing points as well as the line direction. In addition, to evaluate the lane detection rate a real-time inter-frame similarity scheme was proposed that decrease the false detection rate. As the lane geometry properties do not vary expressively, the real-time assessment scheme was under the postulation. However, the system cannot be worked for shapeless roads. Ozgunalp et al. introduced a vanishing point lane detection technique for multiple curved lanes. The system combined the disparity information with a lane marking technique that is able to estimate the PVP for a non-flat street condition. The redundant information of obstacles is removed by comparing the correct and fitted disparity values. Moreover, the estimation of PVP affected by the outliers at the operation of Least Squares Fitting and sometimes unsuccessful detection of lane happens due to the plus-minus peaks value selection. Piao et al. developed a lane marking technique based on the binary blob analysis. To eliminate the perspective consequence from the road surfaces, the system used vanishing point detection and inverse perspective mapping. The binary blob filtering and blob verification methods are proposed to advance the effectiveness of the lane detection scheme. The outcomes of the system show that the average detection rate for the multi-lane dataset is

97.7%. However, the system failed to perform in a real-time environment. Jung et al. developed a lane marking modality using spatiotemporal images which are collected from the video. The spatiotemporal image created by accruing a set of pixels that are mined on a horizontal scan- line having a static location in every frame along with a time axis. Hough transform is applied to the collected images to detect lanes. The system is very effective for shortterm noises such

as mislaid lanes or obstruction by vehicles. The system obtained the computational efficacy as well as the higher detection rate. Borkar et al. proposed a lane detection technique based on inverse perspective mapping (IPM). The adaptive threshold technique is used to convert the input image to a binary image, and the predefined lane templates are used to select the lane marker candidates. RANSAC eliminated the outliers, and the Kalman filter tracked the lanes on the road surfaces. Kang et al. introduced a multi-lane detection technique based on the ridge attribute and the inverse perspective mapping (IPM). The features that are extracted have four local maxima as the identical lane is disseminated with a minor variance on the x-axis in the IPM image. The clustering-based techniques are used to detect the lanes by clustering the lane attributes nearby every indigenous supreme point.

**CHAPTER 2**

**LITERATURE SURVEY**

**2.1 INTRODUCTION**

Mapping on autonomous robotics with cameras has attracted great interest recently, partly due to advancements in computing power and optics. These developments have led to more advanced systems capable of previously impossible complexity. Some prominent and popular techniques from the literature are discussed in this chapter to give an overview of the relevant.

**2.2 STEREO VISION**

A method of measuring the distances that objects are from the sensor is an essential part of any autonomous mapping system. Stereo vision is a popular choice, since it offers large amounts of information per time step when compared to other methods such as sonar or radar, and is generally less expensive than LIDAR. This systems helps in tracking lane of the unmapped roads.

**2.2.1 Passive Stereo**

The Passive Stereo system depends on the available light in the environment and doesn’t employ any kind of external light. e-con Systems Tara and TaraXL are some examples of Passive Stereo vision cameras. Passive stereo is suitable for well lighted textured regions and works well in sunlight.

1. Pros:

• Performs well in sunlight

• Cost Effective

2. Cons:

• Mediocre performance in low light

• Mediocre performance in non-textured scenes

**2.2.2 Active Stereo**

The active stereo vision is a form of stereo vision which actively employs a light such as a laser or a structured light to simplify the stereo matching problem. Active stereo is useful

in regions where there is a lack of light and/or texture. The infrared projector or another light source will flood the scene with texture thereby cutting off the dependency of an external light source. But along with its positive, there are some negatives such as active stereo will lose its effectiveness in direct sunlight and in regions with a high interference of the same external light source technology used.

**2.3 MAPPING**

An autonomous vehicle that navigates and investigates its surroundings autonomously needs a way to represent its environment given measurements. This is commonly done using a map that is capable of describing a region. Preferably, this should be in a memory efficient way which would allow for larger areas to be explored in more detail. Some commonly used mapping techniques are discussed here to and the best option based on the requirements of this application. Each mapping technique will be evaluated based on its ability to store large amounts of data efficiently and accurately, the accessibility of its information, and how flexible the stored map is. To be considered flexible, it must be extendible when additional areas are explored, and its resolution should be variable so it can fit many different applications.

**2.3.1 Point Clouds**

The naive way to represent data points in world coordinates is to use a point cloud, where every known point in the map is represented with a 3D coordinate. Although it offers a very simple way to represent information and does not require additional processing, it provides limited usability for navigation and processing techniques due to the complexity of accessing information about a particular region. The problem with point clouds is that they do not combine measurements that represent essentially the same information, and therefore its memory usage can grow without bound. Eventually, many points will contain redundant information. The below figure demonstrates how point cloud data is visualized. 2D & 3D Occupancy Grid Mapping Using Depth Sensors

**2.3.2 Occupancy Grids**

Currently the most popular solution for mapping used in robotics is the occupancy grid mapping technique that represents an environment as a tessellated map1 consisting of a number of 2D squares or 3D cubes. The technique was originally coined by Elfes, who created a probabilistic implementation capable of creating 2D or 3D maps. He also developed a method of incorporating the uncertainty in measurements from sonar and stereo vision sensors by modelling their errors.The basic principle of an occupancy grid is to discretise the world into a

number of cells and assigning each a binary classification where it can be either occupied or empty. To incorporate uncertainty into the representation, the classification is stored as a probability, where 1 indicates that it is definitely occupied, 0 indicates it is definitely empty and

0:5 that no information about its occupancy isavailable. The technique is developed specifically to map static environments that do not change, since mapping dynamic objects entails an entirely different approach where dynamic object are identified and tracked over time. 2D &

3D Occupancy Grid Mapping Using Depth Sensors The original implementation divided the environment into equally sized blocks and stored only a probability that each contained an obstacle. This could result in a large number of superfluous homogeneous neighboring blocks, which was improved by adaptive grid mapping.

**CHAPTER 3**

**METHODOLOGY**

**3.1 Preprocessing**

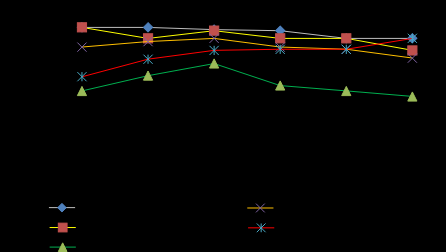
Preprocessing is an important part of image processing and an important part of lane detection. Preprocessing can help reduce the complexity of the algorithm, thereby reducing subsequent program processing time. The video input is a RGB-based color image sequence obtained from the camera. In order to improve the accuracy of lane detection, many researchers employ different image preprocessing techniques.Smoothing and filtering graphics is a common image preprocessing technique. The main purpose of filtering is to eliminate image noise and enhance the effect of the image. Low-pass or high-pass filtering operation can be performed for

2D images, low-pass filtering (LPF) is advantageous for denoising, and image blurring and high-pass filtering (HPF) are used to find image boundaries. In order to perform the smoothing operation, an average, median , or Gaussian filter could be used. In , in order to preserve detail and remove unwanted noise, Xu and Li firstly use a median filter to filter the image and then use an image histogram in order to enhance the grayscale image.

**3.2. 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. Figures 5(a) and 5(b) are images of RGB colour space and HSV colour space, respectively. HSV represents hue, saturation, and value. As can be seen in Figure 6, 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.



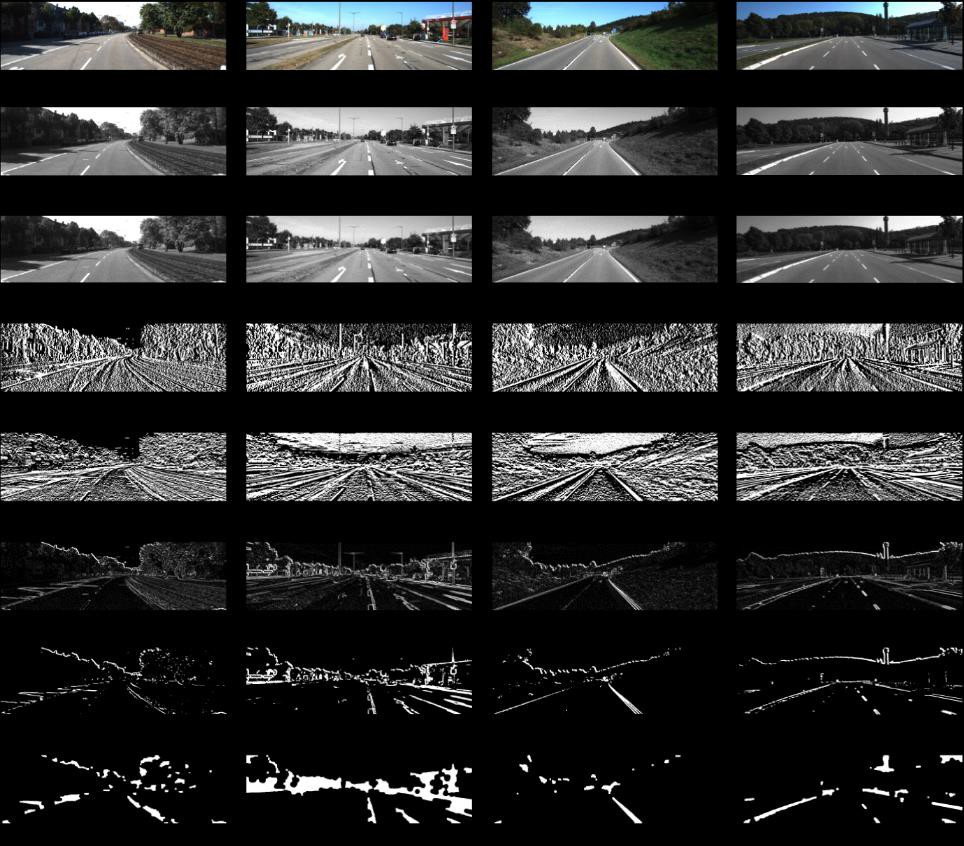


**Figure 3.2 Colour transformed image**

**3.3. Basic Preprocessing**

A large number of frames in the video will be preprocessed. The images are individually gray scaled, blurred, X-gradient calculated, Y-gradient calculated, global gradient calculated, thresh of frame, and morphological closure. In order to cater for different lighting conditions, an adaptive threshold is implemented during the preprocessing phase. Then, we remove the spots in the image obtained from the binary conversion and perform the morphological closing operation. As can be seen from Figure [7](https://www.hindawi.com/journals/am/2018/8320207/fig7/), the basic preprocessed frames cannot be very good at removing noise. It can be seen from the results after the morphological closure that although preliminary lane information can be obtained, there is still a large amount of noise.

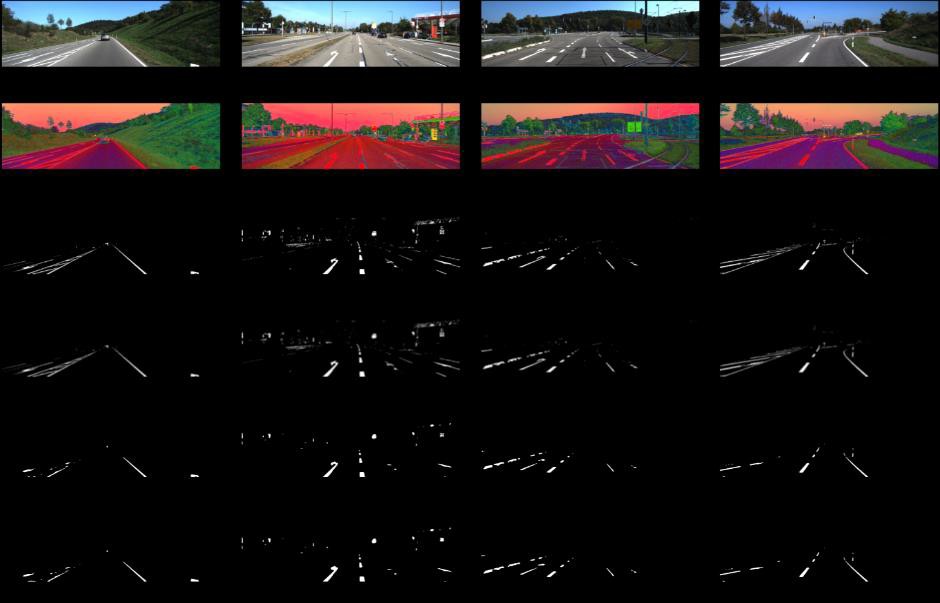
**Figure 3.3 Gray scaled, Blurred images**



**3.4. Adding Colour Extraction in Preprocessing**

In order to improve the accuracy of lane detection, we add a feature extraction module in the preprocessing stage. The purpose of feature extraction is to keep any features that may be lane and remove features that may be nonlane. This paper mainly carries on the feature extraction to the colour. After the graying of the image and colour model conversion, we add the white feature extraction and then carry out the conventional preprocessing operation in turn. The process of the colour extraction proposed in this paper is shown in Figure 8.

**Figure 3.4 Adding Colour Extraction in Preprocessing**



**3.5. Adding Edge Detection in Preprocessing**

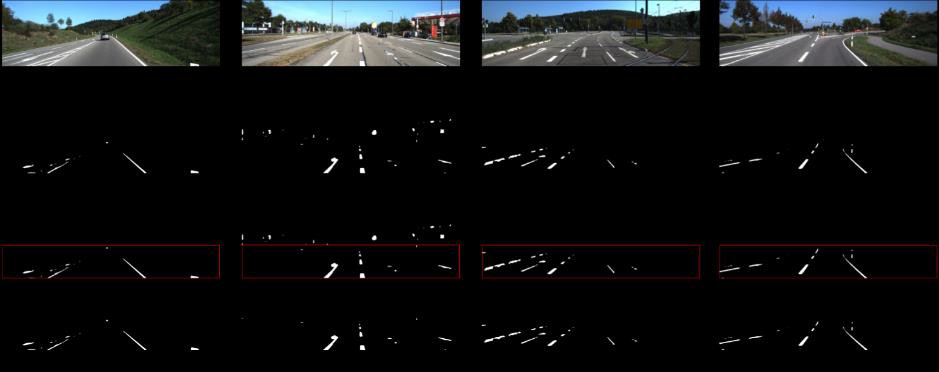
This paper has carried out edge detection two times successively; the first time is to perform a wide range of edge detection extraction in the entire frame image. In the second, the edge detection is performed again after the lane detection after ROI selection. This detection further improves the accuracy 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 nonmaximal value of the gradient amplitude. Finally, we need to use a double-threshold algorithm to detect and connect edges. Figure 9 shows the image after extraction with Canny edge detection.



**Figure 3.5 Adding edge detection in preprocessing**

**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. We only set the input image on the ROI area and this method can increase the speed and accuracy of the system. In this paper, we use the standard KITTI 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 10 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.



**Figure 3.5 ROI selecting images**

**Lane Detection**

The lane detection module is mainly divided into lane edge detection and linear lane detection. This section implements the basic functions of lane detection and performs lane detection based on improved preprocessing and the proposed ROI selection.

**Edge Detection**

Feature extraction is very important for lane detection. There are many common methods used for edge detection, such as Canny transform, Sobel transform, and Laplacian transform. We have selected Canny transform which is better. As shown in Figure 11, we performed Canny edge detection after the proposed ROI selection.

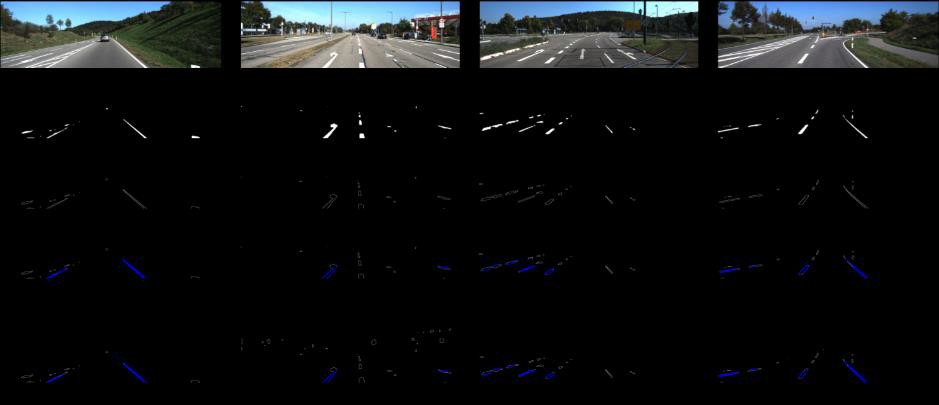


**Figure 3.5.1 Edge detected image**

The methods of lane detection include feature based methods and model-based methods. The method based feature is used in this paper to detect the colour and edge features of lanes in order to improve the accuracy and efficiency of lane detection.There are two methods to achieve straight lane detection. One is to use the Hough line detected function encapsulated by the OpenCV library commonly used for image processing, and draw lane lines in the corresponding area of the original image. The other is self-programming. In the header file, the ROI area is traversed to perform line detection for a specific range of angles.Both methods can be reflected in the video, and the first method runs faster. Since this article focuses on the accuracy and efficiency of lane detection, we chose the first method (Hough line function in the OpenCV library) to run faster for linear detection. Moreover, because the Hough transform is insensitive

to noise and can process straight lines well, Hough transform is used to extract lane line parameters in each frame of the image sequence for lane detection.

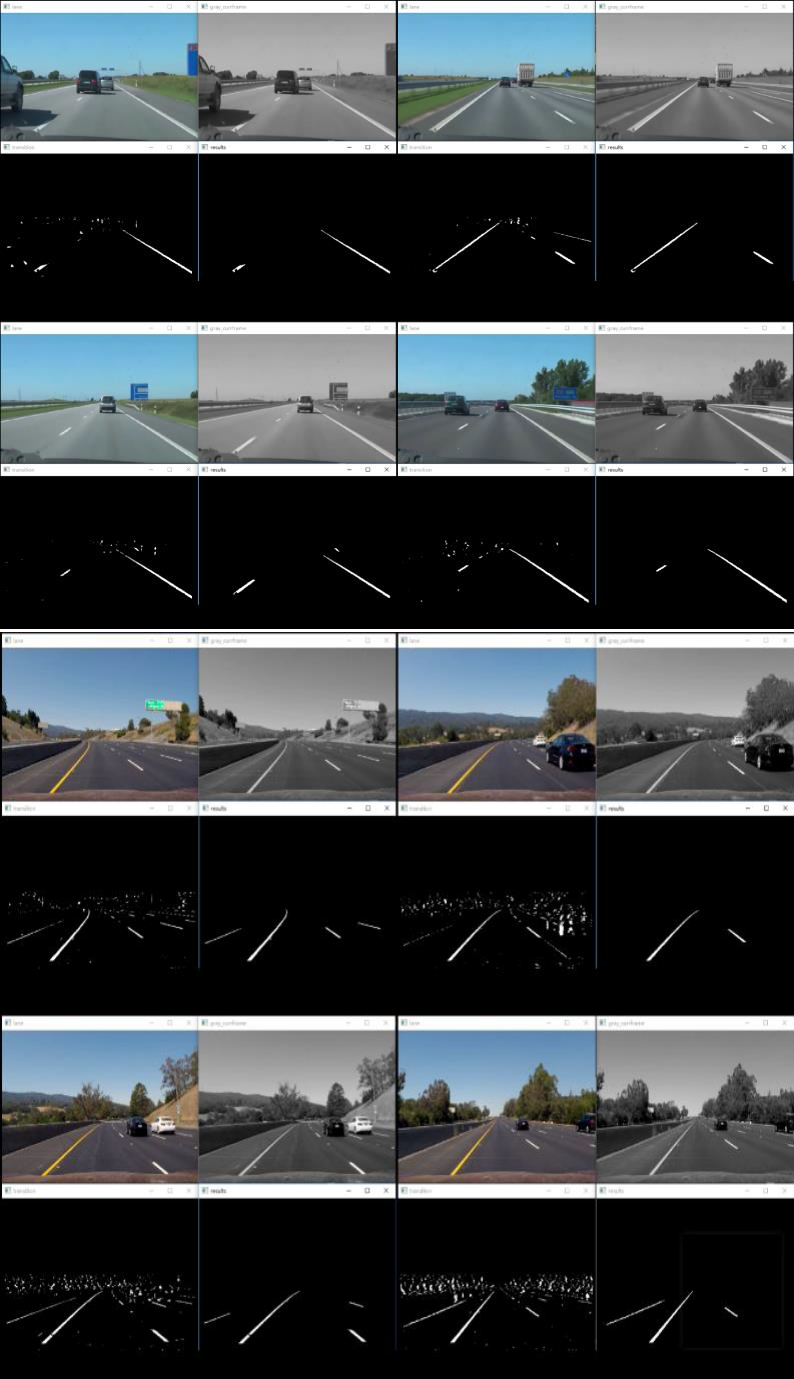
In image processing, the Hough transform is used to detect any shape that can be expressed in a mathematical formula, even if the shape is broken or somewhat distorted. Compared with other methods, the Hough transform can find noise reduction better than other methods. The classic Hough transform is often used to detect lines, circles, ellipses, etc. As shown in Figure 12, lane detection uses Hough of sample frames (a), (b), (c), and (d).



**Figure 3.5.2 Lane detected image**

**3.6 Lane Tracking Using Extended Kalman Filter**

After completing the lane detection, the next step is to track the lane, which is also a key technology for smart and automated vehicle (SAV).Image edge detection technology and linear lane detection are technologies used to detect lane; then EKF is used to track these parameters one by one. In this way, the tracking of lane lines is converted into the tracking of lane line parameters, which not only improves the tracking speed, but also introduces the method of Kalman tracking to improve the tracking accuracy.The experimental results are shown in Figures 13 and 14.The real-time tracking lane line is detected in the video stream. Figure 13 shows different results of lane detection at different times (i), (ii), (iii), and (iv) in one video. Figure 14 shows different results of lane detection at different times (i), (ii), (iii), and (iv) in another video.



**3.6 Lane Tracking Using Extended Kalman Filter image**

**CHAPTER 4**

**SOFTWARE TOOLS REQUIRED**

**INTODUCTION**

There are different techniques, which can be applied based on the domain, task at hand, and the availability of data.

**4.1 Tools**

Tools required for constructing occupancy grid mapping:

**4.1.1 Python**

Python is a multiparadigm, general-purpose, interpreted, high-level programming language. Python allows programmers to use different programming styles to create simple or complex programs, get quicker results and write code almost as if speaking in a human language.

**4.1.2 Graphical Processing Unit (GPU):**

A GPU is a parallel programming setup involving GPUs &amp; CPUs which can process &amp; analyse data in a similar way to image or other graphic form. GPUs were created for better and more general graphic processing, but were later found to fit scientific computing well. This is because most of the graphic processing involves applying operations on large matrices.Due to large datasets, the CPU takes up a lot of memory while training the model. The standalone GPU, on the other hand, comes with a dedicated VRAM memory. Thus, CPU’s memory can be used for other tasks. But, transferring large chunks of memory from CPU to GPU is a bigger challenge. Computing huge and complex jobs takes up a lot of clock cycles in CPU. The reason being, CPU takes up the jobs sequentially and it has a fewer number of cores than its counterpart, GPU. But though GPUs are faster,the time taken to transfer huge amounts of data from CPU to GPU can lead to higher overhead time depending on the architecture of the processors. The best CPUs have about 50GB/s while the best GPUs have 750GB/s memory bandwidth**.** The use of GPUs for scientific computing started some time back in 2001 with implementation of Matrix multiplication. One of the first common algorithm to be implemented on GPU in faster manner was LU factorization in 2005. But, at this time researchers had to code

every algorithm on a GPU and had to understand low level graphic processing. In 2006, Nvidia came out with a high-level language CUDA, which helps you write programs from graphic processors in a high-level language. This was probably one of the most significant change in the way researchers interacted with GPUs.

**4.1.3 OpenCV**

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code. The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 18 million. The library is used extensively in companies, research groups and by governmental bodies. Along with well established companies like Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota that employ the library, there are many start-ups such as Applied Minds, VideoSurf, and Zeitera, that make extensive use of OpenCV. OpenCV’sdeployed uses span the range from stitching street view images together, detecting intrusions in surveillance video in Israel, monitoring mine equipment in China helping robots navigate and pick up objects at Willow Garage, detection of swimming pool drowning accidents in Europe, running interactive art in Spain and New York,checking runways for debris in Turkey, inspecting labels on products in factories around the world on to rapid face detection in Japan.It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux,Android and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available. A full-featured CUDA and OpenCL interfaces are being actively developed right now. There are over 500algorithms and about 10 times as many functions that compose or support those algorithms. OpenCV is written natively in C++ and has a templated interface that works seamlessly with STL containers.

**4.1.4 NumPy**

NumPy is a library for the Python programming language, adding support for large, multi- dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. You already read in the introduction that NumPy arrays are a bit like Python lists, but still very much different at the same time. For those of you who are new to the topic, let’s clarify what it exactly is and what it’s good for. As the name gives away, a NumPy array is a central data structure of the NumPy library. The library’s name is short for “Numeric Python” or “Numerical Python”. In other words, NumPy is a Python library that is the core library for scientific computing in Python. It contains a collection of tools and techniques that can be used to solve on a computer mathematical models of problems in Science and Engineering. One of these tools is a high-performance multidimensional array object that is a powerful data structure for efficient computation of arrays and matrices. To work with these arrays, there’s a vast amount of high-level mathematical functions operate on these matrices and arrays.

• **Basic of NumPy**

NumPy’s main object is the homogeneous multidimensional array. It is a table of elements (usually numbers), all of the same type, indexed by a tuple of non-negative integers. In NumPy dimensions are called axes. For example, the coordinates of a point in 3D space [1, 2, 1] has one axis. That axis has 3 elements in it, so we say it has a length of 3. In the example pictured below, the array has 2 axes. The first axis has a length of 2, the second axis has a length of 3. [[

1., 0., 0.], [ 0., 1., 2.]] NumPy’s array class is called ndarray. It is also known by the alias array. Note that NumPy. Array is not the same as the Standard Python Library class array. Array, which only handles one-dimensional arrays and offers less functionality. The more important attributes of a ndarray object are:

**ndarray.ndim**

The number of axes (dimensions) of the array.

**ndarray.shape**

The dimensions of the array. This is a tuple of integers indicating the size of the array in each dimension. For a matrix with n rows and m columns, shape will be (n,m). The length of the shape tuple is therefore the number of axes, ndim.

**ndarray.size**

The total number of elements of the array. This is equal to the product of the elements of shape.

**ndarray.dtype**

An object describing the type of the elements in the array. One can create orspecify dtype using standard Python types. Additionally, NumPy provides types of its own. numpy.int32, numpy.int16, and NumPy. float64 are some examples.

**ndarray.itemsize**

The size in bytes of each element of the array. For example, an array of elements of type float64 has item size 8 (=64/8), while one of type complex32 has item size 4 (=32/8). It is equivalent to ndarray.dtype.itemsize.

**ndarray.data**

The buffer containing the actual elements of the array. Normally, we won’t need to use this

attribute because we will access the elements in an array using indexing facilities.

**Example:** -Using NumPy to mask an image import numpy as np

**4.1.5 SciPy**

SciPy is a free and open-source Python library used for scientific computing and technical computing. SciPy contains modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers and other tasks common in science and engineering. SciPy is a collection of mathematical algorithms and convenience functions built on the NumPy extension of Python. It adds significant power to the interactive Python session by providing the user with high-level commands and classes for manipulating and visualizing data. With SciPy an interactive Python session becomes a data- processing and system-prototyping environment rivaling systems such as MATLAB, IDL, Octave, R-Lab, and SciLab. The additional benefit of basing SciPy on Python is that this also makes a powerful programming language available for use in developing sophisticated programs and specialized applications. Scientific applications using SciPy benefit from the development of additional modules in numerous niches of the software landscape by developers across the world. Everything from parallel programming to web and data- base subroutines and classes have been made available to the Python programmer. All of this power is available in addition to the mathematical libraries in SciPy. SciPy is another of Python&#39;s core scientific modules (like NumPy) and can be used for basic image manipulation and processing tasks. In particular, the submodule scipy.ndimage (in SciPy v1.1.0) provides functions operating on n-

dimensional NumPy arrays. The package currently includes functions for linear and non-linear filtering, binary morphology, B-spline interpolation, and object measurements.Image processing and analysis are generally seen as operations on two-dimensional arrays of values. There are however a number of fields where images of higher dimensionality must be analyzed. Good examples of these are medical imaging and biological imaging. numpy is suited very well for this type of applications due its inherent multidimensional nature.

**Example:-** Using SciPy for blurring using a Gaussian filter :

from scipy import misc,ndimage face = misc.face()

blurred\_face = ndimage.gaussian\_filter(face, sigma=3)

very\_blurred = ndimage. gaussian\_filter (face, sigma=5)

#Results

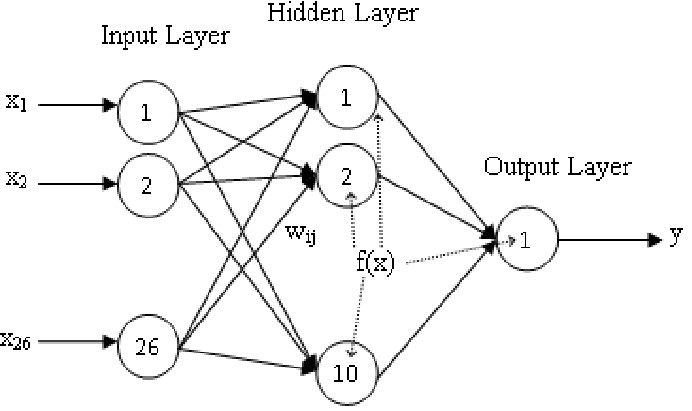
plt.imshow(&lt;image to be displayed&gt;)

OpenCV was started at Intel in 1999 by Gary Bradsky and the first release came out in

2000. Vadim Pisarevsky joined Gary Bradsky to manage Intel’s Russian software OpenCV team. In 2005, OpenCV was used on Stanley, the vehicle who won 2005 DARPA Grand Challenge. Later its active development continued under the support of Willow Garage, with Gary Bradsky and Vadim Pisarevsky leading the project. Right now, OpenCV supports a lot of algorithms related to Computer Vision and Machine Learning and it is expanding day-by-day. Currently OpenCV supports a wide variety of programming languages like C++, Python, Java etc and is available on different platforms including Windows, Linux, OS X, Android, iOS etc. Also, interfaces based on CUDA and OpenCL are also under active development for high-speed GPU operations. OpenCV-Python is the Python API of OpenCV. It combines the best qualities of OpenCV C++ API and Python language. OpenCV (Open Source Computer Vision) is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage then Itseez (which was later acquired by Intel). The library is cross-platform and free for use under the open-source BSD license. OpenCV supports the deep learning frameworks TensorFlow, Torch/PyTorch and Caffe.

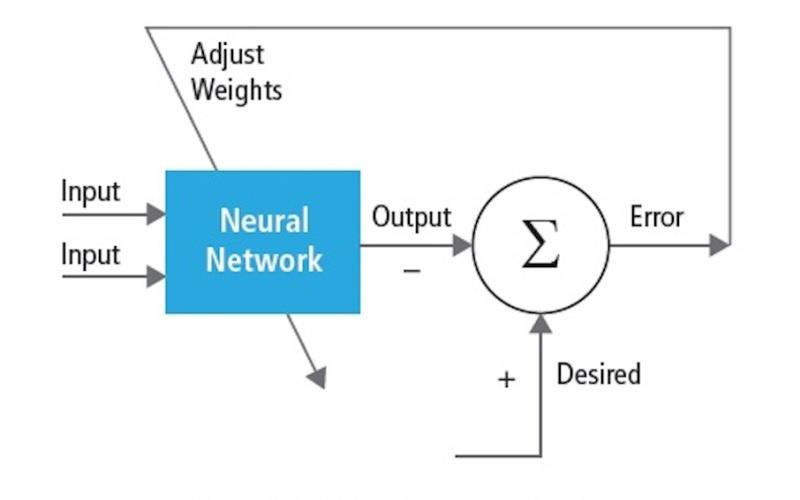
**4.2 Introduction to CNN**

A neural network is a system of interconnected artificial “neurons” that exchange messages between each other. The connections have numeric weights that are tuned during the training process, so that a properly trained network will respond correctly when presented with an image or pattern to recognize. The network consists of multiple layers of feature-detecting “neurons”. Each layer has many neurons that respond to different combinations of inputs from the previous layers. As shown in Figure 1, the layers are built up so that the first layer detects a set of primitive patterns in the input, the second layer detects patterns of patterns, the third layer detects patterns of those patterns, and so on. Typical CNNs use 5 to 25 distinct layers of pattern recognition.



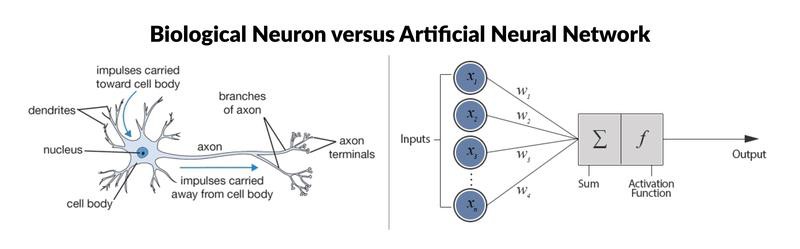
**Figure 4.2.1 An artificial neural network**

Training is performed using a “labeled” dataset of inputs in a wide assortment of representative input patterns that are tagged with their intended output response. Training uses general- purpose methods to iteratively determine the weights for intermediate and final feature neurons. Figure 2 demonstrates the training process at a block level.



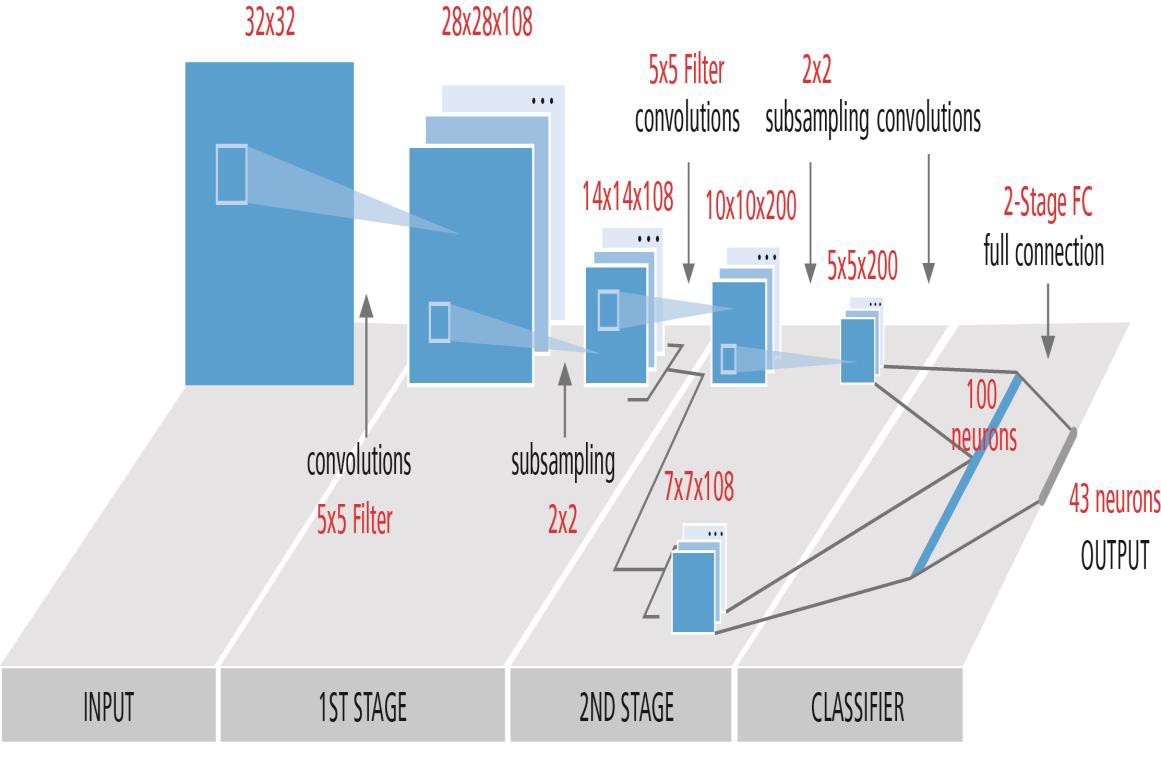
**Figure 4.2.2 Training of neural networks**

Neural networks are inspired by biological neural systems. The basic computational unit of the brain is a neuron and they are connected with synapses. Figure 3 compares a biological neuron with a basic mathematical model



In a real animal neural system, a neuron is perceived to be receiving input signals from its dendrites and producing output signals along its axon. The axon branches out and connects via synapses to dendrites of other neurons. When the combination of input signals reaches some threshold condition among its input dendrites, the neuron is triggered and its activation is communicated to successor neurons. In the neural network computational model, the signals that travel along the axons (e.g., x0) interact multiplicatively (e.g., w0x0) with the dendrites of the other neuron based on the synaptic strength at that synapse (e.g., w0). Synaptic weights are learnable and control the influence of one neuron or another. The dendrites carry the signal to the cell body, where they all are summed. If the final sum is above a specified threshold, the neuron fires, sending a spike along its axon. In the computational model, it is assumed that the precise timings of the firing do not matter and only the frequency of the firing communicates information. Based on the rate code interpretation, the firing rate of the neuron is modeled with an activation function f that represents the frequency of the spikes along the axon. A common choice of activation function is sigmoid. In summary, each neuron calculates the dot product of inputs and weights, adds the bias, and applies non-linearity as a trigger function (for example, following a sigmoid response function). A CNN is a special case of the neural network described above. A CNN consists of one or more convolutional layers, often with a subsampling layer, which are followed by one or more fully connected layers as in a standard neural network. The design of a CNN is motivated by the discovery of a visual mechanism, the visual cortex, in the brain. The visual cortex contains a lot of cells that are responsible for detecting light in small, overlapping sub-regions of the visual field, which are called receptive fields. These cells act as local filters over the input space, and the more complex cells have larger receptive fields.

The convolution layer in a CNN performs the function that is performed by the cells in the visual cortex . A typical CNN for recognizing traffic signs is shown in Figure 4. Each feature of a layer receives inputs from a set of features located in a small neighborhood in the previous layer called a local receptive field. With local receptive fields, features can extract elementary visual features, such as oriented edges, end-points, corners, etc., which are then combined by the higher layers. In the traditional model of pattern/image recognition, a hand-designed feature extractor gathers relevant information from the input and eliminates irrelevant variability. The extractor is followed by a trainable classifier, a standard neural network that classifies feature vectors into classes. In a CNN, convolution layers play the role of feature extractor. But they are not hand designed. Convolution filter kernel weights are decided on as part of the training process. Convolution layers are able to extract the local features because they restrict the receptive fields of the hidden layers to be local.



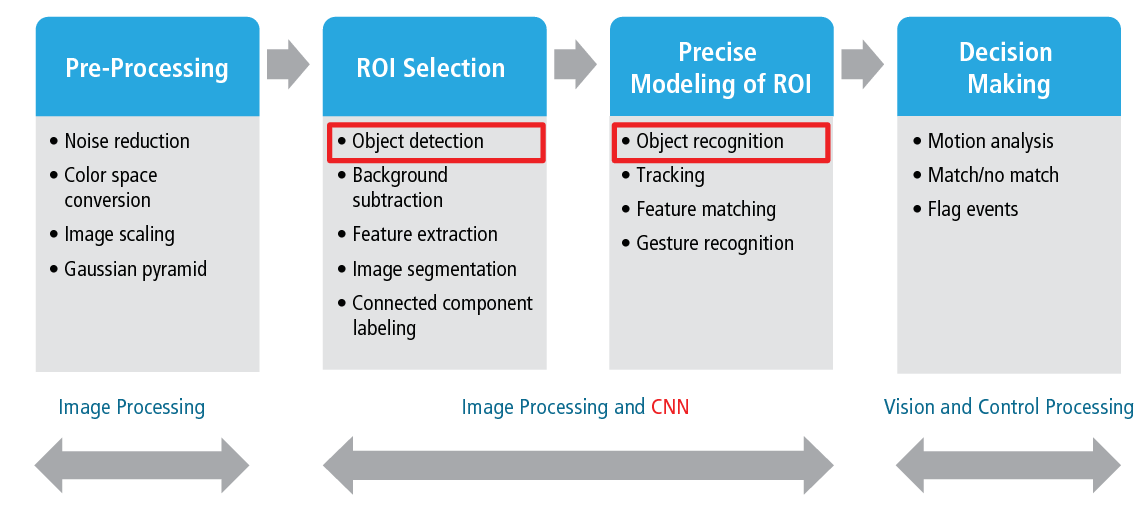
**Figure 4.2.3 Typical block diagram of a CNN**

CNNs are used in variety of areas, including image and pattern recognition, speech recognition, natural language processing, and video analysis. There are a number of reasons that convolutional neural networks are becoming important. In traditional models for pattern recognition, feature extractors are hand designed. In CNNs, the weights of the convolutional layer being used for feature extraction as well as the fully connected layer being used for classification are determined during the training process. The improved network structures of CNNs lead to savings in memory requirements and computation complexity requirements and, at the same time, give better performance for applications where the input has local correlation (e.g., image and speech).

Large requirements of computational resources for training and evaluation of CNNs are sometimes met by graphic processing units (GPUs), DSPs, or other silicon architectures

optimized for high throughput and low energy when executing the idiosyncratic patterns of CNN computation. In fact, advanced processors such as the Tensilica Vision P5 DSP for Imaging and Computer Vision from Cadence have an almost ideal set of computation and memory resources required for running CNNs at high efficiency.

In pattern and image recognition applications, the best possible correct detection rates (CDRs) have been achieved using CNNs. For example, CNNs have achieved a CDR of 99.77% using the MNIST database of handwritten digits, a CDR of 97.47% with the NORB dataset of 3D objects [6], and a CDR of 97.6% on ~5600 images of more than 10 objects . CNNs not only give the best performance compared to other detection algorithms, they even outperform humans in cases such as classifying objects into fine-grained categories such as the particular breed of dog or species of bird .shows a typical vision algorithm pipeline, which consists of four stages: pre-processing the image, detecting regions of interest (ROI) that contain likely objects, object recognition, and vision decision making. The pre-processing step is usually dependent on the details of the input, especially the camera system, and is often implemented in a hardwired unit outside the vision subsystem. The decision making at the end of pipeline typically operates on recognized objects—It may make complex decisions, but it operates on much less data, so these decisions are not usually computationally hard or memory-intensive problems. The big challenge is in the object detection and recognition stages, where CNNs are now having a wide impact.



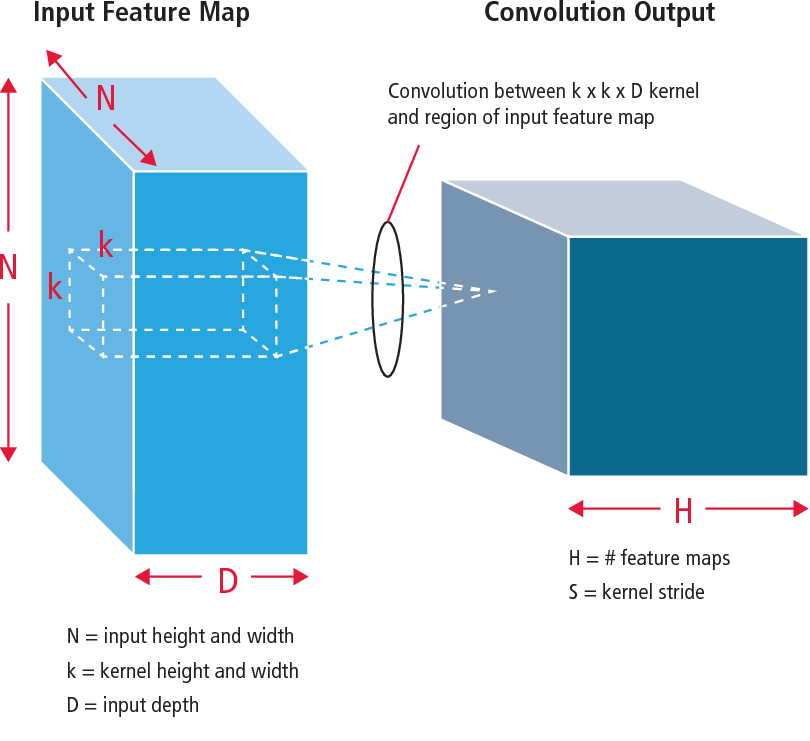
**Figure 4.2.4 Vision algorithm pipeline**

**4.3 Layers of CNNs**

By stacking multiple and different layers in a CNN, complex architectures are built for classification problems. Four types of layers are most common: convolution layers, pooling/subsampling layers, non-linear layers, and fully connected layers.

**Convolution layers**

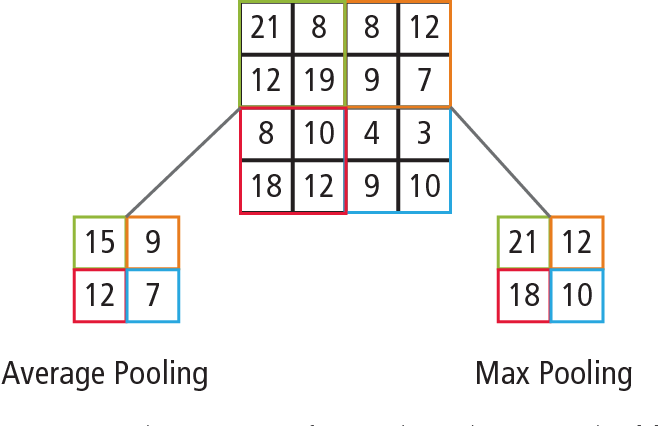
The convolution operation extracts different features of the input. The first convolution layer extracts low-level features like edges, lines, and corners. Higher-level layers extract higher-level features. Figure 6 illustrates the process of 3D convolution used in CNNs. The input is of size N x N x D and is convolved with H kernels, each of size k x k x D separately. Convolution of an input with one kernel produces one output feature, and with H kernels independently produces H features. Starting from top-left corner of the input, each kernel is moved from left to right, one element at a time. Once the top-right corner is reached, the kernel is moved one element in a downward direction, and again the kernel is moved from left to right, one element at a time. This process is repeated until the kernel reaches the bottom-right corner. For the case when N = 32 and k = 5 , there are 28 unique positions from left to right and 28 unique positions from top to bottom that the kernel can take. Corresponding to these positions, each feature in the output will contain 28x28 (i.e., (N-k+1) x (N-k+1)) elements. For each position of the kernel in a sliding window process, k x k x D elements of input and k x k x D elements of kernel are element-byelement multiplied and accumulated. So to create one element of one output feature, k x k x D multiply-accumulate operations are required.



**Figure 4.3 Pictorial representation of convolution process**

**4.4 Pooling/subsampling layers:-**

The pooling/subsampling layer reduces the resolution of the features. It makes the features robust against noise and distortion. There are two ways to do pooling: max pooling and average pooling. In both cases, the input is divided into non-overlapping two-dimensional spaces. For example, in Figure 4, layer 2 is the pooling layer. Each input feature is 28x28 and is divided into 14x14 regions of size 2x2. For average pooling, the average of the four values in the region are calculated. For max pooling, the maximum value of the four values is selected. Figure 7 elaborates the pooling process further. The input is of size 4x4. For 2x2 subsampling, a 4x4 image is divided into four non-overlapping matrices of size 2x2. In the case of max pooling, the maximum value of the four values in the 2x2 matrix is the output. In case of average pooling, the average of the four values is the output. Please note that for the output with index (2,2), the result of averaging is a fraction that has been rounded to nearest integer.



**Figure 4.4 Pictorial representation of max pooling and average pooling**

**4.5 Non-linear layers:-**

Neural networks in general and CNNs in particular rely on a non-linear “trigger”

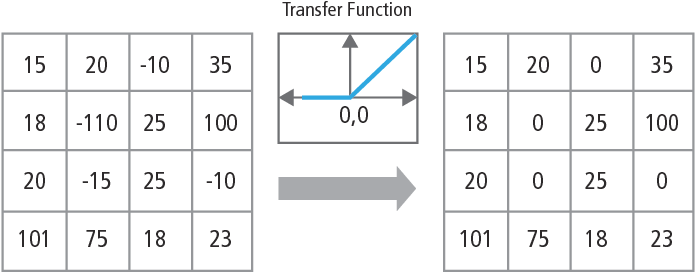
function to signal distinct identification of likely features on each hidden layer. CNNs may use

a variety of specific functions —such as rectified linear units (ReLUs) and continuous trigger

(non-linear) functions—to efficiently implement this non-linear triggering.

**ReLU**

A ReLU implements the function y = max(x,0), so the input and output sizes of this layer are the same. It increases the nonlinear properties of the decision function and of the overall network without affecting the receptive fields of the convolution layer. In comparison to the other non-linear functions used in CNNs (e.g., hyperbolic tangent, absolute of hyperbolic tangent, and sigmoid), the advantage of a ReLU is that the network trains many times faster. ReLU functionality is illustrated in Figure 8, with its transfer function plotted above the arrow.



**Figure 4.5.1 Pictorial representation of ReLU functionality**

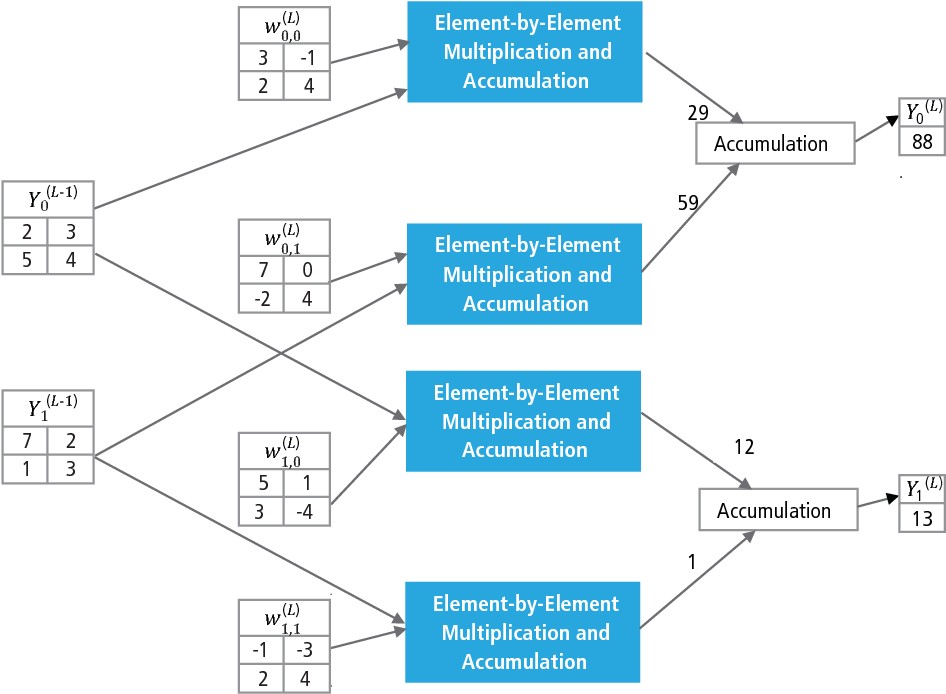
Continuous trigger (non-linear) function The non-linear layer operates element by element in each feature. A continuous trigger function can be hyperbolic tangent , absolute of hyperbolic tangent or sigmoid .

Fully connected layers Fully connected layers are often used as the final layers of a

CNN. These layers mathematically sum a weighting of the previous layer of features, indicating

the precise mix of “ingredients” to determine a specific target output result. In case of a fully connected layer, all the elements of all the features of the previous layer get used in the calculation of each element of each output feature. Figure 13 explains the fully connected layer L. Layer L-1 has two features, each of which is 2x2, i.e., has four elements. Layer L has two

features, each having a single element.



**4.6 Why CNN?**

**Figure 4.5.2 Processing of a fully connected layer**

While neural networks and other pattern detection methods have been around for the past 50 years, there has been significant development in the area of convolutional neural networks in the recent past. This section covers the advantages of using CNN for image recognition.

**Ruggedness to shifts and distortion in the image**

Detection using CNN is rugged to distortions such as change in shape due to camera lens, different lighting conditions, different poses, presence of partial occlusions, horizontal and

vertical shifts, etc. However, CNNs are shift invariant since the same weight configuration is used across space. In theory, we also can achieve shift invariantness using fully connected layers. But the outcome of training in this case is multiple units with identical weight patterns at different locations of the input. To learn these weight configurations, a large number of training instances would be required to cover the space of possible variations.

**Fewer memory requirements :-**

In this same hypothetical case where we use a fully connected layer to extract the features, the input image of size 32x32 and a hidden layer having 1000 features will require an order of 106 coefficients, a huge memory requirement. In the convolutional layer, the same coefficients are used across different locations in the space, so the memory requirement is drastically reduced.

**Easier and better training :-**

Again using the standard neural network that would be equivalent to a CNN, because the number of parameters would be much higher, the training time would also increase proportionately. In a CNN, since the number of parameters is drastically reduced, training time is proportionately reduced. Also, assuming perfect training, we can design a standard neural network whose performance would be same as a CNN. But in practical training, a standard neural network equivalent to CNN would have more parameters, which would lead to more noise addition during the training process. Hence, the performance of a standard neural network equivalent to a CNN will always be poorer.

**Recognition Algorithm for GTSRB Dataset**:-

The German Traffic Sign Recognition Benchmark (GTSRB) was a multi-class, single-image classification challenge held at the International Joint Conference on Neural Networks (IJCNN)

2011, with the following requirements:

• 51,840 images of German road signs in 43 classes (Figures 14 and 15)

• Size of images varies from 15x15 to 222x193

• Images are grouped by class and track with at least 30 images per track

• Images are available as color images (RGB), HOG features, Haar features, and color

histograms

• Competition is only for the classification algorithm; algorithm to find region of interest in the frame is not required

• Temporal information of the test sequences is not shared, so temporal dimension cannot be

used in the classification algorithm

**Figure 4.6.1 GTSRB ideal traffic signs**



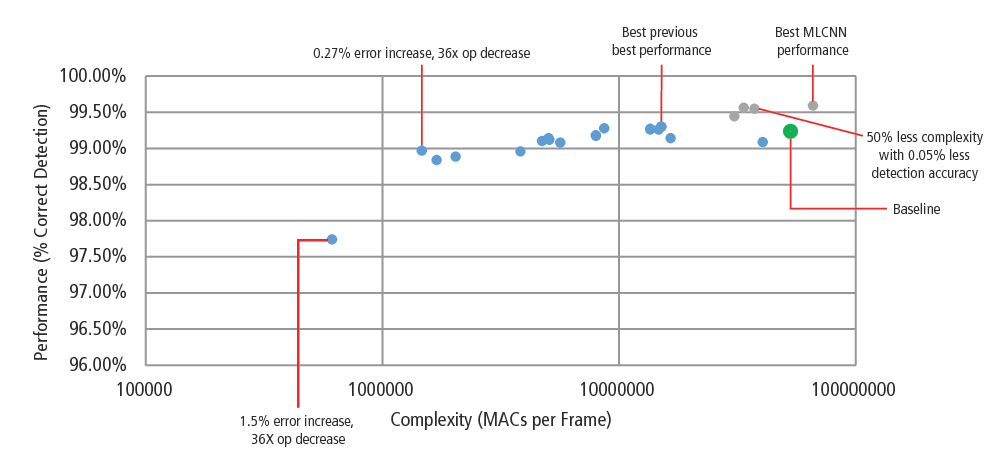
**Cadence Algorithm for Traffic Sign Recognition in GTSRB Dataset :-**

Cadence has developed various algorithms in MATLAB for traffic sign recognition using the GTSRB dataset, starting with a baseline configuration based on a well-known paper on sign recognition [1]. The correct detection rate of 99.24% and compute effort of almost >50 million multiply-adds per sign is shown as a thick green point in Figure 16. Cadence has achieved meaningfully better results using our new proprietary Hierarchical CNN approach. In

this algorithm, 43 traffic signs have been divided into five families. In total, we implement six smaller CNNs. The first CNN decides which family the received traffic sign belongs to. Once the family of the sign is known, the CNN (one of the remaining five) corresponding to the family detected is run to decide the traffic sign within that family. Using this algorithm, Cadence has achieved a correct detection rate of 99.58%, the best CDR achieved on GTSRB to date.

**Algorithm for Performance vs. Complexity Tradeoff**

In order to control the complexity of CNNs in embedded applications, Cadence has also developed a proprietary algorithm using eigenvalue decomposition that reduces a trained CNN to its canonical dimension. Using this algorithm, we have been able to drastically reduce the complexity of the CNN without any performance degradation, or with a small controlled CDR reduction. Figure 16 shows the results achieved:



**Figure 4.6.2 Performance vs Complexity**

• The green point is the baseline configuration. This configuration is quite close to the configuration suggested in Reference [4]. It requires 53 MMACs per frame for an error rate of

0.76%.

• The second point from the left requires 1.47 million MACs per frame for an error rate of

1.03%, i.e., for an increase in the error rate of 0.27%, the MAC requirement has been reduced by a factor of 36.14.

• The leftmost point requires 0.61 MMACs per frame for achieving an error rate of 2.26%,

i.e., the number of MACs is reduced by a factor of 86.4 times.

• The points in blue are for a single-level CNN, whereas the points in red are for a hierarchical CNN. A best-case performance of 99.58% is achieved by the hierarchical CNN.

**CNNs on Tensilica Processors**

The Tensilica Vision P5 DSP is a high-performance, low-power DSP specifically designed for image and computer vision processing. The DSP has a VLIW architecture with SIMD support. It has five issue slots in an instruction word of up to 96 bits and can load up to

1024-bit words from memory every cycle. Internal registers and operation units range from 512 bits to 1536 bits, where the data is represented as 16, 32, or 64 slices of 8b, 16b, 24b, 32b, or

48b pixel data. The DSP addresses all the challenges for implementing CNNs in embedded systems as discussed in the previous section.

• Availability of high computational performance: In addition to the advanced support for implementing image signal processing, the DSP has instruction support for all stages of CNNs. For convolution operations, it has a very rich instruction set supporting multiply/multiply- accumulate operations supporting 8b x 8b, 8b x 16b and 16b x 16b operations for signed/unsigned data. It can perform up to 64 8b x 16b and 8b x 8b multiply/multiplyaccumulate operations in one cycle and 32 16b x 16b multiply/multiply-accumulate operations in one cycle. For max pooling and ReLU functionality, the DSP has instructions to do 64 8-bit comparisons in one cycle. For implementing non-linear functions with finite ranges like tanh and signum, it has instructions to implement a look-up table for 64 7-bit values in one cycle. In most of the cases, instructions for comparison and look-up table get scheduled in parallel with multiply/multiply-accumulate instructions and do not take any extra cycles.

• Larger load/store bandwidth: The DSP can perform up to two 512-bit load/store operations per cycle.

• Low dynamic power requirement: The DSP is a fixed-point machine. Due to the flexible handling of a variety of data types, full performance and energy advantage of mixed 16b and

8b computation can be achieved at minimal loss of accuracy. [www.cadence.com](http://www.cadence.com/) 10 Using

Convolutional Neural Networks for Image Recognition

• Flexibility: Since the DSP is a programmable processor, the system can be upgraded to a new version just by performing a firmware upgrade.

• Floating Point: For algorithms requiring an extended dynamic range for their data and/or coefficients, the DSP has an optional vector floating-point unit. The Vision P5 DSP is delivered with a complete set of software tools that includes a high-performance C/C++ compiler with automatic vectorization and scheduling to support the SIMD and VLIW architecture without the need to write assembly language. This comprehensive toolset also includes the linker, assembler, debugger, profiler, and graphical visualization tools. A comprehensive instruction set simulator (ISS) allows the designer to quickly simulate and evaluate performance.

When working with large systems or lengthy test vectors, the fast, functional TurboXim simulator option achieves speeds that are 40X to 80X faster than the ISS for efficient software development and functional verification. Cadence has implemented a single-layer architecture CNN on the DSP for German traffic sign recognition. Cadence has achieved a CDR of 99.403% with 16-bit quantization for data samples and 8-bit quantization for coefficients in all the layers for this architecture.

It has two convolution layers, three fully connected layers, four ReLU layers, three max pooling layers, and one tanh non-linear layer. Cadence has achieved a performance of 38.58

MACs/ cycle on an average for the complete network including the cycles for all the max pooling, tanh, and ReLU layers. Cadence has achieved best-case performance of 58.43 MACs per cycle for the third layer, including the cycles for tanh and ReLU functionalities. This DSP running at 600MHz can process more than 850 traffic signs in one second.

**4.7 The Future of CNNs:-**

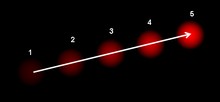
Among the promising areas of neural networks research are recurrent neural networks (RNNs) using long shortterm memory (LSTM). These areas are delivering the current state of the art in time-series recognition tasks like speech recognition and handwriting recognition.

RNN/auto encoders are also capable of generating handwriting/ speech/images with some known distribution. Deep belief networks, another promising type of network utilizing restricted Boltzmann machines (RMBs)/auto encoders, are capable of being trained greedily, one layer at a time, and hence are more easily trainable for very deep networks .

OpenCV's application areas include:

[**Egomotion**](https://en.wikipedia.org/wiki/Egomotion) **estimation**

In [robotics](https://en.wikipedia.org/wiki/Robotics) and [computer vision](https://en.wikipedia.org/wiki/Computer_vision), visual odometry is the process of determining the position and orientation of a robot by analyzing the associated camera images. It has been used in a wide variety of robotic applications, such as on the [Mars Exploration Rovers](https://en.wikipedia.org/wiki/Mars_Exploration_Rover)



**Figure 4.8 Th**[**e optic**](https://en.wikipedia.org/wiki/Optical_flow)**al flow vector of a moving object in a video sequence.**

**4.8 COMPARISON OF VARIOUS LANE DETECTION AND TRACKING ALGORITHMS**:

In this section, Lane detection and following algorithms are discussed. Presents a detailed analysis of various lane detection and lane tracking algorithms. also investigates the best lane detection and following algorithms that can be selected for a specific road condition. The method that offer it Y. U. Yim and S.-Y. Oh in 2003. Used in pre-processing sobel operator. And detection lane use hough transform and three feature vectors. Either in tracking use temporal predictor is used to predict current lane vector. Where Works fine for rainy and shady road. Either method that offer it S. Sehestedt, et al. in 2007. Used in pre-processing Inverse perspective mapping. And in detection lane use weak modelbased vector. Either in tracking use clustered particle filter. Where result is robust in difficult lighting conditions. The method that offer it M. Aly in 2008. Used in preprocessing Inverse perspective mapping, selective oriented gaussian filters. And in detection lane use hough transform and RANSAC spline fitting. Where result is comparable results to Algorithms using both detection and tracking. The method that offer it A. Borkar et al. in 2009. Used in preprocessing adaptive thresholding. And in lane detection use low resolution hough transform and matched filter. Where result is Robust in night time.

The method that offer It Y.-C. Leng And C.-L. Chen in 2010. used in preprocessing sobel operator. and in lane detection use heugh transform. where result is successful detection in worn-out road surface, signs, graphs, warning lines and image shaking. The method that offerG. Liu et al. in 2011. used in preprocessing Color, position and gradient Descriptors and Sobel operator. and in lane detection use Statistical transform. either in tracking use Partitioned particle filter Where result is Computationally expensive. The method that offer H.Jung et al. in 2013. used in preprocessing Steerable filter. and in lane detection use Haar like features. where result is Robust in illumination changes. The method that offer it V. S.,Bottazzi et al. in

2014 . used in preprocessing Histogram. and in lane detection use Segmentation. either in tracking use Lucas-Kanade tracking. Where result is Robust in illumination changes. the method that offer it Kumar S. in 2016. used in preprocessing binary mask image and applying range threshold. and in lane detection applying contrast analysis at pixel level intensity values. A set of points, available in an image frame, based lane feature models are used for detecting lanes on color image frame captured from video. Our proposed method was used in preprocessing divide RGB image in its bands and find histogram to each band we use image in blue band that represent road area. Either in lane detection use threshold Otsu’s method and Enhancement image by using function (bwareaopen) in MATLAB. and in following lane we use Find neighborhood for each pixel on the edge line, calculate theta to each pixel with its neighborhood

and Determine direction of road using theta. Where result is robust and efficient method for tracking road that used in driving car in special condition.

**CODE :**

import cv2 as cv

def do\_canny(frame):

# Converts frame to grayscale because we only need the luminance channel for detecting edges - less computationally expensive

gray = cv.cvtColor(frame, cv.COLOR\_RGB2GRAY)

# Applies a 5x5 gaussian blur with deviation of 0 to frame - not mandatory since Canny will do this for us

blur = cv.GaussianBlur(gray, (5, 5), 0)

# Applies Canny edge detector with minVal of 50 and maxVal of 150 canny = cv.Canny(blur, 50, 150)

return canny import numpy as np

import matplotlib.pyplot as plt

# def do\_canny(frame):

# gray = cv.cvtColor(frame, cv.COLOR\_RGB2GRAY)

# blur = cv.GaussianBlur(gray, (5, 5), 0)

# canny = cv.Canny(blur, 50, 150)

# return canny

def do\_segment(frame):

# Since an image is a multi-directional array containing the relative intensities of each pixel in the image, we can use frame.shape to return a tuple: [number of rows, number of columns, number of channels] of the dimensions of the frame

# frame.shape[0] give us the number of rows of pixels the frame has. Since height begins from 0 at the top, the y-coordinate of the bottom of the frame is its height

height = frame.shape[0]

# Creates a triangular polygon for the mask defined by three (x, y) coordinates polygons = np.array([

[(0, height), (800, height), (380, 290)]

])

# Creates an image filled with zero intensities with the same dimensions as the frame mask = np.zeros\_like(frame)

# Allows the mask to be filled with values of 1 and the other areas to be filled with values of 0

cv.fillPoly(mask, polygons, 255)

# A bitwise and operation between the mask and frame keeps only the triangular area of the frame

segment = cv.bitwise\_and(frame, mask)

return segment

# cap = cv.VideoCapture("input.mp4")

# while (cap.isOpened()):

# ret, frame = cap.read()

# canny = do\_canny(frame)

# First, visualize the frame to figure out the three coordinates defining the triangular mask plt.imshow(frame)

plt.show()

segment = do\_segment(canny)

def calculate\_lines(frame, lines):

# Empty arrays to store the coordinates of the left and right lines left = []

right = []

# Loops through every detected line for line in lines:

# Reshapes line from 2D array to 1D array x1, y1, x2, y2 = line.reshape(4)

# Fits a linear polynomial to the x and y coordinates and returns a vector of coefficients which describe the slope and y-intercept

parameters = np.polyfit((x1, x2), (y1, y2), 1)

slope = parameters[0]

y\_intercept = parameters[1]

# If slope is negative, the line is to the left of the lane, and otherwise, the line is to the right of the lane

if slope < 0:

left.append((slope, y\_intercept))

else:

right.append((slope, y\_intercept))

# Averages out all the values for left and right into a single slope and y-intercept value for each line

left\_avg = np.average(left, axis = 0)

right\_avg = np.average(right, axis = 0)

# Calculates the x1, y1, x2, y2 coordinates for the left and right lines left\_line = calculate\_coordinates(frame, left\_avg)

right\_line = calculate\_coordinates(frame, right\_avg)

return np.array([left\_line, right\_line])

def calculate\_coordinates(frame, parameters):

slope, intercept = parameters

# Sets initial y-coordinate as height from top down (bottom of the frame)

y1 = frame.shape[0]

# Sets final y-coordinate as 150 above the bottom of the frame y2 = int(y1 - 150)

# Sets initial x-coordinate as (y1 - b) / m since y1 = mx1 + b x1 = int((y1 - intercept) / slope)

# Sets final x-coordinate as (y2 - b) / m since y2 = mx2 + b x2 = int((y2 - intercept) / slope)

return np.array([x1, y1, x2, y2])

def visualize\_lines(frame, lines):

# Creates an image filled with zero intensities with the same dimensions as the frame lines\_visualize = np.zeros\_like(frame)

# Checks if any lines are detected if lines is not None:

for x1, y1, x2, y2 in lines:

# Draws lines between two coordinates with green color and 5 thickness

cv.line(lines\_visualize, (x1, y1), (x2, y2), (0, 255, 0), 5)

return lines\_visualize

# cap = cv.VideoCapture("input.mp4")

# while (cap.isOpened()):

# ret, frame = cap.read()

# canny = do\_canny(frame)

# # plt.imshow(frame)

# # plt.show()

# segment = do\_segment(canny)

# hough = cv.HoughLinesP(segment, 2, np.pi / 180, 100, np.array([]), minLineLength =

100, maxLineGap = 50)

# Averages multiple detected lines from hough into one line for left border of lane and one line for right border of lane

lines = calculate\_lines(frame, hough)

# Visualizes the lines

lines\_visualize = visualize\_lines(frame, lines)

# Overlays lines on frame by taking their weighted sums and adding an arbitrary scalar value of 1 as the gamma argument

output = cv.addWeighted(frame, 0.9, lines\_visualize, 1, 1)

# Opens a new window and displays the output frame cv.imshow("output", output)

# cap = cv.VideoCapture("input.mp4")

# while (cap.isOpened()):

# ret, frame = cap.read()

canny = do\_canny(frame)

# The video feed is read in as a VideoCapture object cap = cv.VideoCapture("input.mp4")

while (cap.isOpened()):

# ret = a boolean return value from getting the frame, frame = the current frame being projected in the video

ret, frame = cap.read()

# Frames are read by intervals of 10 milliseconds. The programs breaks out of the while loop when the user presses the 'q' key

if cv.waitKey(10) & 0xFF == ord('q'):

break

# The following frees up resources and closes all windows cap.release()

cv.destroyAllWindows()

**CHAPTER 5**

**RESULT AND CONCLUSION**

Figure [15](https://www.hindawi.com/journals/am/2018/8320207/fig15/) shows the preprocessing of four frames of images. Frame (a.i) and frame (b.i) are processed by basic preprocessing (without white feature extraction), and frame (a.ii) and frame (b.ii) are processed by the proposed preprocessing (with the white feature extraction). From the Figure [15](https://www.hindawi.com/journals/am/2018/8320207/fig15/) we can see that frame (a.ii) and frame (b.ii) which are processed by the proposed preprocessing can display the lane line. But there is a large amount of white residue in frame (a.i) and frame (b.i), and it is difficult to detect lane lines. Therefore, the basic preprocessing of the frame does not work well for lane detection. In view of these, we propose to add HSV colour conversion in the preprocessing stage and then extract the white features of the frame before the blurry ones, so as to achieve a better detection effect and improve the detection accuracy.

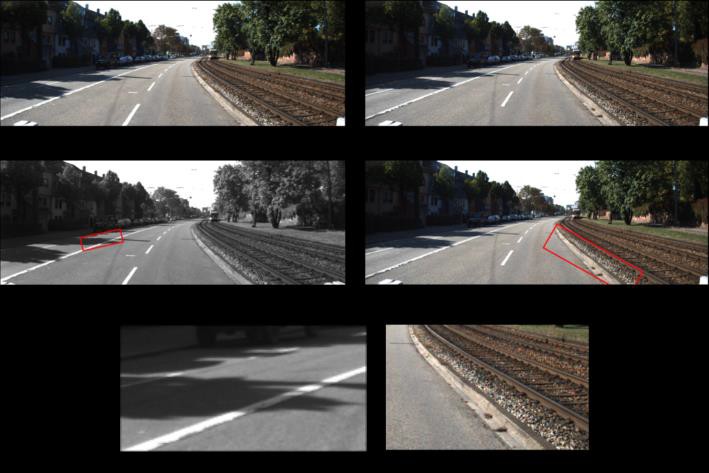




**Figure 5.1 HSV colour conversion**

Most research scholars directly perform ROI selection on the original image. In this paper, a new ROI selection method is proposed. Experiments show that the proposed ROI selection can improve the accuracy and efficiency of lane detection.

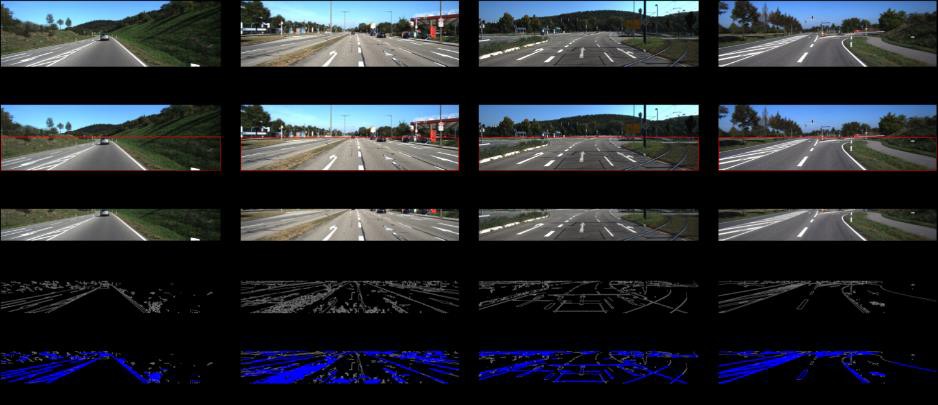
Figures 17 and 18 show the ROI selection of white feature. It can be seen from the figures that ROI selection of white feature cannot accurately detect the area of lane line, which will eventually produce a great error.





**Figure 5.2 ROI selection**

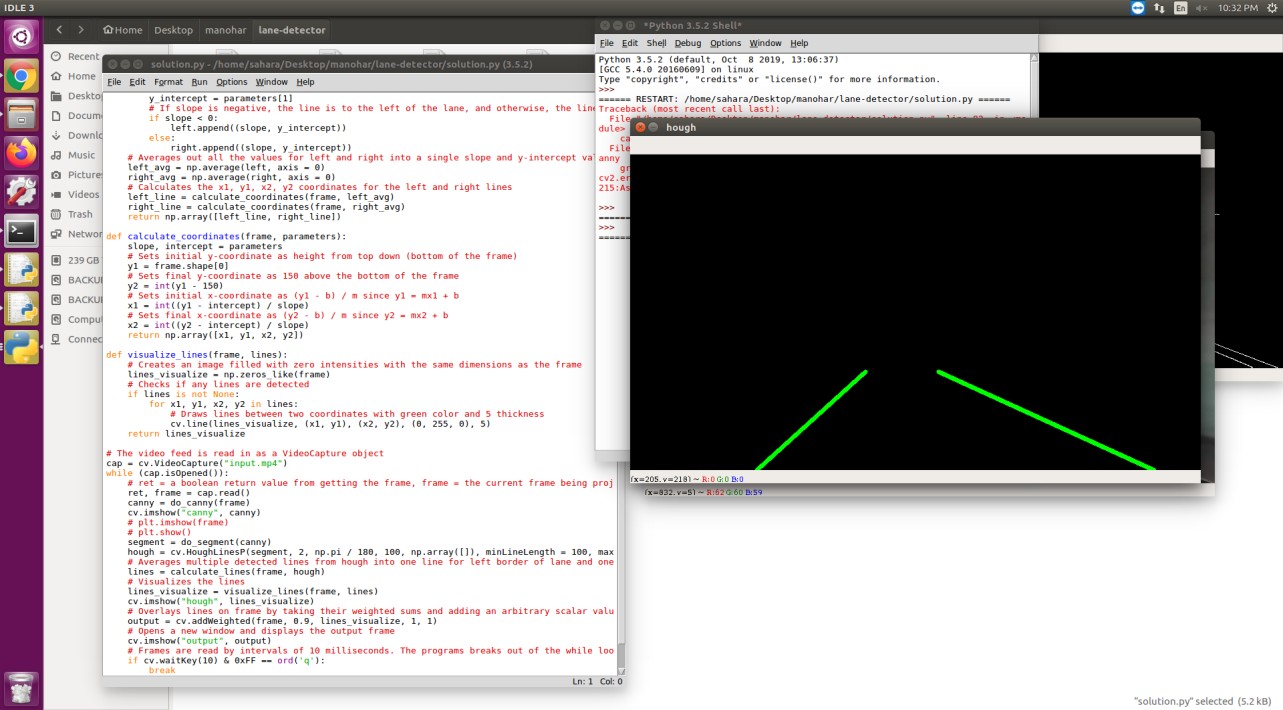
Half of the input frames are proposed as ROI selection. As shown in Figure [19,](https://www.hindawi.com/journals/am/2018/8320207/fig19/) the ROI selection implemented on the original image is followed by edge detection and lane detection on the selected ROI area. Compared with Figure [12](https://www.hindawi.com/journals/am/2018/8320207/fig12/), the result of the final lane detection contains many nonlane areas, and the effect of the lane detection is poor. The more the lane parameters are marked, the less efficient the calculation is. Therefore, the proposed method in this paper can lower the number of lane parameters, thereby reducing the calculation time and improving the detection efficiency.

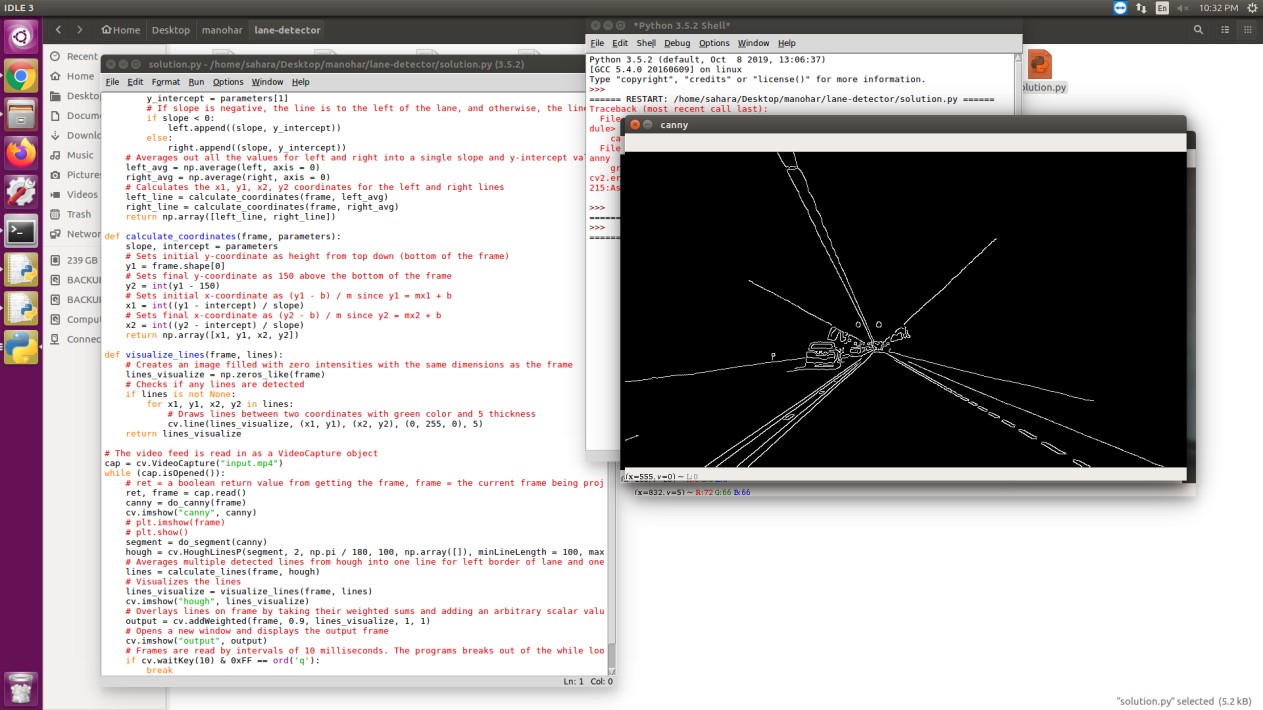


**Figure 5.3 Edge and Lane detection**

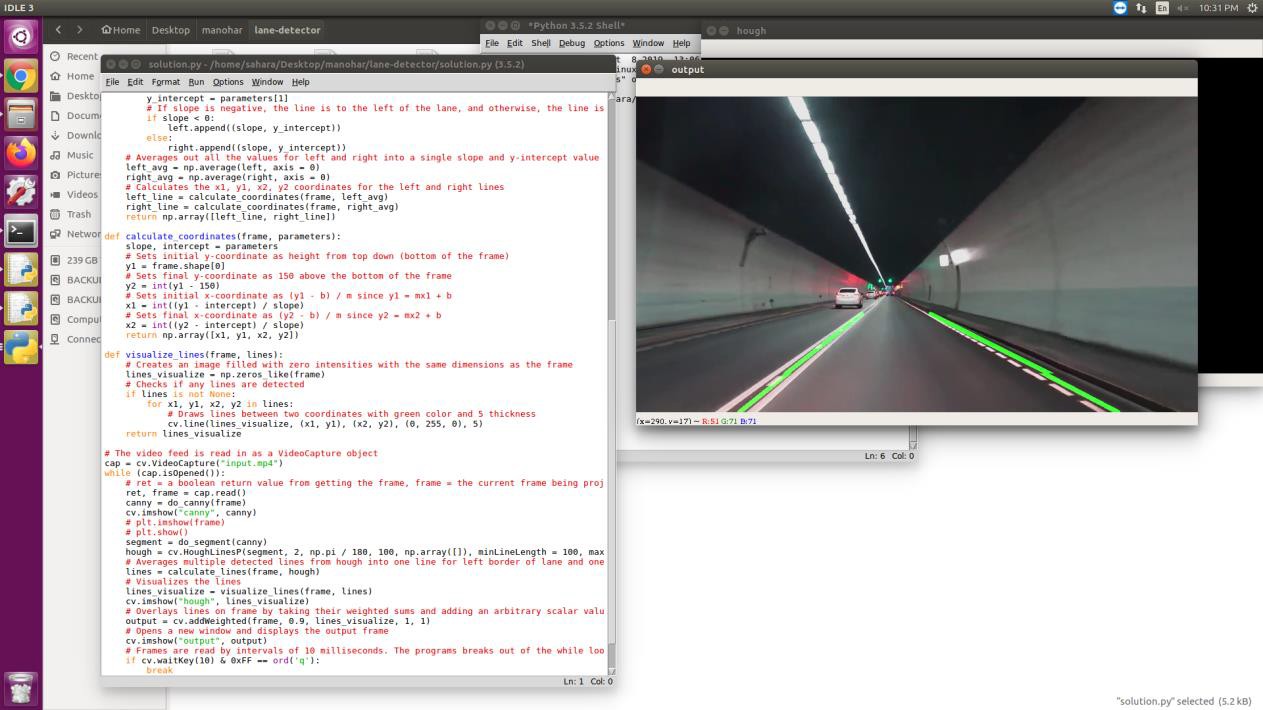
To quantify the accuracy of lane detection, we used the correct detection rate to evaluate the performance of our proposed method for lane detection under the data set used. For better results of the proposed method, we first set the size of the image in the data set to the same size and randomly take 300, 500, 800, 1000, and 1500 images as a test set in training sets. In order to verify the excellence of our proposed method, as shown in Figure [20,](https://www.hindawi.com/journals/am/2018/8320207/fig20/) we compared the detection efficiency of the basic preprocessing method for lane detection with the detection efficiency of the proposed preprocessing method. Moreover, as shown in Figure [21,](https://www.hindawi.com/journals/am/2018/8320207/fig21/) we also compare the lane detection efficiency of the lane detection method that selects the ROI area only based on the lane colour with the lane detection efficiency of the proposed ROI selection method. From Figures 20 and 21, we can see that the results of the proposed method achieve the highest correct detection rate to prove the effectiveness of our proposed method.

**Figure 5.4 Hough transform**





**Figure 5.5 Canny edge detection**



**Figure 5.6 Output**

**CONCLUSION**

In this paper, we proposed a new lane detection preprocessing and ROI selection methods to design a lane detection system. The main idea is to add white extraction before the conventional basic preprocessing. Edge extraction has also been added during the preprocessing stage to improve lane detection accuracy. We also placed the ROI selection after the proposed preprocessing. Compared with selecting the ROI in the original image, it reduced the nonlane parameters and improved the accuracy of lane detection. Currently, we only use the Hough transform to detect straight lane and EKF to track lane and do not develop advanced lane detection methods. In the future, we will exploit a more advanced lane detection approach to improve the performance.

**Data Availability**

The proposed lane detection data used to support the findings of this study are available from the corresponding author upon request.Different techniques like preprocessing, thresholding, perspective transform are fused together in the proposed lane detection system. Gradient and HLS thresholding detect the lane line in binary images efficiently. Sliding window search is used to recognize the left and right lane on the road. The cropping technique worked only the particular region that consists of the lane lines. From the experimental results, it can be

concluded that the system detects the lanes efficiently with any conditions of the environment. The system can be applied to any road having well-marked lines and implemented to the embedded system for the assistance of Advanced Driver Assistance Systems and the visually impaired people for navigation to keep them in proper track. In future, a real-time system with hardware implementation will be developed that will capture the images from the real-time scenario and detect the lanes based on the proposed technique as well as generate a warning for the concerned persons (drivers or visually impaired people)