AUTONOMOUS VEHICLE

A PROJECT REPORT

Submitted by

VAIKUNTH GURUSWAMY - 2019504602

TARUN U - 2019504597

ENESH NAREN A - 2019504517

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

MADRAS INSTITUTE OF TECHNOLOGY

ANNA UNIVERSITY: CHENNAI 600 044

May 2023

ANNA UNIVERSITY: CHENNAI 600 044

BONAFIDE CERTIFICATE

Certified that this project report **"AUTONOMOUS VEHICLE"** is the bonafide work of **VAIKUNTH GURUSWAMY (2019504602), TARUN U (2019504597), ENESH NAREN A (2019504517)** who carried out the work under my supervision. Certified further that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on earlier occasion on this or any other candidate.

Dr. P. INDUMATHI
Professor and Head of the Department
Department of Electronics Engineering
Madras Institute of Technology
Anna University
Chennai- 600 044

Dr. V. SATHIESH KUMAR
Assistant Professor
Department of Electronics Engineering
Madras Institute of Technology
Anna University
Chennai- 600 044

ACKNOWLEDGEMENT

We would like to record our sincere thanks to **Dr. J PRAKASH**, Dean, Madras Institute of Technology, Anna University for having given consent to carry out the project work.

We wish to express our sincere gratitude to **Dr. P INDUMATHI**, Professor and Head, Department of Electronics Engineering, M.I.T campus, Anna University.

We wish to express our gratitude and to our guide **Dr. V SATHIESH KUMAR**, Assistant Professor, Department of Electronics Engineering, for giving us the opportunity to undertake this project, for his valuable guidance, advice, motivation, encouragement and support during the situations of problems we faced while doing the project. We gained a great amount of knowledge during the course of the project with his extensive vision, creative thinking and ideas have been a source of inspiration. We take immense pleasure in thanking him for his continuous guidance throughout this project.

We wish to extend our sincere thanks to the review panel faculty members **Dr. S P JOY VASANTHA RANI** Professor and **Dr. K.MARIAMMAL** Associate Professor of Department of Electronics Engineering, for their valuable suggestions during all the reviews which took our project to greater heights.

We would like to thank our project coordinators, all the teaching and non-teaching staff members of the Department of Electronics Engineering, for their support in all aspects.

VAIKUNTH GURUSWAMY - 2019504602
TARUN U – 2019504597
ENESH NAREN A - 2019504517

ABSTRACT

The objective of this project is to develop a decentralized autonomous vehicle that enhances traffic cooperation and safety through sensor data sharing, without relying on a centralized server. Secure protocols and behavioral guidelines will be established for autonomous navigation. The aim is to achieve level 4 vehicle automation, enabling vehicles to detect their surroundings and make informed decisions, thereby advancing the automation levels and ensuring secure navigation. This thesis describes the entire process of its development from hardware requirements and implementation. The project concentrates on developing a rural road-based vehicle automation using python language and feasible hardware components. Our implementation features an improved steering mechanism inspired by the conventional Ackermann steering mechanism. To enhance accuracy, we employ bagging and boosting techniques with advanced neural network algorithms such as You Only Look Once -v3, Spatial Convolutional Neural Network, Support Vector Machine , and Deep Q-Network Algorithm, achieving an impressive 97.9% accuracy rate. Furthermore, by incorporating a thread-based parallel processing approach, we significantly improve processing time, resulting in a remarkable reflex time of 1ms and a frame rate of 0.25s. These advancements contribute to the successful deployment of a robust UGV by ensuring precise steering control and efficient navigation.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATION

NOTATION	Expansion
CAG	Compound Annual Growth rate
FSD	Full Self-Driving
UNECE	United Nations Economic Commission for Europe
ISO	International Organization for Standardization
BLE	Bluetooth Low Energy
PoE	Power over Ethernet
LAN	Local Area Network
SD	Secure Digital
USB	Universal Serial Bus
HDMI	High-Definition Multimedia Interface
IEEE	Institute of Electrical and Electronics Engineers
GPIO	General-Purpose Input/Output
MIPI	Mobile Industry Processor Interface
DSI	Display Serial Interface
CSI	Camera Serial Interface
GPS	Global Positioning System
PC	Personal Computer
UART	Universal Asynchronous Receiver-Transmitter
SPI	Serial Peripheral Interface
SBAS	Satellite-Based Augmentation Systems
WAAS	Wide Area Augmentation System
EGNOS	European Geostationary Navigation Overlay Service
MSAS	MTSAT Satellite Augmentation System
MTSAT	Multifunctional Transport Satellites

GAGAN	GPS Aided Geo Augmented Navigation
EEPROM	Electrically Erasable Programmable Read-Only Memory
HD	High Definition
ML	Machine Learning
API	Application Programming Interface
URL	Uniform Resource Locator
OSM	Open Street Map
ORS	Open Route Service
YOLO	You Only Look Once
CUDA	Compute Unified Device Architecture
CPU	Central Processing Unit
GPU	Graphics Processing Unit
BGR	Blue Green Red Format
HSV	Hue Saturation Value
SCNN	Spatial Convolutional Neural Network
SVM	Support Vector Machine
DQNA	Deep Q-Learning Network Architecture
UGV	Unmanned Ground Vehicle

CHAPTER 1
INTRODUCTION

1.1 ABOUT AUTONOMOUS VEHICLE

A fully functional self-driving car has no human involvement and can sense the surroundings without any human interactions. A recent study from JILA, a joint research centre between the Massachusetts Institute of Technology and the National Institute of Standards and Technology (NIST), shows how software can provide braking assistance in vehicles in the front of a self-driving car. To take this technology to the real world, there are currently several companies developing autonomous vehicles, such as Mercedes-Benz, Tesla, Toyota, Audi, Volvo and Google[1][2].The interest in self-driving cars from manufacturers stems from their value in reducing the number of accidents in all kinds of vehicles.

According to the National Highway Traffic Safety Administration, about 32 per cent of all fatal car accidents are due to distracted driving. According to “Critical Reasons for Crashes Investigated in the National Motor Vehicle Crash Causation Survey.” : "94 per cent of critical road accidents occur due to human error, the potential of automated vehicle technologies to reduce accidents on roads demands to take necessary actions”. Self-Driving vehicles have the potential to reduce deaths and injuries from car crashes, particularly those that result from driver distraction.[7]

Autonomous vehicles have been researched since 1920, they were first called ‘phantom autos’. These cars were impressive technological breakthroughs for the time, being remotely controlled through Morse keys and a radio signal. People saw these innovations as the future of driving safety. The most popular nowadays include Tesla, Waymo, Zoox and Aptiv.[3][7]

Tesla’s AI includes some mechanics: An FSD chip powers automated driving and focuses on performance optimization and redundancy. The Dojo Systems powered by a Dojo Chip contains high-power cooling and decision-making. Along with these intricate capabilities, Tesla’s ultrasonic and radar sensors contribute significantly to the vehicle’s safety, predicting how far objects are and the distance between other cars.

The Worldwide Autonomous Car Market's value in 2021 was worth USD 25.14 billion, and by 2030 will reach USD 196.97 billion at a 25.7% CAGR(Compound annual growth rate)[4]

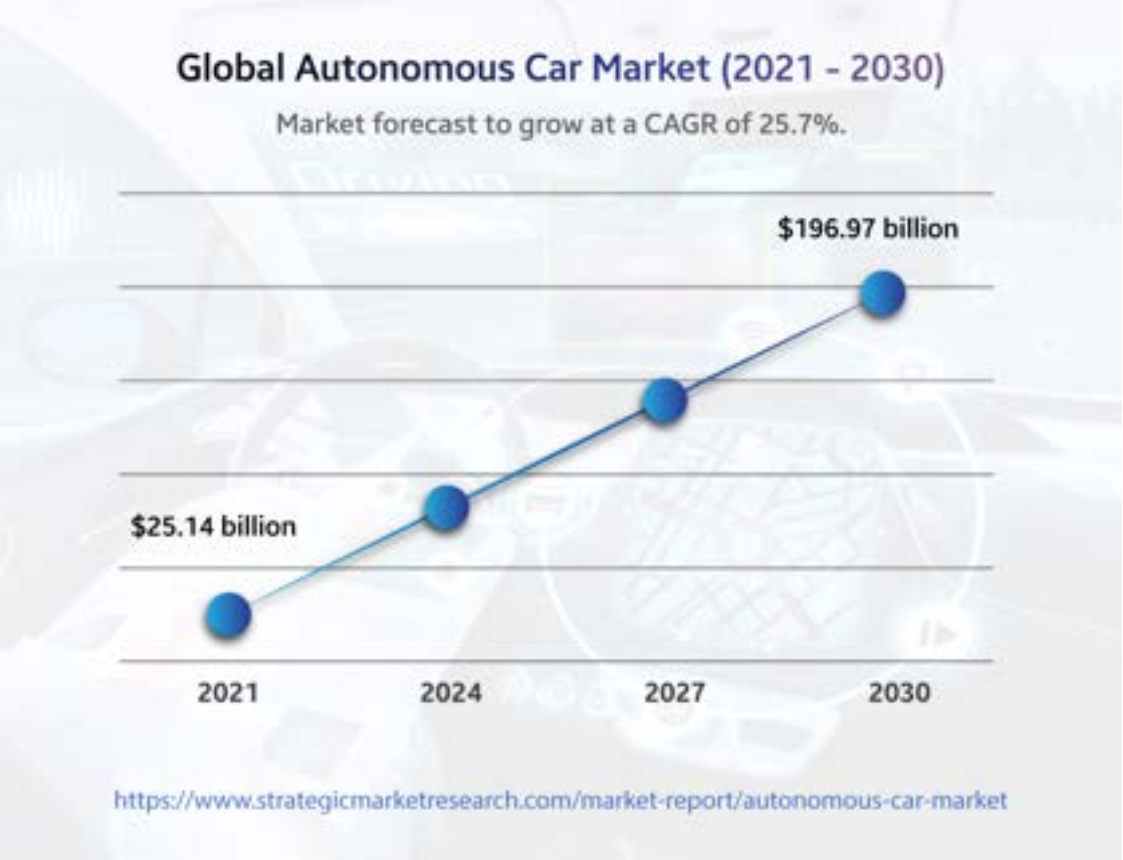


Fig 1.1 Representation of global autonomous car market[4]

1.2 CURRENT TECHNOLOGIES IN AUTONOMOUS VEHICLES

Inculcation of technology in vehicles will be fully automated in all safety-critical situations[5]. It will also offer improved driving performance, increased travel efficiency, and fewer accidents and will allow us to have a better travel experience. But the importance of this technology should not be underestimated. Autonomous vehicles can make transportation safer for everyone[5].

A vehicle that is fully autonomous is safer because it can sense the surroundings without any human interactions. It can then communicate this information to the self-driving control systems and enable them to drive safely and in an optimal way. A vehicle that is fully autonomous can monitor and detect hazards in its environment in order to avoid them and to act accordingly

Technologies used in autonomous vehicles are already able to see the environment and react to its environment. These technologies include laser sensors, radars, cameras, sensors and more. The technology also needs to be integrated into the self-driving vehicle, as otherwise, it will not have the necessary stability and self-driving capabilities.

Autonomous control systems are the technology-based support systems that are activated in situations when a driver needs to take over a safety-critical situation. The driver is able to take over safety-critical situations and is equipped to handle any safety-critical situations.[3]

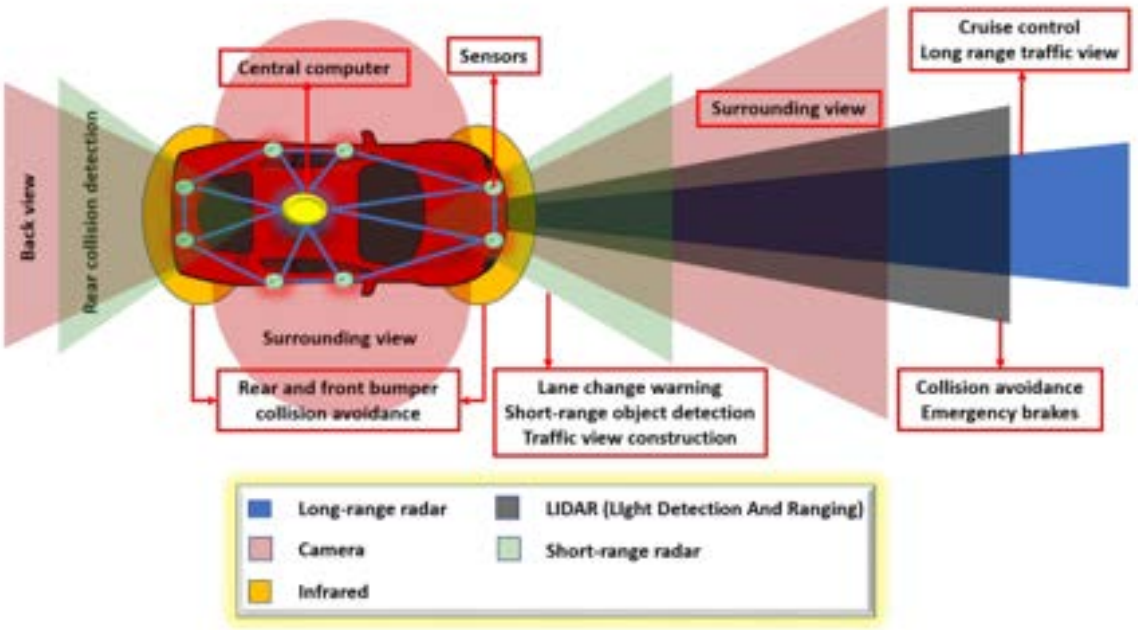


Fig 1.2 Sensor perspective of vehicle[6]

Autonomous vehicles can analyze and develop a map of their surroundings based on various sensors situated in different parts of the vehicle. The video cameras track other vehicles, read road signs, detect traffic lights, and look for pedestrians. Lidar (light detection and ranging) sensors bounce pulses of light off the vehicle's surroundings to detect road edges, measure distances, and identify lane markings. Ultrasonic sensors situated on the wheels help in detecting curbs and other vehicles when parking.[6]

Sophisticated software then processes all these sensory inputs, plots a path-way, and sends instructions to the actuators, which control the acceleration, braking, and steering of the vehicle. Hard-coded rules, predictive modelling, obstacle avoidance algorithms, and object recognition help the software follow traffic rules and navigate the obstacles.

1.3 LEVELS OF AUTONOMOUS VEHICLES

SAE International (**Society of Automotive Engineers)**, is a United States-based, globally active association and standards-developing organization for engineering professionals in industries. SAE developed a classification system that defines the degree of driving automation of a car. Ranging from levels zero to five, the driving automation spectrum begins with vehicles without this technology and ends with entirely self-driving vehicles.

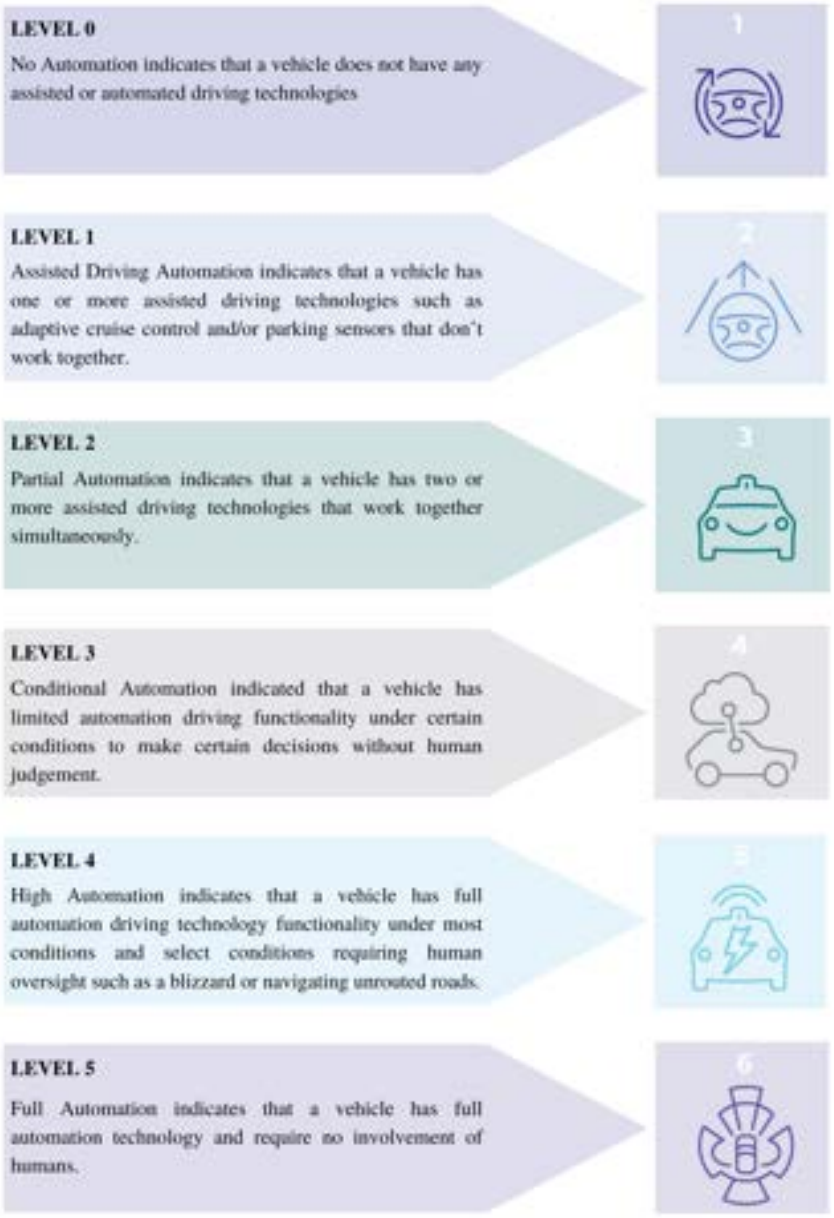


Fig 1.3 Levels of Autonomous Vehicle

1.4 SAFETY AND SECURITY

Many autonomous driving technologies are emerging, ranging from advanced and highly functional systems to versatile systems with limited features and functions such as vehicle-to-vehicle communication[34]. The advent of these technologies has spurred the development of safety and security issues.

Current technical and ethical challenges in self-driving vehicle development include a collaborative approach to ensuring and protecting human passengers and self-driving vehicles, as well as an appropriate approach. to detect and prevent malicious activity. Technology development is necessary. Self-driving vehicles are believed to be more vulnerable to security threats than human-driven vehicles due to the absence of human oversight of the vehicle's security mechanisms.

Studies show that the main risk for advanced and secure vehicles is the failure of the software that controls the automated functions and not enough information from the safety functions to analyze and interpret the information. generated by the sensor system. , and it turned out to be misinterpreting the sensor data. Such accidents can endanger residents. Without a safety mechanism, the driver may be forced to steer the vehicle manually. In addition, the driver may become more of an obstacle than a safety device.

Standards are global in scope and come from multiple vetted sources, including the International Organization for Standardization, and the UNECE World Forum for Harmonization of Vehicle Regulations (WP.29)[8]

- ISO/SAE 21434, “Road Vehicles - Cybersecurity Engineering” mentions the engineering requirements for cybersecurity risk management highlighting the conception, production, operation, product development, maintenance, and

decommissioning of electrical and electronic (E/E) systems in road vehicles, including their components and interfaces.

- UN Regulations contain provisions for vehicles, their systems, parts, and equipment, related to safety and environmental aspects. They include performance-oriented test requirements, as well as administrative procedures. Specifically, UN R155 and UN R156 are a set of regulations that deal with Vehicle Cybersecurity. UN R155 deals with the general requirements for Vehicle Cybersecurity, while UN R156 deals with the specific requirements for heavy vehicles.

Fig 1.4 Global safety perspective of self driving car[8]

CHAPTER 2
LITERATURE SURVEY

This chapter discusses the existing work of the Autonomous vehicle's separate features and integration as a whole.

Rasheed Hussain and Sherali Zeadally [9] have presented a NavLab experimental analysis on Autonomous Land vehicles (ALV). Detailed descriptions of different modes of communication used in autonomous cars have been analyzed in detail. Further utilization and analysis of LIDAR (Light Detection and Ranging) based generated data and other RGB images as input and predict a three-dimensional (3D) representation of that data. Descriptions of Software complexities and manoeuvre motor controls have been represented in this paper.

Mehdi Masmoudi, Hakim Ghazali, Mounir Frikha, and Yehia Massoud [37] have presented an Inter-Vehicle Communication analysis and examined each of the vehicle movements to better identify the driving behaviour. Effective utilization of YOLO3 (You Only Look Once) to detect obstacles in various timestamps using vehicle vision has been studied. For navigation purposes, the Feature Pyramid Network algorithm and Q-learning-based Reinforcement learning algorithm have been developed.

Prof. Rujata Chaudhari, Prof. Vishal Badgujar, Hiral Thadeshwar, Vinit Shah and Mahek Jain [10] developed a Convolutional Neural Network with a Monocular vision algorithm. Further utilization of Haar cascade classifier using Convolutional Neural Network used for detecting signals. Implementation of ultrasonic sensors for front collision avoidance. A COCO dataset is used with a model of MobileNet v2 COCO quantized model. Used OpenCV for Lane line detection.

Irfan and Karunakar Pothuganti [38] designed and trained a neural network system in the field of a self-driving vehicle/autonomous vehicle/Driving Assistant system (DAS). detailed explanation about the utilization of deep convolutional neural networks and various image processing techniques for detecting real-time objects.

Further detecting lanes using a neural network was developed. The developed models have further been implemented on raspberry pi along with the camera module.

Abdur R. Fayjie, Sabir Hossain, Doukhi Oualid, and Deok-Jin Lee [11] developed a vehicle using Deep Reinforcement Learning in Urban environments. Implementation using sensors on two types of input sensor data: a vision-based camera sensor and a laser sensor. High-end optimization and detailed analysis using Hokuyo LIDAR sensor and camera sensors embedded with Nvidia Jetson TX2 embedded GPU. A deep neural network was developed based on fully connected CNN to approximate Deep reinforcement learning-based Q-function.

The network consists of 3 convolutional layers and 4 dense layers. Two types of data input have been implemented to increase vehicle accuracy one is front camera image and the other is LIDAR data (grid-map of size 80 x80) are fed into the neural network, processed using the convolutional-layers for each data type presented a reinforcement-learning based approach with Deep Q Network implemented in autonomous driving. Concatenation of LIDAR data and camera images is used to provide high accuracy. Another predominant feature was the deployment and training on custom-build environments created using Unity.

Qi-Chao Mao, Hongwei Sun, Yanbo Liu, and Rui-Sheng Jia [12] of Mini-YOLOv3, introduced a Multi-Scale Feature Pyramid Network with a U-shaped structure to improve the performance of multi-scale object detection. Compared with YOLOv3, Mini-YOLOv3 has a smaller model size and fewer trainable parameters and FLOPs. After evaluating Mini-YOLOv3 on the MS-COCO benchmark dataset, the parameter size was seen to be only 23% of YOLOv3's yet the detection accuracy was still comparable. Furthermore, Mini-YOLOv3 only required 1/2 of the detection time, with an mAP-50 of 52.1 at 67 fps.

A. Vidyavani , K. Dheeraj, M. Rama Mohan Reddy, KH. Naveen Kumar [13] presented an algorithm that makes predictions across 3 different scales, using feature maps of strides 32, 16 and 8 using YOLOv3. With an input size of 416 x 416, this means the scales are 13 x 13, 26 x 26, and 52 x 52. It also applies the strategy of regression to predict the bounding box scores, and the cross-entropy loss is used to predict the classes that the box may contain. This provides a multi-label classification for objects detected in images, and the average precision for small objects is higher than that of Faster RCNN.

Manjiri Bichkar, Suyasha Bobhate, and Prof. Sonal Chaudhari [14] demonstrate the use of deep learning methods for achieving accuracy in traffic sign recognition, even when there are disturbances present. This dataset consists of both grey and colour images and is divided into training and testing sets. To achieve this, efficient methods such as Convolutional Neural Networks and Support Vector Machines are used. Kolachalama Venkata Naga Sreya [15] has shown that a system using a skilled convolutional neural network (CNN), which incorporates crucial data of traffic signs and images, and assigns them to corresponding classes in parallel, works with great efficiency.

Shao, F., Wang, X., Meng, F., Rui, T., Wang, D., & Tang, J [16] converted the images of the road scene to grayscale and filtered them with optimized simplified Gabor wavelets (SGW). This step strengthened the edges of the traffic signs, which was beneficial for the following stage. Then, they used the maximally stable extremal regions algorithm to extract the region of interest and the support vector machine (SVM) to classify the superclass of traffic signs.

Our proposed work integrates features from all the aforementioned reference papers, utilizing their insights to enhance the comprehensiveness and effectiveness of our approach.

	Generating Map	Hardware Deployment	Traffic Sign Detection	Traffic Signal Detection	Rural Road Detection
Rasheed Hussain et al. [9]		✓			✓
Mehdi Masmoudi et al. [37]	✓		✓	✓	
Hiral Thadeshwar et al.[10]	✓	✓			✓
Irfan et al.[38]		✓	✓	✓	
Abdur R. Fayjie et al. [11]			✓	✓	✓
Qi-Chao Mao et al.,[12]			✓	✓	
A. Vidyavani et al.[13]			✓	✓	
Manjiri Bichkar et al.[14]			✓		
Shao, F., Wang, X et al.[16]			✓		
Proposed methodology	✓	✓	✓	✓	✓

Table 1.1 Literature Survey

CHAPTER 3
PROPOSED METHODOLOGY

Our project aims to develop a level 4-based autonomous vehicle that incorporates advanced features such as road sign detection, rural road detection, traffic light detection, and lane detection. The primary objective is to implement these features in real-time scenarios. The upcoming images provide visual representations of the hardware and software architectures utilized, while subsequent subsections outline the specific utilization of each component.

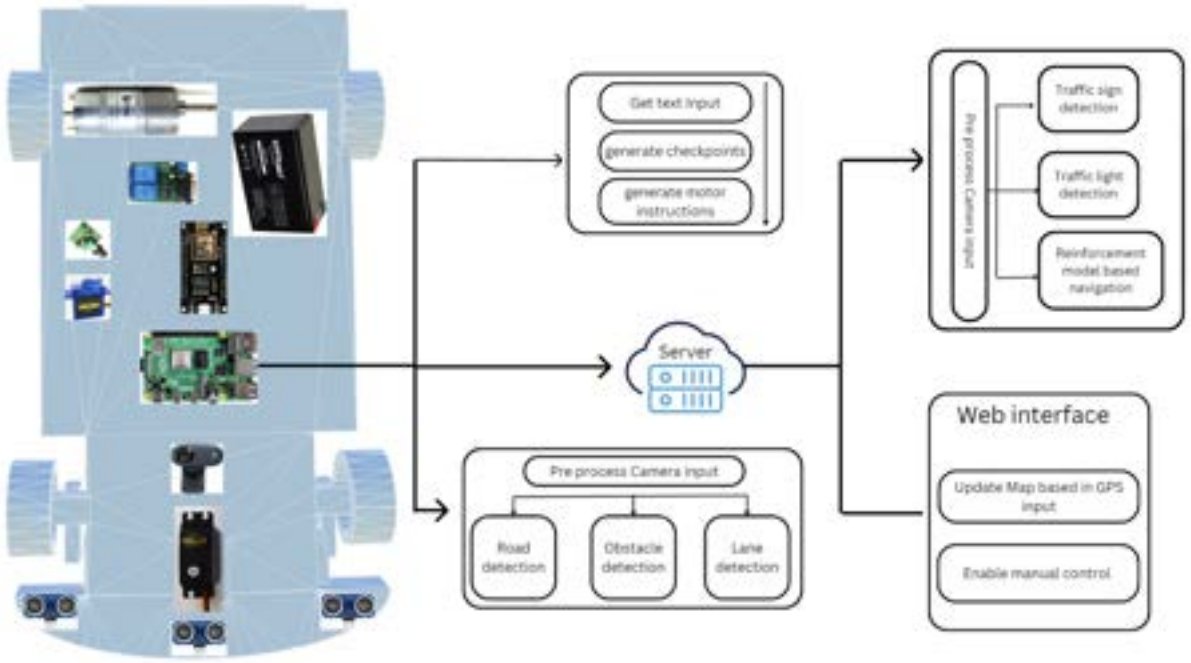


Fig 3.1 Complete hardware and software block diagram

3.1 GETTING USER COMMANDS

The project utilizes the SpeechRecognition library to enable user commands in the autonomous vehicle. User input, including destination and starting location, is extracted effectively. The SpeechRecognition library supports various engines and APIs for speech recognition, such as Google Speech Recognition. This functionality

involves acoustic modelling, converting speech into electrical energy, and then to text. Natural Language Processing and Neural Networks are employed, while Hidden Markov models can improve accuracy by identifying temporal patterns in speech. Relevant information, including the destination location and source destination (if provided), is extracted from the input.

Fig 3.2 Speech-to-text conversion

3.2 OBTAINING COORDINATES

In the project, obtaining coordinates is a crucial aspect. The user input is used to extract the corresponding latitude and longitude. Geolocation techniques are employed to obtain these details. Geolocation involves tracking a device's location using GPS, cell towers, or Wi-Fi hotspots. It utilizes a positioning system to track the user's location by latitude and longitude coordinates or a physical address. Additionally, the project optionally employs web data extraction and interpretation techniques to ensure accurate latitude and longitude values are obtained.

3.3 MAPS

Further finding the coordinates of the source and destination, marking the corresponding endpoints in a map and using the map to estimate travel path

Regarding the map application two layer of map is being implemented:

- Folium map
- Google Earth Engine map

3.3.1 FOLIUM MAPS

The project utilizes Folium, a Python library, to visualize geospatial data. Folium acts as a convenient Python wrapper for Leaflet.js, a popular JavaScript library used for creating interactive maps. By combining the capabilities of Leaflet.js with the simplicity of Python, Folium becomes a highly effective mapping tool. It places emphasis on simplicity, performance, and user-friendliness, providing an efficient and flexible solution with an API that is easy to use. With Folium, users can generate base maps of specific dimensions using either default tilesets or custom tileset URLs. It also supports a variety of tile layer views, such as Stamen Terrain, Stamen Toner, Stamen WaterColor, Cartodbpositron, and Cartobdark_matter, giving users the option to choose from different map styles based on their needs. Overall, Folium empowers users to visualize geospatial data in a straightforward and efficient manner.

Fig 3.3 OpenStreet map

The obtained points are plotted on the Folium map using Folium Marker, with the user's source location as the centre of focus. Besides the default tilesets, Folium also supports other custom-built and open-source tile layers. The purpose of using Folium maps is to display maps even in areas with poor network connectivity. As a lightweight solution, it can present less detailed maps with minimalistic navigation information.

3.3.2 GOOGLE EARTH ENGINE

Google Earth Engine is a powerful platform that combines an extensive catalogue of satellite imagery and geospatial datasets with advanced analysis capabilities. It enables scientists, researchers, and developers to detect changes, map trends, and quantify differences on Earth's surface. The platform offers tools for visualizing, manipulating, editing, and creating spatial data in a fast and user-friendly manner. It stores over 30 years of historical imagery and scientific data sets, constantly updated and expanded.

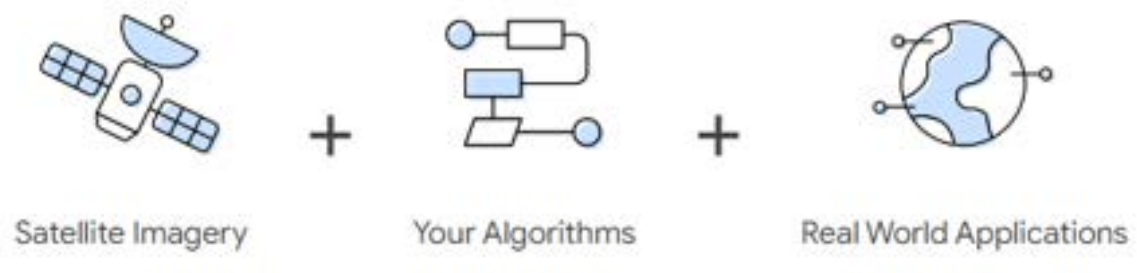


Fig 3.4 Working of Google Earth Engine

By integrating Google Earth Engine into a Python script and accessing the necessary API keys, users can leverage its advanced mapping capabilities. Google Earth Engine provides more detailed map previews compared to Folium but requires a high-speed network connection for proper functionality and content display. It can be used as the top layer in a map to showcase a more detailed environment.

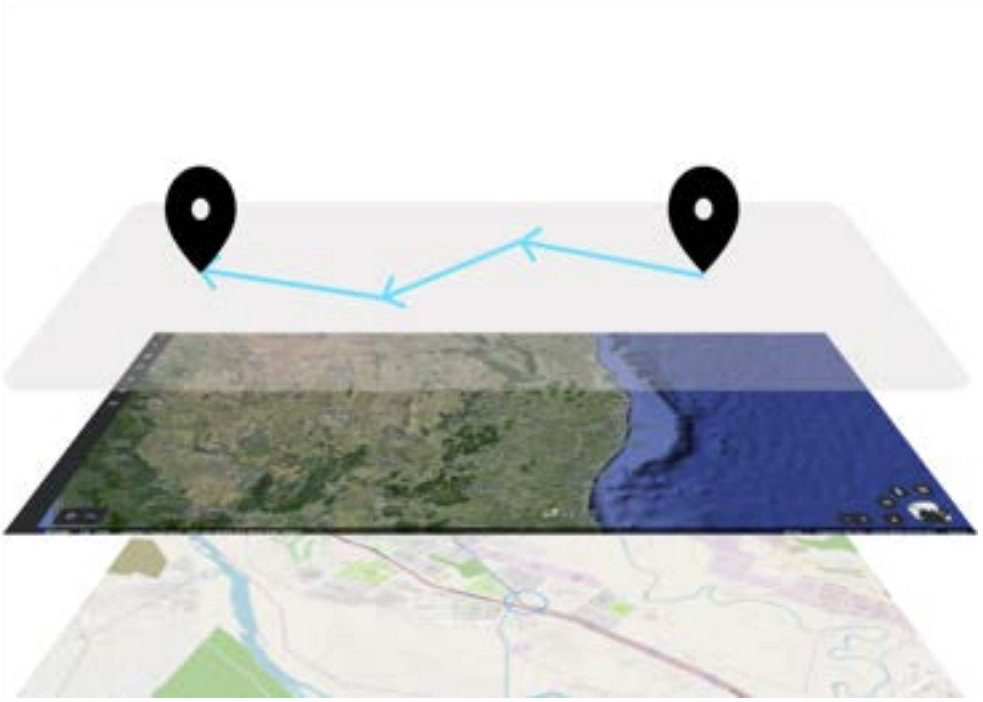


Fig 3.5 3 Layered maps

The bottom layer of the map will be displaying a folium map, and the intermediate layer will access the features of google earth engine maps on the top layer the source and destination markings and the travel path is plotted.

3.4 TRAVEL PATH

Based on the map and the available coordinates, the travel path needs to be extracted. For this, we are utilizing Openrouteservice for estimating travel paths. In this process, we are utilizing the Directions service which determines travel routes and navigation information according to diverse criteria.

On registration and creating a client session the API service will provide the needful path based upon the map and destination locations.

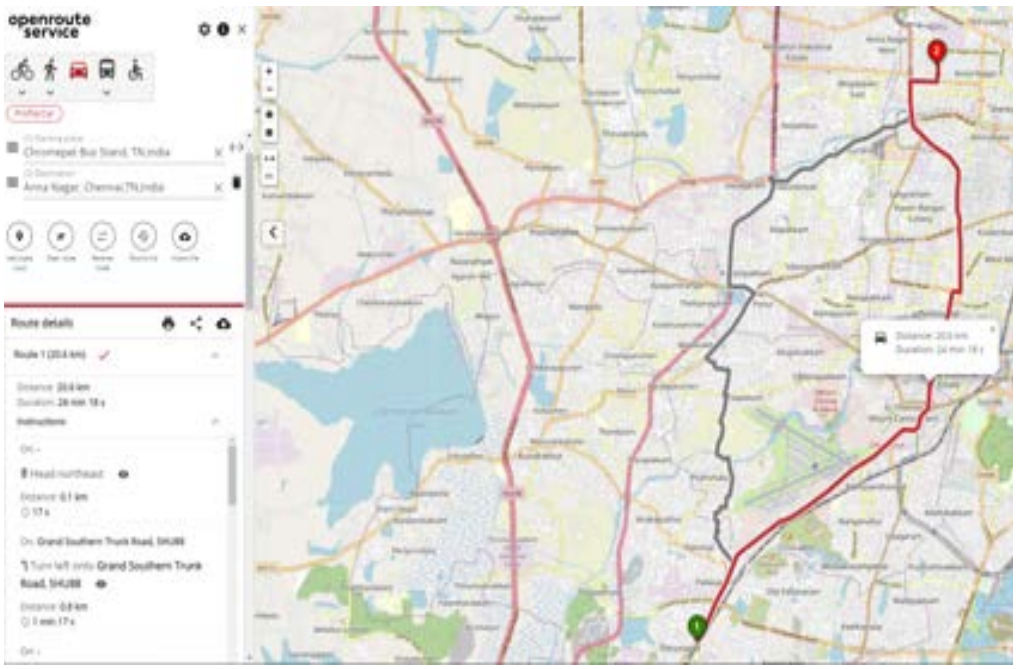


Fig 3.6 ORS preview

Details such as

- Total travel distance
- Estimated time of travel
- Travel route
- Instructions with it corresponding distance and time of travel
- Provide alternative routes to travel and establish multiple paths to reach destination

3.5 MOTOR INSTRUCTIONS

Motor instructions are feasible only by knowing the distance to be covered and time is taken to cover the estimated distance.

Intermediate points are determined and extracted from ORS using these points as checkpoints the distance between those points is determined

The Haversine formula is used to calculate the distance between two points. The haversine formula determines the great circle distance between two points on a sphere given longitude and latitude.[22]

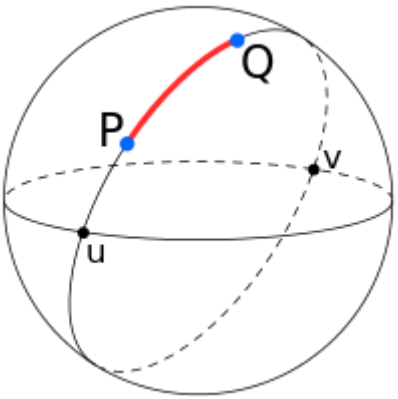


Fig 3.7 Haversine distance[22]

Central angle Haversine can be computed, between two points with r as the radius of the earth, d as the distance between two points, ϕ_1, ϕ_2 is the latitude of two points and λ_1, λ_2 is the longitude of two points respectively, as[22]:

$$haversine\ (d/r) = haversine\ (\varphi 2 - \varphi 1) + cos(\varphi 1)cos(\varphi 2)haversine(\lambda 2 - \lambda 1) \quad (3.1)$$

Haversine formula to calculate distance between two as:

$$a = \sin^2(\Delta\varphi/2) + \cos\varphi 1 \cdot \cos\varphi 2 \cdot \sin^2(\Delta\lambda/2) \quad (3.2)$$

$$c = 2 \cdot \operatorname{atan2}(\sqrt{a}, \sqrt{(1-a)}) \quad (3.3)$$

$$d = R.c \quad (3.4)$$

where,

$\Delta\varphi$ = lat1 – lat2 (difference of latitude)

$\Delta\lambda$ = lon1 – lon2 (difference of longitude)

R is the radius of earth i.e 6371 KM or 3961 miles

On determining the intermediate distance, slope or deviation required to move to the next checkpoint needs to be determined. For a line whose endpoints are (x1, y1) and (x2, y2), the slope of the line is given by the equation

$$m = (y_2 - y_1) / (x_2 - x_1) \quad (3.5)$$

The angle between the two lines can be found by calculating the slope of each line and then using them in the formula to determine the angle between two lines when the slope of each line is known from the equation

$$\tan\theta = \pm(m1-m2)/(1+m1 \ . \ m1) \quad (3.6)$$

Combining the instruction from ORS, distance found using haversine formula, and the angle of deviation is used to produce the final travel instruction in the format of {{motor instruction}} – {{distance to be covered}} – {{angle of deviation required}} Which will be passed on to the hardware device for further functionalities.

3.6 WEBSITE

For manual control a user-accessible feature is implemented through a web-app where basic motor controls are initially developed.

Controls such as:

- Front
- Back
- Stop
- Left
- Right

The development of this application is based on the Python Flask framework. By sending post requests to the website's route, users receive commands that are

frequently updated in real-time. The web application is hosted on the PythonAnywhere portal, providing a reliable platform for its operation. You can access the web app by visiting the following URL: **https://motorcontrol.pythonanywhere.com/**. Through this website, users can interact with the application and receive the latest dynamically updated commands[41]. The Python Flask framework enables the efficient handling of these requests and ensures a smooth user experience. With the hosting on PythonAnywhere, the web app remains accessible to users at all times, allowing them to benefit from its functionalities seamlessly.

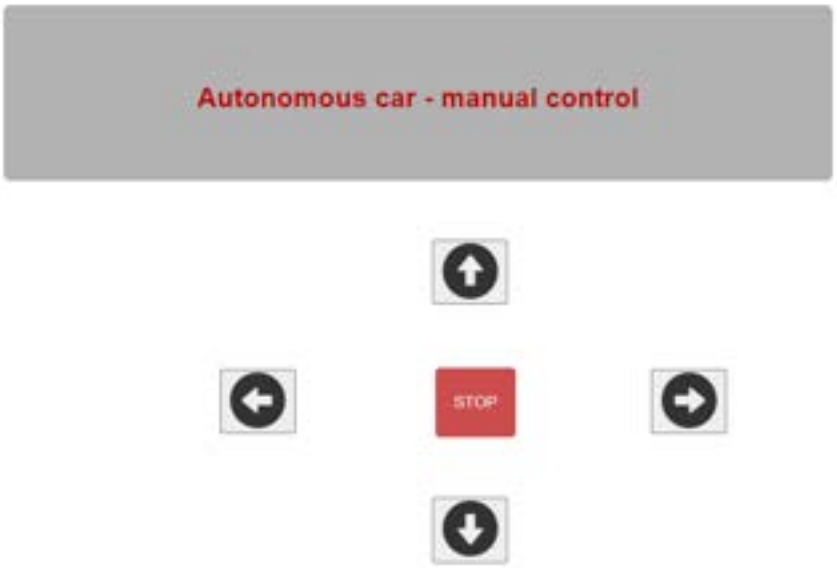


Fig 3.8 Website for Manual Control

To be aware of the vehicle's current position, accurate GPS location is required and should be updated to the user periodically. In Order to achieve that accurate location-sharing methods have been utilized to share the live location to the server and thereby resulting in the successful vehicle's corresponding latitude and longitude as well as using Google map API to present the map interface to the web interface.

Fig 3.9 Current Location

In addition to the location it is mandatory for the user at remote locations to get access to the vision of the vehicle. Based on the camera feed received from the raspberry pi to the server the feed input is further passed on to the web interface to enable this functionality. The communication is done by converting the image to bytes and transmitting it to the server.

Fig 3.10 Data from camera

3.7 ROAD SIGN DETECTION

The process of road sign detection consists of two sequential steps. The first step involves determining the location of the road sign within the vehicle's field of vision. The second step is to identify and understand the message or information conveyed by the detected road sign to the vehicle. By following this two-step approach, the system can effectively detect road signs, locate them accurately, and interpret their meanings for the vehicle's awareness and decision-making.

3.7.1 DETECTING LOCATION OF ROAD SIGN

The location of road signs is detected using the YOLO (You Only Look Once) object detection algorithm. The DarkNet framework, a fast and easy-to-install neural network framework written in C and CUDA, is used for accessing the YOLO model. YOLO-V3, based on ResNet and FPN architectures, features Darknet-53 with skip connections and 3 prediction heads. YOLO employs a direct location prediction technique by predicting offsets to anchor boxes, allowing diverse and shape-specific predictions.

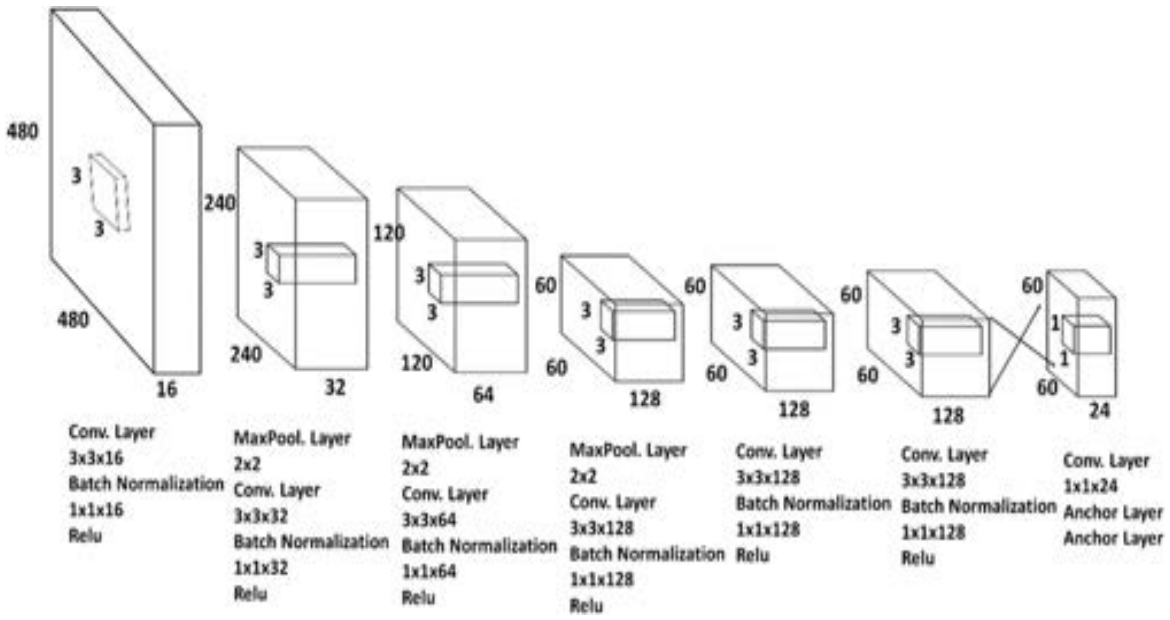


Fig 3.11 YOLO V3 architecture [48]

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Fig 3.12 YOLOV3 detailed visualization

3.7.2 ROAD SIGN CLASSIFICATION

After identifying the regions of road signs, a Convolutional Neural Network (CNN) model is employed to classify these road signs [30]. The implementation process utilizes a dataset consisting of traffic signs from 205 different classes [31]. The CNN model plays a crucial role in accurately categorizing the detected road signs based on their specific class, enabling effective recognition and understanding of various types of traffic signs.

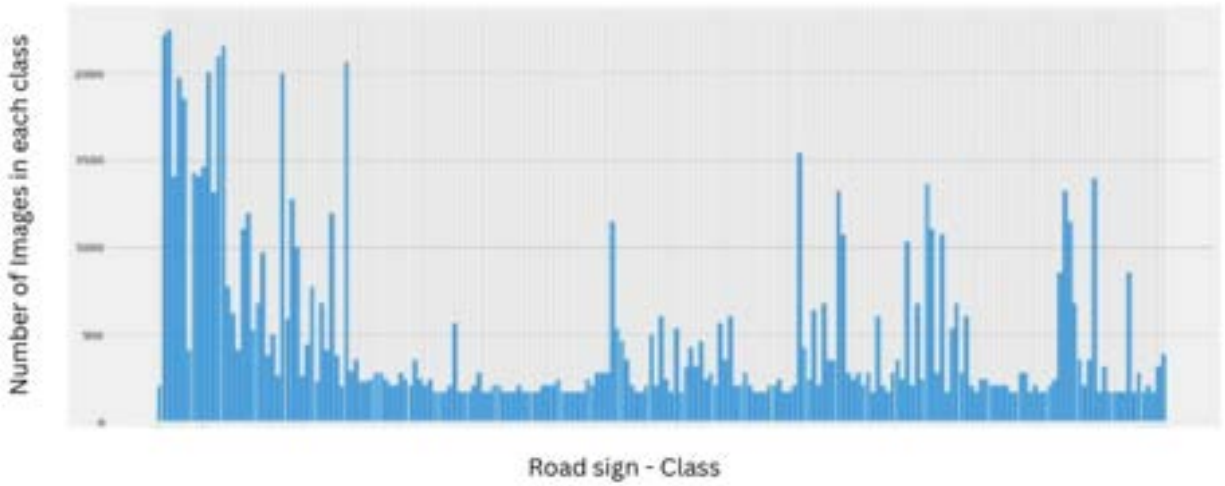
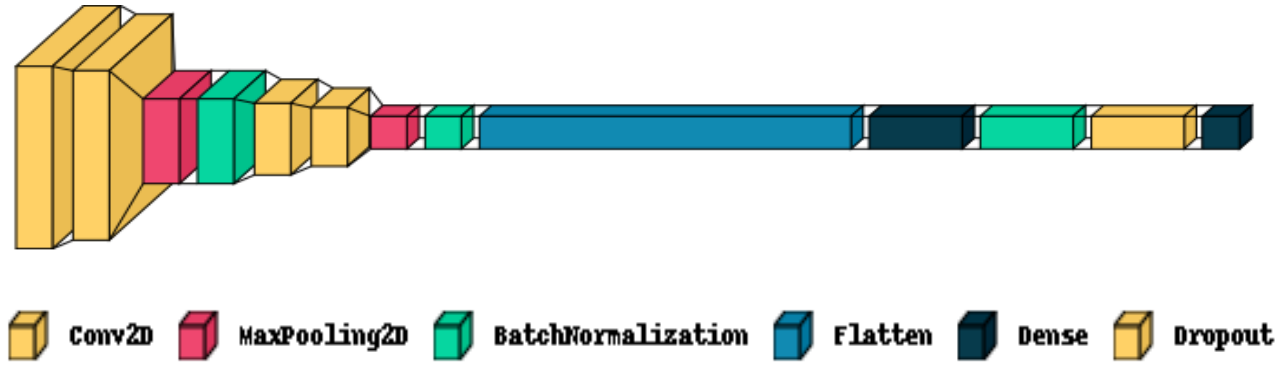


Fig 3.13 Distribution of the Dataset

Fig 3.14 Sample Perspective in the Dataset

General traffic sign images taken from varying perspectives and with noise . Total 97,426 images are being used to train the CNN model[16][33][32].

Using Tensorflow framework to build the neural network and perform further analysis.The dimensions of 30x30x3 is passed as the input to the model.




```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_4 (Conv2D)           (None, 28, 28, 16)        448

 conv2d_5 (Conv2D)           (None, 26, 26, 32)        4640

 max_pooling2d_2 (MaxPooling  (None, 13, 13, 32)       0
 2D)

 batch_normalization_3 (Batc  (None, 13, 13, 32)       128
 hNormalization)

 conv2d_6 (Conv2D)           (None, 11, 11, 64)        18496

 conv2d_7 (Conv2D)           (None, 9, 9, 128)         73856

 max_pooling2d_3 (MaxPooling  (None, 4, 4, 128)        0
 2D)

 batch_normalization_4 (Batc  (None, 4, 4, 128)        512
 hNormalization)

 flatten_1 (Flatten)         (None, 2048)              0
...
Total params: 1,254,381
Trainable params: 1,253,037
Non-trainable params: 1,344
```

Fig 3.15 Custom Model Architecture

On compiling and training the model it yielded the following growth in accuracy and drop with respect to loss

	Mean square error
Optimizer	Adam
Metrics	accuracy
Monitor	‘val-loss’
Epochs	100

Table 3.1 Road Sign detection Hyper parameters

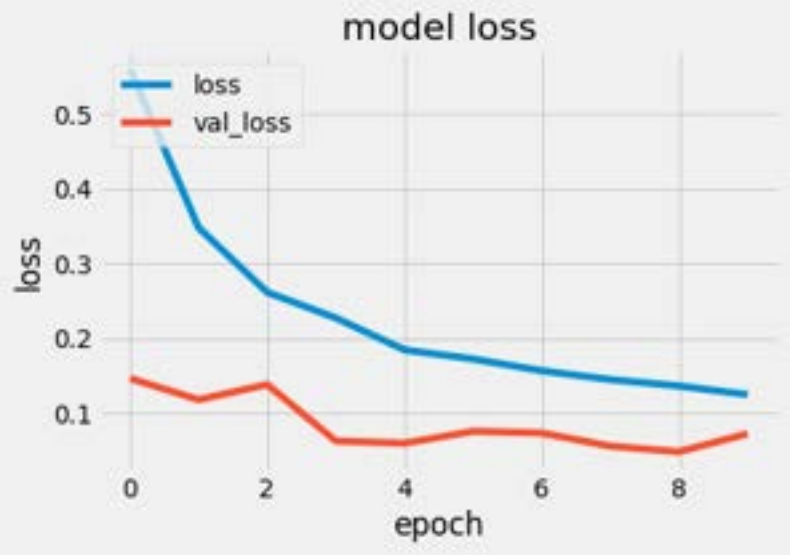


Fig 3.16 Model loss

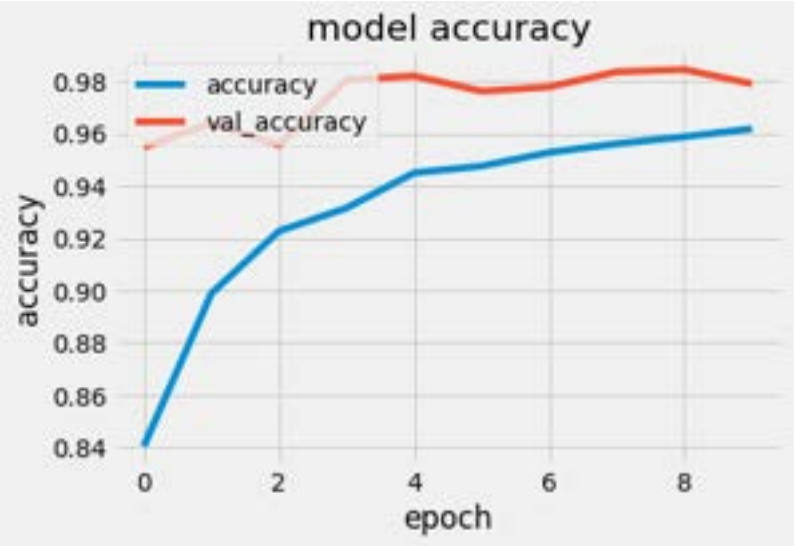


Fig 3.17 Model accuracy

Fig 3.18 Result for Traffic Sign Detection

The model gave an validation accuracy of 97.9% and validation loss of 7.24%, along with a training accuracy of 96.19% and training loss of 12.47% proving a perfectly balanced model without being overfit.

3.8 TRAFFIC LIGHT DETECTION[40]

Detection of traffic lights and understanding the rules play a vital role in autonomous vehicle navigation. In this project an diversified dataset is being used to detect traffic signals - daySequence and nightSequence data images have been used with a total of 96275 images.

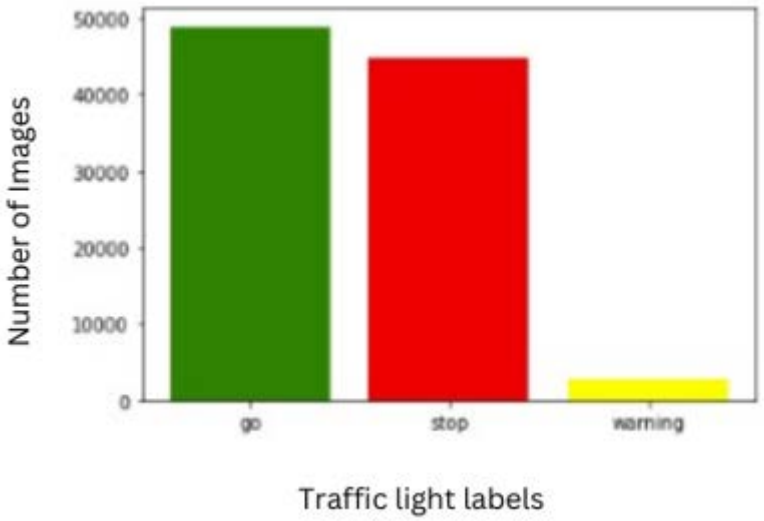


Fig 3.19 Traffic light dataset distribution

Further implementing the Support Vector Classifier (SVC) from the Support Vector Machine (SVM) methodology. SVM aims to find a hyperplane that classifies data points in an N-dimensional space. The SVM algorithm uses hyperplanes as decision boundaries to classify the three classes (Red, Yellow, Green) in the dataset. The objective is to maximize the margin between data points and the hyperplane using hinge loss as the loss function.

On training the model the following results have been obtained.

	Precision	Recall	f1-score	Support
go	0.96	1.00	0.98	100
stop	0.99	0.99	0.99	100
warning	0.99	0.95	0.97	100

Table 3.2 Traffic Light Detection Results

3.9 RURAL ROAD DETECTION

Current autonomous vehicles are applicable only in regions where there are proper lane markings and travel paths but in rural situations where the vehicle is congested and road markings are barely visible, navigation of vehicles becomes difficult. Hence an optimal algorithm for traversing through these roads is vital.

In this project an 3 channel color image is taken as input. Being in BGR format it is hard to interpret the image, therefore it is converted to HSV format from that manual threshold for the road has been devised. Further after detection of the kernel based arena of the road a contour is drawn for estimation of available path in front of the vehicle.

3.10 REINFORCEMENT LEARNING[46]

Reinforcement learning empowers autonomous vehicles to learn optimal control policies, adapt to complex environments, optimize rewards, explore and exploit effectively, and continuously improve performance. These capabilities are essential for autonomous vehicles to handle challenging real-world scenarios where precise control, quick decision-making, and adaptability are crucial factors.

The Box2D Car Racing environment[45] in OpenAI Gym is a complex and dynamic environment with various factors affecting the vehicle's movement, such as road conditions, obstacles, and friction. Reinforcement learning allows an autonomous vehicle to adapt and learn optimal control policies in such complex environments.
In the Box2D Car Racing environment, the rewards can be designed to align with specific objectives, such as completing a track quickly or avoiding collisions. By optimizing rewards, reinforcement learning allows the autonomous vehicle to learn policies that achieve desired behavior.

3.10.1 RL ENVIRONMENT [46]

The environment is built using the Box2D physics engine, which provides realistic physics simulation. It consists of a 2D racetrack with various features like curves, turns, hills, and obstacles.The objective is to control a car and navigate it through the track as quickly as possible.

State Space:

The state space of the environment consists of a combination of continuous and discrete variables.The continuous variables include the car's position, velocity, and angle.
The discrete variables represent the track segments the car is currently on.
Observation Space Box(0, 255, (96, 96, 3), uint8)

Action Space:

The action space is continuous and represents the control inputs to the car.
Commonly used action dimensions include steering, acceleration, and braking.
Agents can choose values within predefined ranges to control the car's movements.
Action Space Box([-1. 0. 0.], 1.0, (3,), float32)

If continuous there are 3 actions :

- 0: steering, -1 is full left, +1 is full right
- 1: gas
- 2: breaking

If discrete there are 5 actions:

- 0: do nothing
- 1: steer left
- 2: steer right
- 3: gas
- 4: brake

Rewards:

The environment provides rewards to guide the agent's learning process. Commonly used rewards include rewards for completing laps, penalties for collisions or going off-track, and rewards based on speed or smoothness of driving.

The rewards are designed to encourage the agent to learn optimal driving behavior.
The reward is -0.1 every frame and +1000/N for every track tile visited, where N is the total number of tiles visited in the track. For example, if you have finished in 732 frames, your reward is 1000 - 0.1*732 = 926.8 points.

3.10.2 TRAINING AND EVALUATION

The environment is typically used in reinforcement learning scenarios for training and evaluating autonomous driving agents.

Reinforcement learning algorithms, such as Deep Q-Networks (DQN) or policy gradients, can be applied to train agents to navigate the track efficiently. Agents are trained through trial and error, where they interact with the environment, receive rewards, and update their policies to maximize cumulative rewards. The episode

finishes when all the tiles are visited. The car can also go outside the playfield - that is, far off the track, in which case it will receive -100 reward and die.

Policy based learning algorithm:

Policy-based learning algorithms, denoted as π (pi), are a class of reinforcement learning algorithms that directly optimize a policy function to determine the agent's actions based on the observed state. Unlike value-based methods that aim to estimate the value function, policy-based methods focus on finding the best policy directly .this will ensure that our policy update will not be too large and that the training is more stable.

Policy Function:

In policy-based learning, a policy function $\pi(a|s)$ is used to determine the probability of taking action a given the current state s. The policy can be represented as a parametric function with parameters θ: $\pi(a|s; \theta)$.

The goal is to find the optimal parameters θ^* that maximize the expected cumulative reward.

Objective Function:

The objective function in policy-based learning is typically defined as the expected cumulative reward, also known as the return $L(\theta)$. PPO imposes a policy ratio, $r(\theta)$ to stay within a small interval around 1. That is the interval between $1-\epsilon$ and $1+\epsilon$.

The return is the sum of rewards obtained over time steps, discounted by a factor γ:

$L(\theta) = \mathbb{E}[\Sigma_i \gamma_i r_i]$, where r_i is the reward at time step i and γ is the discount factor.

$$L^{CLIP}(\theta) = E[min(r_t(\theta) A_t, clip(r_t(\theta), 1 - \epsilon, 1 + \epsilon), A_t] \qquad (3.7)$$

Policy Gradient:

The policy gradient algorithm is a popular approach in policy-based learning. It computes the gradient of the objective function with respect to the policy parameters θ.

The gradient can be computed using the likelihood ratio trick and the score function estimator:

$$\nabla \theta \, J(\theta) = \mathbb{E}[\nabla \theta \log \pi(a|s; \theta) \, Q(s, a)], \quad (3.8)$$

where $\nabla \theta \log \pi(a|s; \theta)$ is the score function and $Q(s, a)$ is the action-value function.

Gradient Ascent:

To update the policy parameters θ, gradient ascent is typically used to maximize the objective function. The update rule for the policy parameters can be written as:

$$\theta \leftarrow \theta + \alpha \, \nabla \theta \, J(\theta), \quad (3.9)$$

where α is the learning rate. The agent interacts with the environment by observing a state. The policy takes the state as input and outputs the probability distribution over possible actions.An action is selected based on the policy distribution, often using techniques like the softmax function.

The agent performs the selected action in the environment and receives a reward. The policy is updated using a gradient ascent method to maximize the expected reward. This process is repeated iteratively to improve the policy over time.

3.10.3 Q-LEARNING AND Q-VALUES

Q-learning is a value-based reinforcement learning algorithm that aims to learn the optimal action-value function Q(s, a), which represents the expected cumulative reward when taking action a in state s and following the optimal policy.

The optimal action-value function satisfies the Bellman equation:

$$Q^*(s, a) = \mathbb{E}[r + \gamma * \max(Q^*(s', a'))], \quad (3.10)$$

where r is the immediate reward, γ is the discount factor, s' is the next state, and a' is the next action.

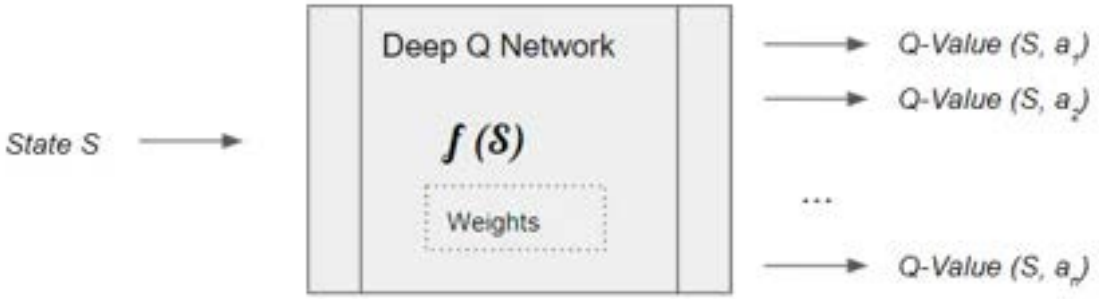


Fig 3.20 Functioning of Q-Network

Deep Q-Network (DQN):

Deep Q-Network (DQN) is a popular reinforcement learning algorithm that combines deep neural networks with Q-learning to learn an optimal policy. It enables agents to learn directly from high-dimensional raw sensory inputs, such as images, and make decisions in complex environments.

DQN extends Q-learning by utilizing deep neural networks as function approximators to estimate the action-value function Q(s, a). The DQN algorithm parameterized the action-value function with weights θ and uses a neural network (often a convolutional neural network) to approximate Q(s, a) given state s.

The network takes the state as input and produces Q-values for each possible action as output.

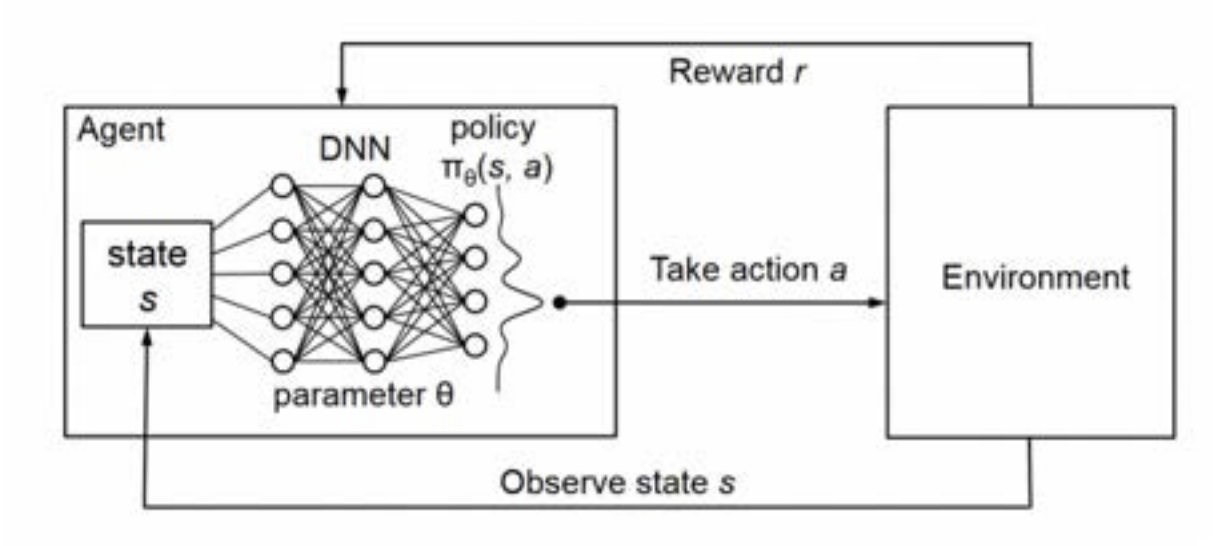


Fig 3.21 Overall representation of DQNA based RL network

Experience Replay:

DQN employs experience replay to stabilize learning and improve sample efficiency. Experience replay stores agent's experiences (state, action, reward, next state) in a replay memory buffer and samples mini-batches of experiences to train the network. This helps break the sequential correlation between samples and reduces the impact of overfitting to the most recent experiences.

Loss Function and Update Rule:

DQN uses the mean squared error (MSE) loss function to update the Q-network's weights. The loss is computed as the squared difference between the predicted Q-value and the target Q-value:

$$L(\theta) = \mathbb{E}[(r + \gamma * \max(Q(s', a'; \theta^-)) - Q(s, a; \theta))^2] \quad (3.11)$$

where θ^- represents the target network's weights.

The weights are updated using gradient descent:

$\theta \leftarrow \theta - \alpha * \nabla\theta L(\theta)$,

where α is the learning rate.

Target Network:

To ensure stable training, DQN (Deep Q-Network) employs a target network. The target network is a separate neural network with fixed weights, periodically updated with the weights of the Q-network. It plays a crucial role in calculating the target Q-values within the loss function, thus offering more consistent and reliable training targets. By utilizing the target network, DQN enhances the stability of the learning process, enabling more effective training and convergence towards optimal Q-values.

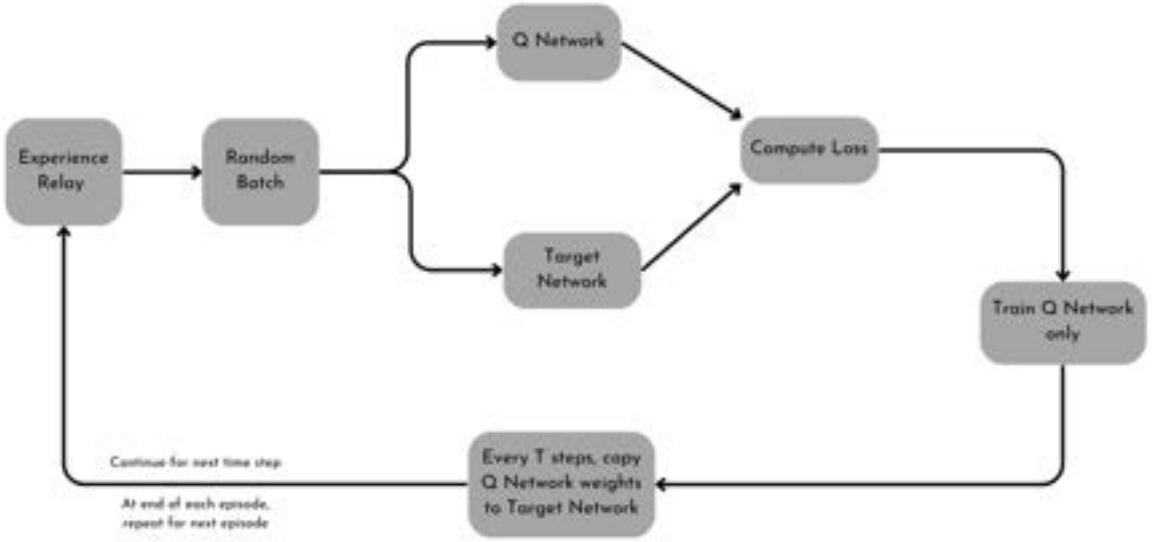


Fig 3.22 Detailed functioning of each block in RL network

Training process:

Fig 3.23 Environment representation

The reinforcement learning was executed for 300000 learning cycles and achieved a maximum score of 947

3.11 LANE DETECTION

One of the most challenging tasks of autonomous driving is traffic scene understanding, which comprises computer vision tasks like lane detection and semantic segmentation. Lane detection helps to guide vehicles and could be used in driving assistance systems.

This project utilizes the CULane dataset. The CULane dataset is a popular benchmark dataset for lane detection in autonomous driving research. It provides a large collection of labeled images and corresponding lane annotations captured from various driving scenarios. The dataset focuses on urban driving scenes and includes diverse environmental conditions, such as different weather conditions, lighting conditions, road types, and traffic densities.

3.11.1 DATASET DESCRIPTION:

The CULane dataset consists of 88,880 labeled images captured from different camera viewpoints in urban driving scenarios. Each image is of size 1640x590 pixels, representing a wide field of view. The dataset provides lane annotations in the form of lane markings, which are labeled with color-coded segmentation masks. The lane annotations include both the visible lane markings and the invisible extrapolated lanes beyond the field of view. Conventional methods uses hardcode image processing technique - Hough lines

Dataset consisting of original image, corresponding mask and coordinates for points.

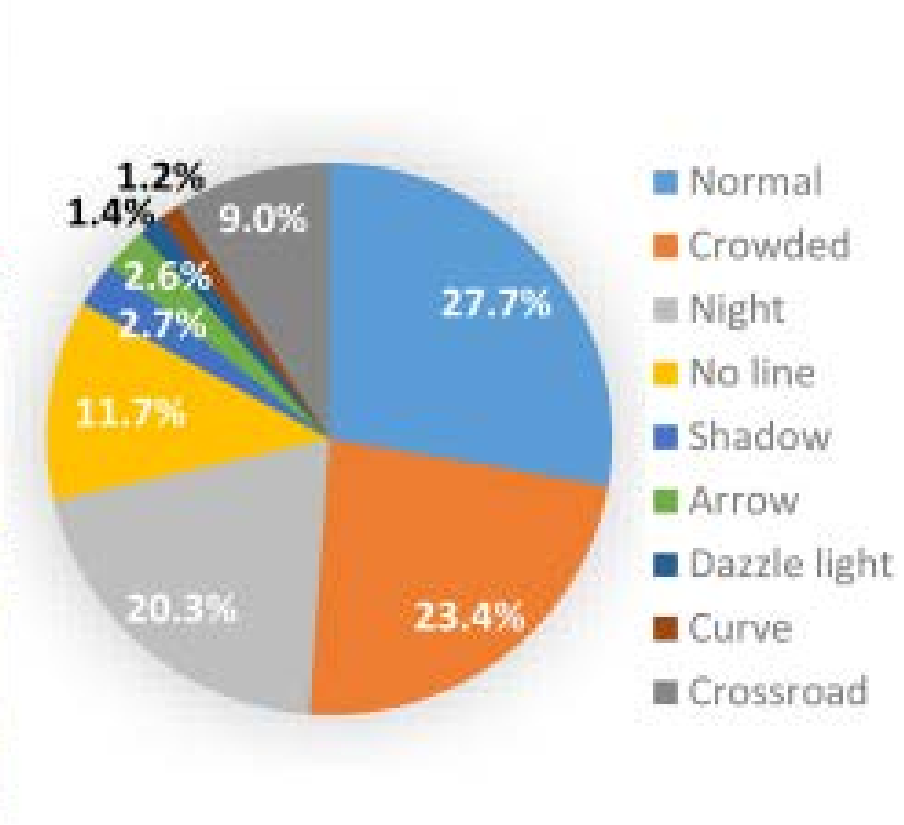


Fig 3.24 Distribution of dataset [47]

Lane Annotations:

The lane annotations in the CULane dataset are represented as pixel-level segmentation masks. Each lane marking is assigned a unique color, allowing the identification and differentiation of individual lane markings. The dataset provides annotations for lane markings in different configurations, including single lane, multiple lanes, curved lanes, and lane merges/diverges.

Lane Count:

The dataset contains images with varying numbers of lanes, ranging from single-lane to multi-lane scenarios. This allows for the evaluation of lane detection algorithms across different lane configurations. With a maximum of 4 lanes of classification is possible

Weather Conditions:

The dataset covers diverse weather conditions, including sunny, rainy, and foggy conditions. This facilitates the evaluation of lane detection algorithms under different visibility and weather challenges.

Lighting Conditions:

Images in the CULane dataset are captured under various lighting conditions, such as daylight, low-light, and nighttime scenarios. This allows researchers to study the performance of lane detection methods in different lighting conditions.

Road Types:

The dataset includes images captured on different road types, including highways, urban roads, and suburban streets. This provides a realistic representation of lane detection challenges in different driving environments.

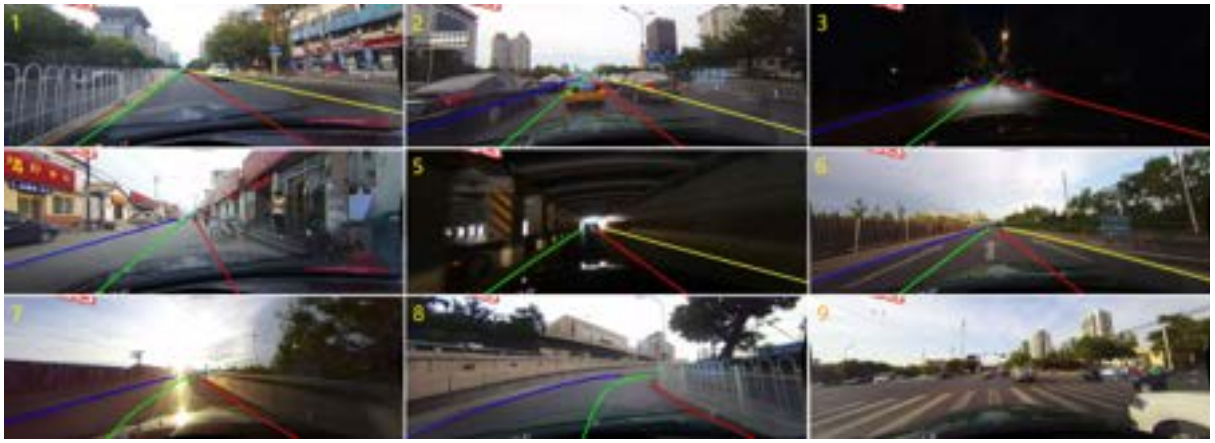


Fig 3.25 Sample images from dataset

3.11.2 NEURAL NETWORK

Spatial CNN (SCNN), a generalization of deep convolutional neural networks to a rich spatial level. In a layer-by-layer CNN, a convolution layer receives input from the former layer, applies convolution operation and nonlinear activation, and sends the

result to the next layer. This process is done sequentially. Similarly, SCNN views rows or columns of feature maps as layers and applies convolution, nonlinear activation, and sum operations sequentially, which forms a deep neural network. In this way information could be propagated between neurons in the same layer. It is particularly useful for structured object such as lanes, poles, or truck with occlusions, since the spatial information can be reinforced via inter-layer. in cases where CNN is discontinuous or is messy, SCNN could well preserve the smoothness and continuity of lane markings and poles

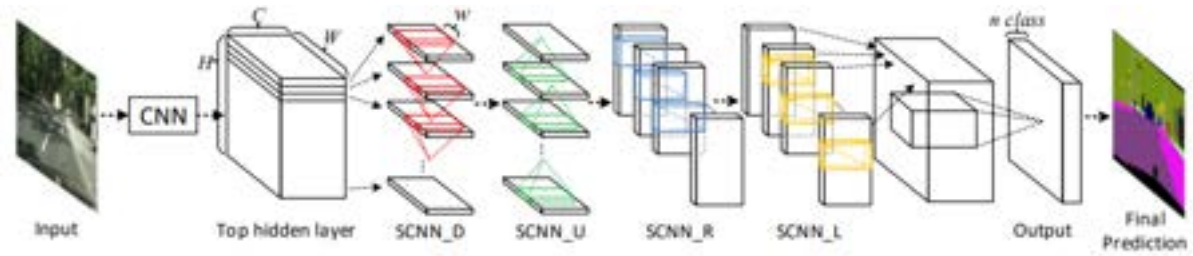
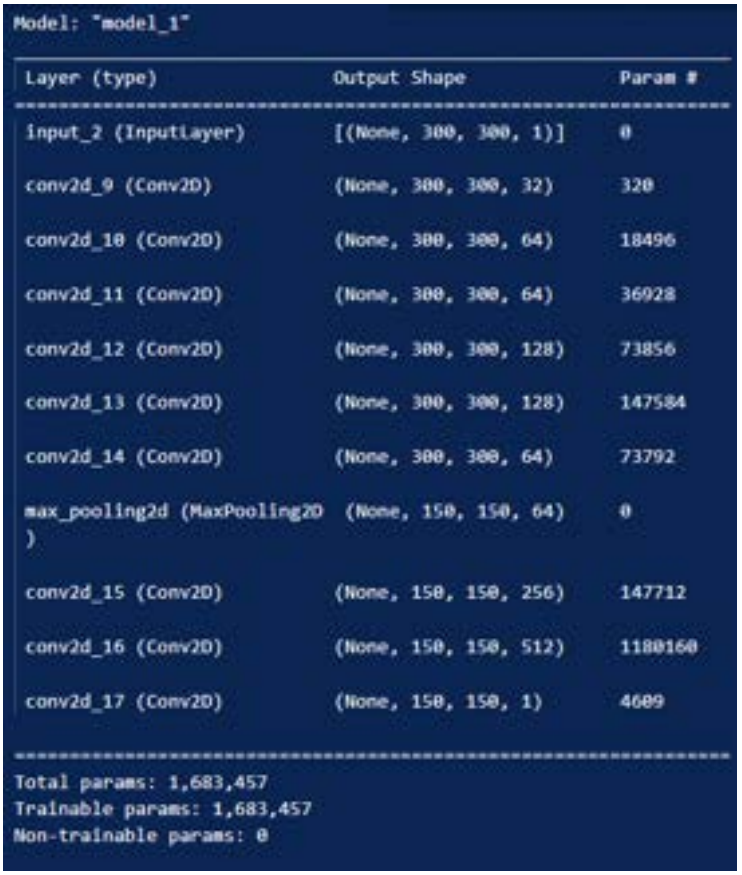


Fig 3.26 CNN architecture representation

SCNN is applied to a 3-D tensor of size C × H × W, where C, H, and W represent the number of channels, rows, and columns respectively. The tensor is divided into H slices, and the initial slice is passed through a convolution layer with C kernels of size C × w, where w denotes the kernel width. In contrast to a traditional CNN, where the output of a convolution layer is directly fed into the next layer, in SCNN, the output is added to the subsequent slice to generate a new slice. This new slice is then processed by the next convolution layer.

By splitting the tensor into slices and incorporating the addition operation, SCNN introduces a unique approach to processing 3-D tensors. This strategy enables the network to capture spatial dependencies within the tensor while maintaining information continuity across slices. Through successive convolutional layers, SCNN gradually refines the slices, leveraging both local and global features to enhance the

overall representation and extraction of meaningful patterns. The iterative updating process ensures that each slice benefits from the knowledge gained by previous slices, contributing to improved feature learning and potential performance gains in tasks where spatial information is crucial.



```
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 300, 300, 1)]     0
conv2d_9 (Conv2D)            (None, 300, 300, 32)      320
conv2d_10 (Conv2D)           (None, 300, 300, 64)      18496
conv2d_11 (Conv2D)           (None, 300, 300, 64)      36928
conv2d_12 (Conv2D)           (None, 300, 300, 128)     73856
conv2d_13 (Conv2D)           (None, 300, 300, 128)     147584
conv2d_14 (Conv2D)           (None, 300, 300, 64)      73792
max_pooling2d (MaxPooling2D  (None, 150, 150, 64)      0
)
conv2d_15 (Conv2D)           (None, 150, 150, 256)     147712
conv2d_16 (Conv2D)           (None, 150, 150, 512)     1180160
conv2d_17 (Conv2D)           (None, 150, 150, 1)       4609
=================================================================
Total params: 1,683,457
Trainable params: 1,683,457
Non-trainable params: 0
```

Fig 3.27 Detailed architecture info for SCNN

	Mean square error
Optimizer	Adam
Metrics	accuracy
Monitor	‘val-loss’
Epochs	100

Table 3.3 Lane Detection Hyper parameters

SCNN's performance is evaluated using metrics such as pixel-level accuracy, mean Intersection over Union (mIoU), F1 score, and precision-recall curves.SCNN effectively captures the spatial information of lane markings, allowing it to handle challenging lane scenarios with complex structures. The model can adapt to varying lighting conditions and lane marking styles.SCNN has demonstrated its capability in accurate and robust lane detection in various driving scenarios. Its ability to capture spatial configuration information contributes to its success in handling challenging lane detection tasks.

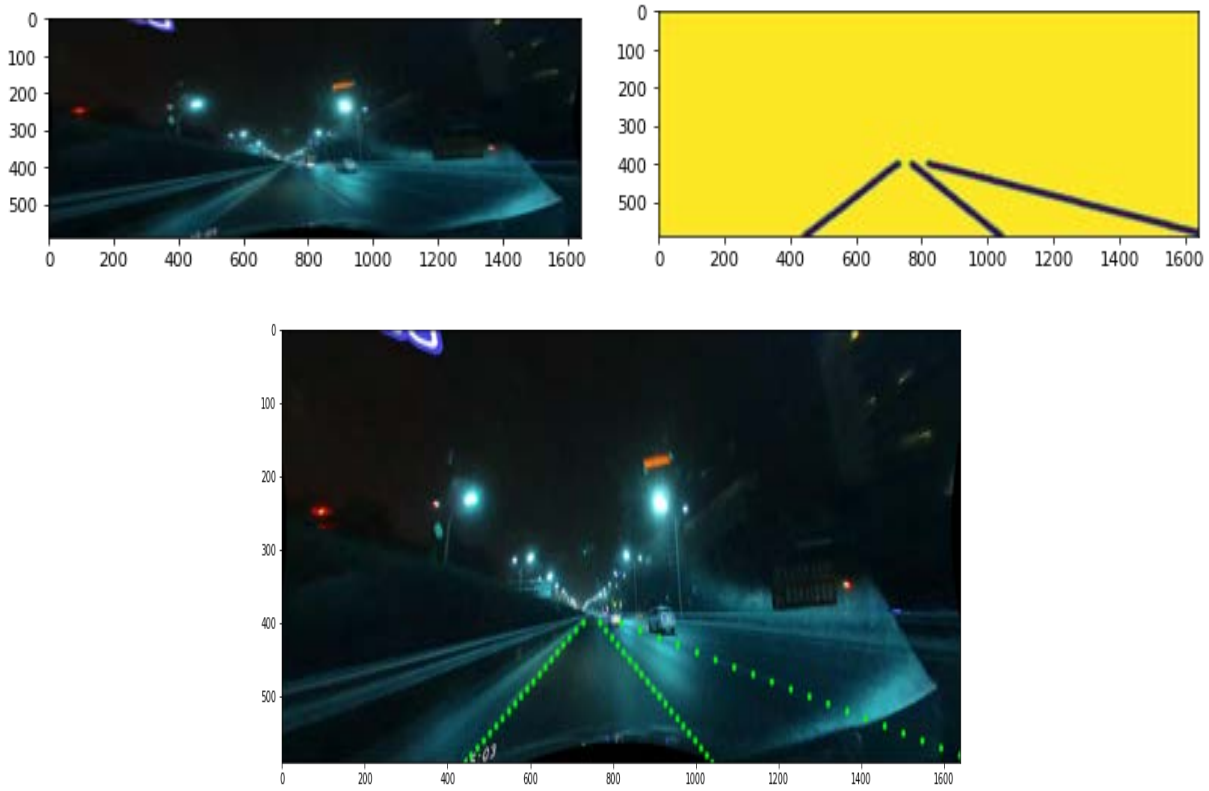


Fig 3.28 Lane Detection

The above picture represents the dataset sample image and its corresponding true values.

Model trained for entire dataset and over 16 Lakhs trainable parameters were trained With an MSE of 11.4K over 10 epochs

3.12 CUDA PROCESSING

CUDA (Compute Unified Device Architecture) parallel processing is a programming model and platform developed by NVIDIA for leveraging the power of GPUs (Graphics Processing Units) to perform massively parallel computations.GPUs are highly parallel processors designed for rendering graphics, where each pixel on the screen can be processed independently. Modern GPUs consist of thousands of cores that can execute multiple threads simultaneously.

These cores are organized into streaming multiprocessors (SMs) that contain multiple execution units, shared memory, and other resources.CUDA provides a programming model that enables a platform to write parallel code for GPUs. Kernels are launched with a specified number of threads, and each thread executes the same code but operates on different data.

CUDA achieves parallel execution by dividing the workload across multiple threads that run simultaneously on the GPU. Each thread processes a small portion of the data, and together, they collectively perform the computation. Threads are organized into blocks, and blocks are further organized into a grid. The GPU scheduler assigns blocks to SMs, and each SM schedules and executes the threads within a block.

Memory Hierarchy:

CUDA provides various memory spaces that facilitate efficient data access and sharing between threads.

Global memory: It is accessible by all threads and persists throughout the lifetime of the program.

Shared memory: It is a fast and shared memory space within an SM that allows threads within a block to communicate and cooperate.

Local memory: It is private to each thread and used for storing private variables and temporary data.

Constant memory: It is read-only memory that provides fast access to constant data.

3.12.1 CUDA IN NEURAL NETWORK

CUDA parallel processing plays a crucial role in accelerating neural network model training. It allows leveraging the computational power of GPUs to perform parallel computations and speed up the training process. GPU based Acceleration has been used for both model training purpose and execution of various features of the vehicle

Parallelization of Matrix Operations: Neural networks heavily rely on matrix operations, such as matrix multiplication and element-wise operations.

GPUs excel at performing these matrix operations in parallel due to their large number of cores and memory bandwidth. CUDA parallel processing allows the distribution of these matrix operations across multiple threads, enabling simultaneous computations on different parts of the matrices.

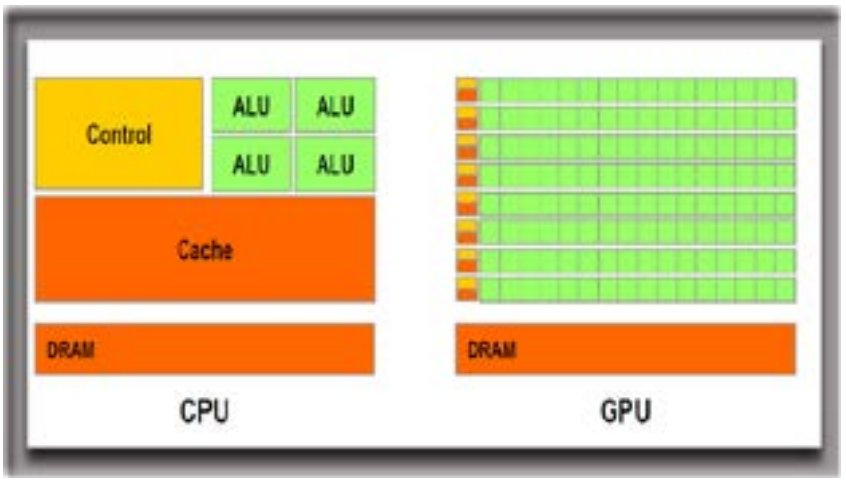


Fig 3.29 CPU vs GPU functional representation

Parallelization of Matrix Operations:

- Neural networks heavily rely on matrix operations, such as matrix multiplication and element-wise operations.
- GPUs excel at performing these matrix operations in parallel due to their large number of cores and memory bandwidth.
- CUDA parallel processing allows the distribution of these matrix operations across multiple threads, enabling simultaneous computations on different parts of the matrices.

By leveraging CUDA parallel processing in neural network model training, significant speedup can be achieved compared to traditional CPU-based training. The massive parallelism offered by GPUs accelerates matrix operations, enables efficient mini-batch, data, and model parallelism, and facilitates asynchronous training. These advancements contribute to faster and more efficient training of complex neural network models.

Features such as Traffic sign, Reinforcement env, lane detection extracts a lot of ALU process within the CPU resulting in less FPS - GPU resolves this difficult
Hardware accelerator used in this project for both model training and execution is NVIDIA GTX 1650. Upon testing there has been a reduction in FPS observed from an average of 6.8 seconds per frame to 0.5 seconds per frame

3.13 REROUTING ALGORITHM

Depending upon the current position of the vehicle if the navigation is stagnant Rerouting takes place. Rerouting is of significant importance in autonomous vehicles as it plays a crucial role in ensuring efficient and safe navigation. Rerouting allows autonomous vehicles to dynamically respond to changing road conditions such as traffic congestion, accidents, road closures, or construction. The vehicle can choose alternative routes to avoid delays and reach the destination more efficiently. Rerouting

can contribute to overall traffic management and congestion reduction efforts. By distributing traffic across different routes, autonomous vehicles can help alleviate congestion in specific areas and contribute to smoother traffic flow, benefiting not only the vehicle occupants but also other road users.

Rerouting in autonomous vehicles provides flexibility, adaptability, and efficiency in navigation. It enables the vehicles to dynamically respond to changing conditions, select optimal routes, ensure passenger safety and comfort, and contribute to traffic management efforts. By leveraging real-time data and intelligent algorithms, rerouting plays a vital role in enhancing the capabilities of autonomous vehicles and improving the overall transportation experience.

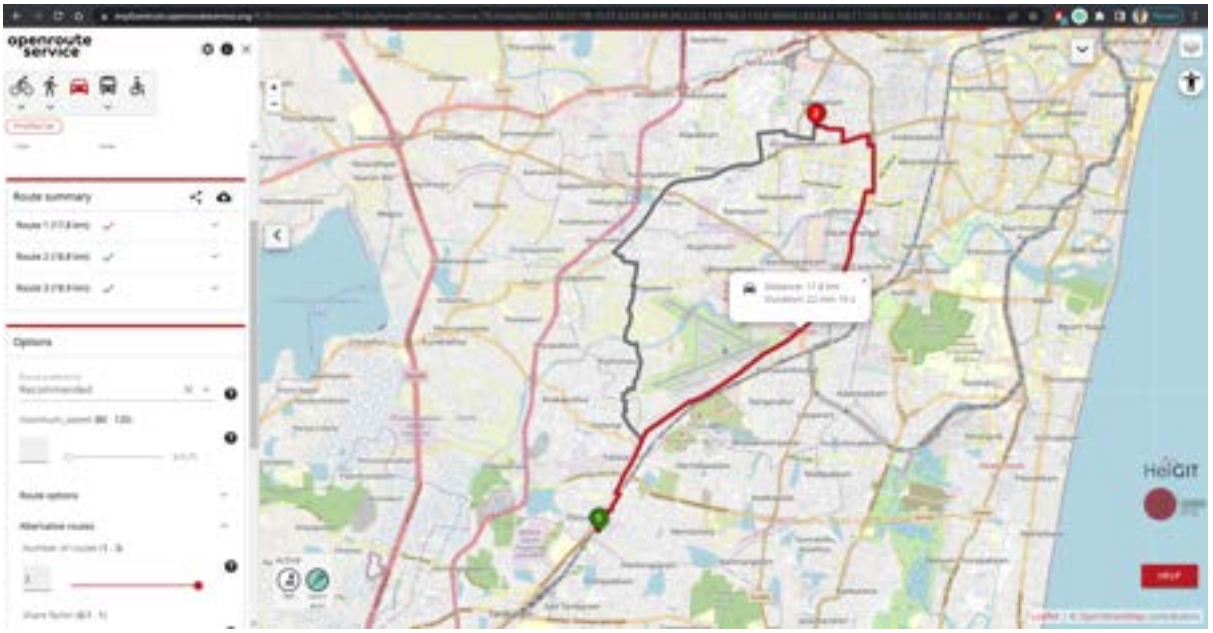


Fig 3.30 Rerouting algorithm route 1

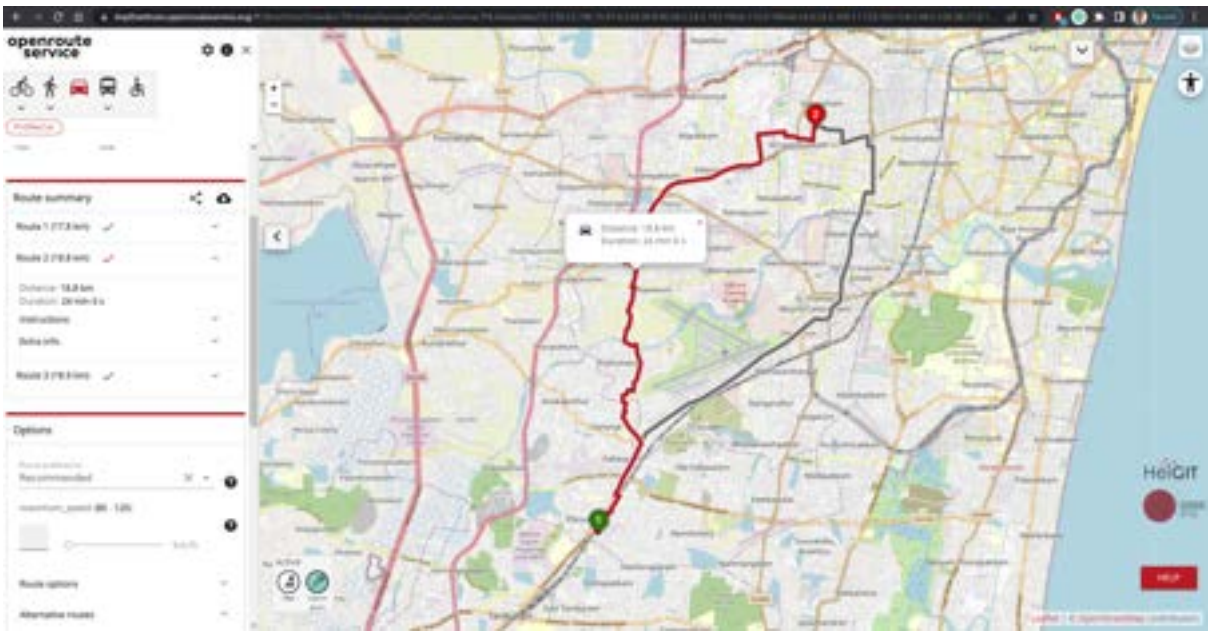


Fig 3.31 Rerouting algorithm route 2

The Rerouting algorithm will commence its operation only after the vehicle remains stationary at a consistent location without any navigation for a duration of 10 seconds. Additionally, if the vehicle's path deviates from the predetermined route due to environmental limitations, a fresh session is initiated to determine the user's requirements. A distance threshold of 90 meters is set to define the extent of deviation allowed from the intended path.

During this process, the current location of the vehicle serves as the initial point, while the destination location is obtained based on the preceding command. Subsequently, a sequence of steps is executed, which involves acquiring coordinates, determining the travel path, and providing motor instructions.

To elaborate, the Rerouting algorithm becomes operational only when the vehicle remains at a fixed location without navigation for at least 10 seconds. If the vehicle strays from the prescribed path due to environmental constraints, a new session is

established to assess the user's needs. This deviation threshold is set at 90 meters, beyond which a reroute is deemed necessary.

3.14 RASPBERRY PI(RASPBERRY PI 4)

The Raspberry Pi 4 is a powerful and versatile single-board computer that can be used for a wide range of applications. It is compatible with various operating systems, including Raspberry Pi OS, Ubuntu, and Windows 10 ARM, and it supports a wide range of programming languages and software libraries. It was released in June 2018 and comes with several improvements over its predecessor, the Raspberry Pi 3.

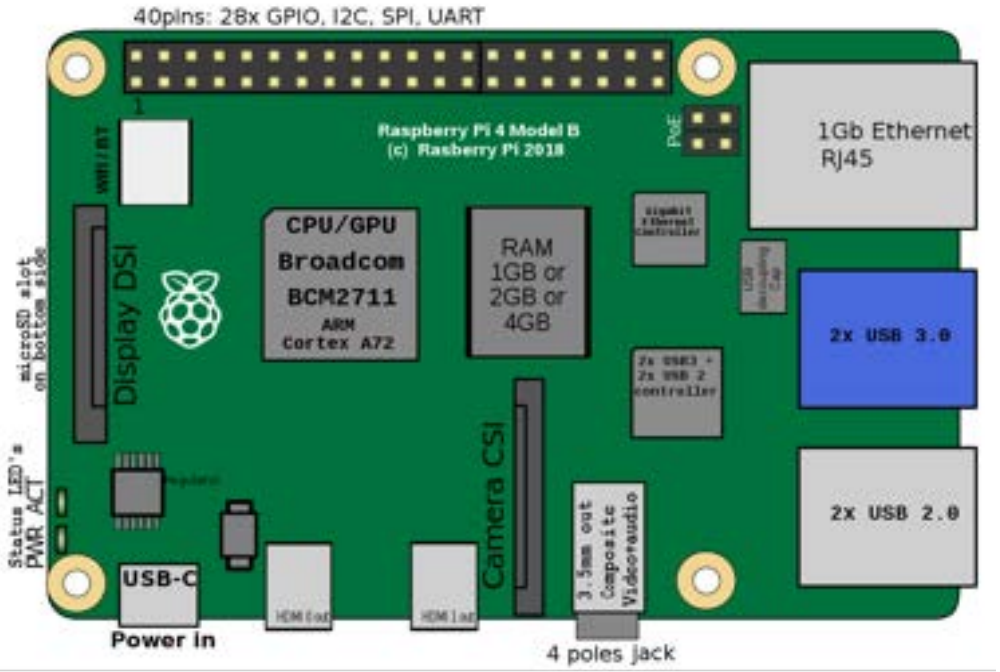


Fig 3.32 Raspberry Pi 4.[42]

Specification:

- Processor: Broadcom BCM2711 quad-core Cortex-A72 (ARMv8) 64-bit SoC (System-on-a-Chip) with a clock speed of 1.5 GHz.
- RAM: Options for 2GB, 4GB, or 8GB LPDDR4 SDRAM.
- Video Output: Dual micro HDMI ports, 4K resolution at 60Hz
- Storage: MicroSD card slot for external storage.

- USB: Two USB 3.0 ports and two USB 2.0 ports for connecting peripherals
- GPIO Pins: 40-pin GPIO header, allowing for connections to various external devices and expansion boards.
- Power: USB-C power connector for powering the Raspberry Pi 4.
- Operating System: Raspbian, Ubuntu, and other Linux distributions.
- Dimensions: The Raspberry Pi 4 Model B has dimensions of approximately 85mm x 56mm x 17mm.

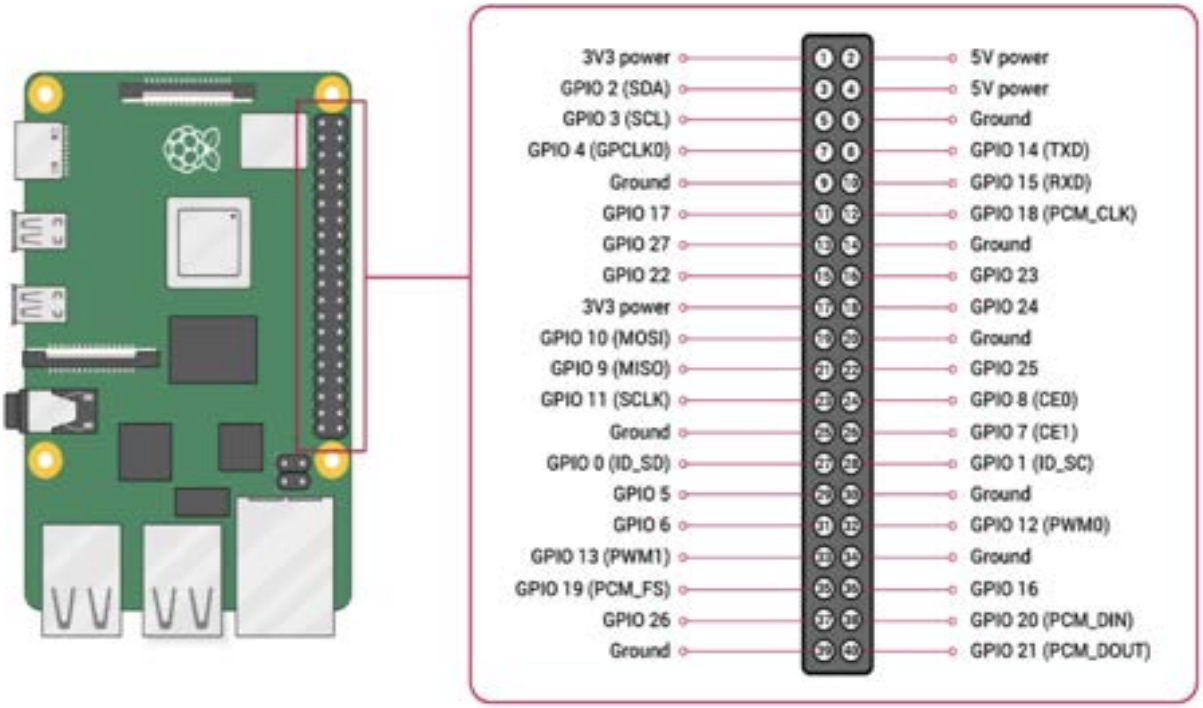


Fig 3.33 Pin configuration of Raspberry Pi[42]

3.15 GPS MODULE(GPS NEO-6M MODULE)

The GPS NEO-6M Module is a GPS receiver module that can be used for a variety of navigation applications. It is based on the Ublox NEO-6M GPS module and is capable of providing accurate positioning information in a variety of conditions. The module is capable of high sensitivity and tracking capabilities and can be used in a variety of

applications including navigation, tracking, and other location-based services. The module is compatible with various GPS software platforms and supports various communication protocols.

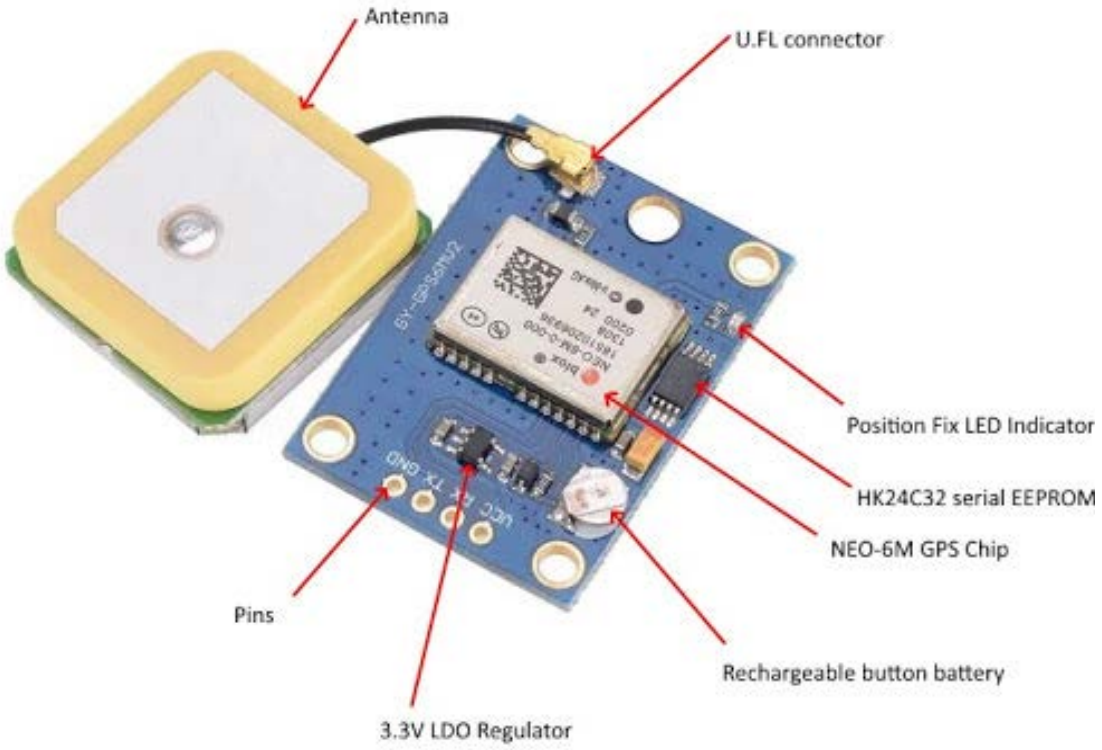


Fig 3.34 GPS Module

Specification:

- Standalone GPS receiver
- Operating temperature range: -40ºC TO 85°C
- UART Interface at the output pins (Can use SPI ,I2C and USB by soldering pins to the chip core)
- Receiver type: GPS L1 frequency - SBAS - 50 Channels
- Maximum navigation update rate: 5Hz
- Anti-jamming technology
- Maximum DC current at any output: 10mA
- Operation limits: Gravity-4g, Altitude-50000m, Velocity-500 m/s
- EEPROM with battery backup

- Sensitivity: -160dBm
- Supply voltage: 3.6V
- Default baud rate: 9600bps

GPS Module	Raspberry Pi
VCC	3.3V (Pin 1)
RX	TXD/GPIO 14 (Pin 8)
TX	RXD/GPIO 15 (Pin 10)
GND	Ground (Pin 6)

Table 3.4 GPS to raspberry pi pin connections

3.16 CAMERA MODULE (LOGITECH HD WEBCAM C270)

The camera module in Fig 3.5 is equipped with a USB A port which makes interface with Raspberry pi much easier.

Fig 3.35 Camera module

Specifications

- Megapixel 3 MP
- Maximum video resolution 1280 x 720 pixels
- Maximum frame rate 30 fps
- Supported video modes 720p
- Diagonal field of view 55°

Weight & Dimensions

- Depth 66.6 mm
- Height 73.9 mm
- Weight 75 g

3.17 BATTERY

The battery that was used to power up this car was rated at 11.1V 3.2A, meaning that 3.2 amps can be drawn from the battery at 11.1 volts for an hour. This battery lasts about 1 hour when it is connected to the motor.

Fig 3.36 12V-7LiPo battery

3.18 MOTOR

This vehicle will be run only on one motor. The motor has a nominal torque of 34.3 N-cm with a drop of 10K. The main feature of this motor is that it has a 27 mm long

shaft with a 6 mm diameter. Usually motors available on the market have shaft lengths from 20~22mm; but we chose this because with a long shaft it is too easy to mount a wheel or any kind of coupling on the shaft.

Specifications:

- Requires 12V DC
- 1000 RPM
- Torque: 34.3 N-cm
- Full load current: 6.836 A

Fig 3.37 12V DC motor

3.19 ULTRASONIC SENSOR

The HC-SR04 ultrasonic distance sensor is a widely used and popular module for measuring distances using ultrasonic waves. It consists of an ultrasonic transmitter and receiver, which work together to emit and detect high-frequency sound waves. By measuring the time taken for the sound waves to travel to an object and return, the sensor can calculate the distance between the sensor and the object. It typically operates in the range of 2cm to 400 cm and provides accurate distance measurements with a resolution of a few millimeters.

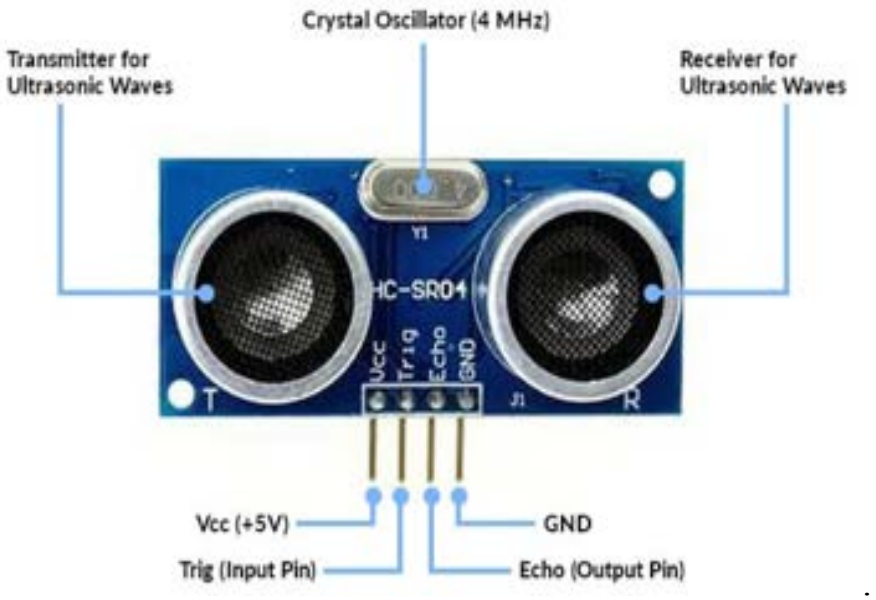


Fig 3.38 Ultrasonic sensor

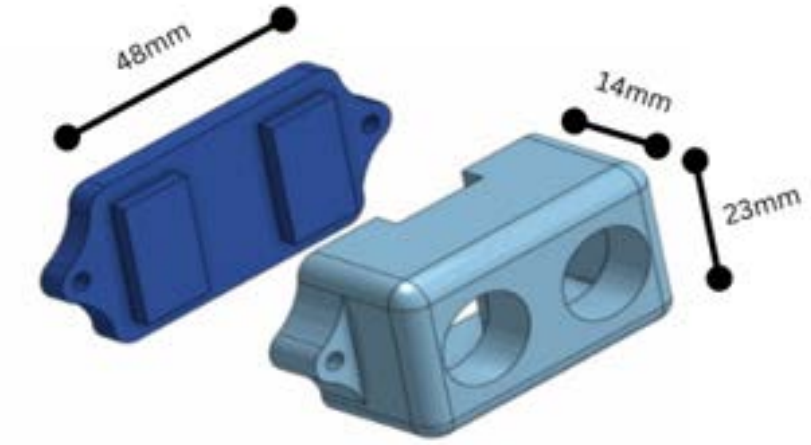


Fig 3.39 CAD model for ultrasonic sensor case

A 3D model for the casing of the ultrasonic sensor was designed using fusion360 for avoiding static energy between the peripherals of the ultrasonic sensor.

3.20 ESP-32

The ESP32 is a powerful and versatile microcontroller developed by Espressif Systems.It is widely used in a variety of IoT applications, including smart home devices, wearables, and industrial automation.

Specifications:

- CPU: Dual-core Tensilica LX6 microprocessor, clocked at up to 240MHz.
- Memory: 520KB SRAM and up to 16MB of external flash memory.
- Connectivity
    - Wi-Fi: 802.11 b/g/n/e/i wireless LAN with WPA/WPA2 PSK support.
    - Bluetooth: Bluetooth Low Energy (BLE) and Classic Bluetooth with support for A2DP and AVRCP profiles.
    - Ethernet: 10/100Mbps Ethernet MAC with dedicated DMA and IEEE 1588 support.
- I/O Interfaces:
    - 34x programmable GPIOs.
    - 18x analog-to-digital converters (ADCs) with a resolution of up to 12 bits.
    - 2x 8-bit digital-to-analog converters (DACs).
    - 2x I2C interfaces.
    - 3x UART interfaces.
    - 4x SPI interfaces.
    - 2x I2S interfaces.
- Operating voltage: 2.2V to 3.6V.
- Power consumption: Ultra-low power consumption in deep sleep mode, with current consumption as low as 5µA.
- Operating temperature: -40°C to +125°C.

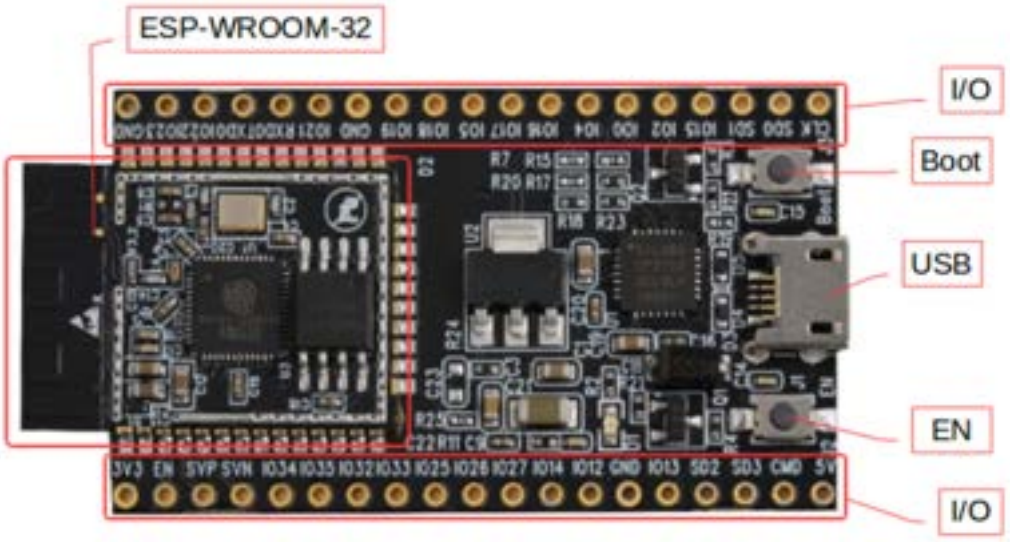


Fig 3.40 ESP 8266

3.21 SG90 MICRO SERVO

It is a popular micro servo motor commonly used in hobbyist and DIY projects. It is a small, low-cost servo motor that can rotate 180 degrees with a maximum torque of 1.8 kg-cm. It operates at 4.8-6V and has a weight of approximately 9 grams, making it ideal for small-scale robotics and model control applications.

SG-90 SPECIFICATIONS

- Operating Voltage is +5V typically
- Torque: 3.5kgf / cm & Operating voltage: 4.8 V (~5V)
- Operating speed is 0.1s/60°
- Gear Type: Plastic
- Rotation : 0°-180°
- Weight of motor: 9gm
- Package includes gear horns & screws
- Dimension:23.2 x 11.8 x 31mm approx

Fig 3.41 Mini Servo Motor

3.22 MG995 HIGH-SPEED DIGITAL SERVO MOTOR

The TowerPro MG995 High-Speed Digital Servo Motor rotates 90° in each direction making it 180° servo motor. It is a Digital Servo Motor that receives and processes PWM signal faster and better. It equips sophisticated internal circuitry that provides good torque, holding power, and faster updates in response to external forces.

MG995 SPECIFICATIONS

- Operating Voltage : 4.8 ~ 6.6v
- Temperature Range : 0- 55℃
- Stall Torque @4.8V : 9.40 kg-cm
- Stall Torque @6.6V : 11 kg-cm
- Operating Speed @4.8V : 0.19sec/60°
- Operating Speed @6.6V : 0.15sec/60°
- Dead Bandwidth : 1uS
- No. of teeth : 25

Fig 3.42 High Torque Servo Motor

3.23 DC MOTOR PWM SPEED REGULATOR

A 1.8V-12V 2A DC Motor PWM Speed Regulator is a device that allows you to control the speed of a DC motor by adjusting the width of the pulses of electricity that are applied to the motor. The speed regulator has a potentiometer that you can use to adjust the pulse width, and it also has a LED indicator that shows you the current speed of the motor.

1.8V-12V 2A DC Motor PWM Speed Regulator specifications:

- Input voltage: 1.8V-12V DC
- Output current: 2A
- Adjustable speed range: 0-100%
- LED indicator
- Resettable fuse

Fig 3.43 PWM Module

CHAPTER 4
WORKING OF VEHICLE

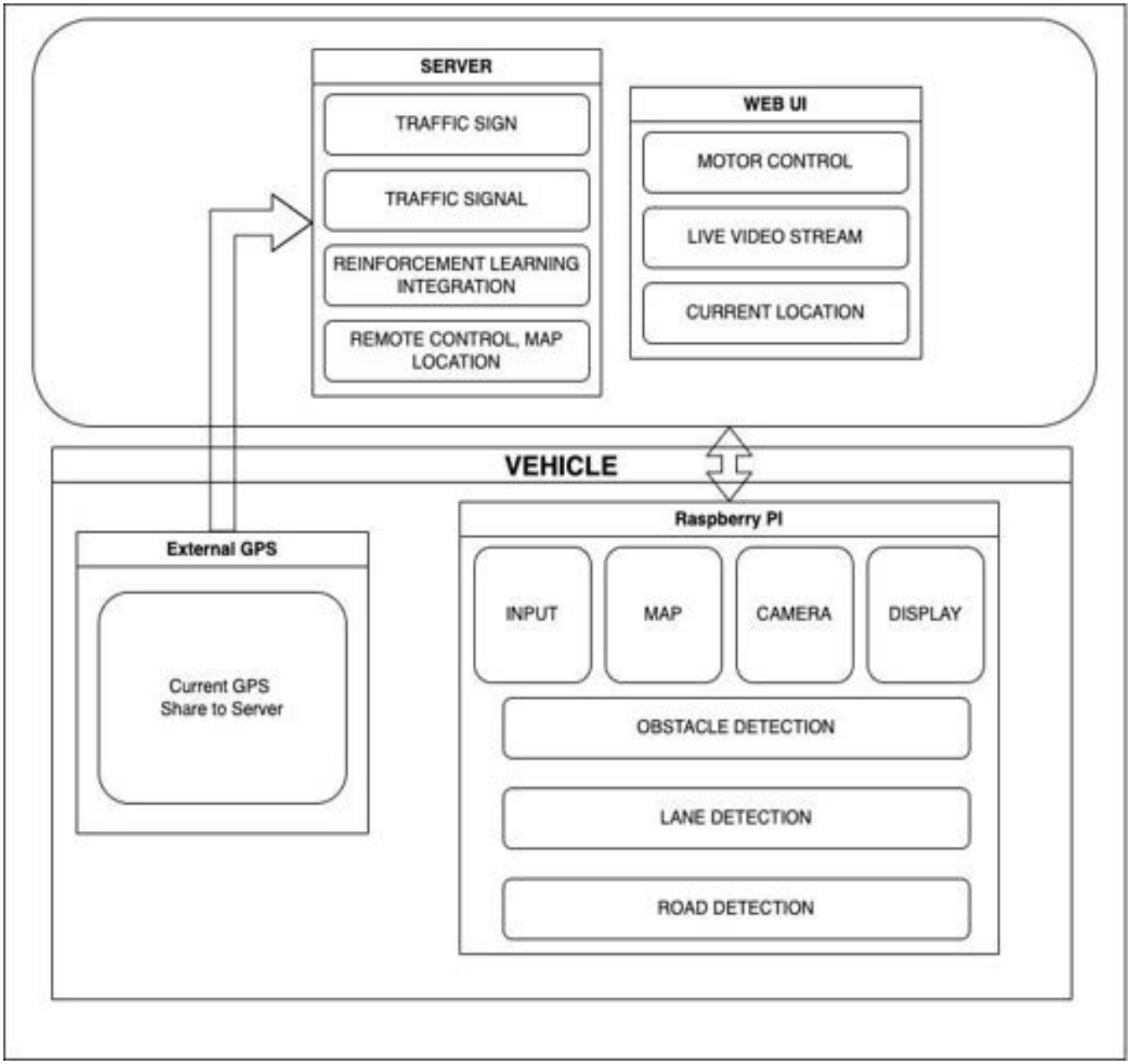


Fig 4.1 Complete Software architecture

The analysis process is divided between two key components: the Raspberry Pi(also referred as native) and the server. Each component plays a crucial role in analyzing the data collected from the camera, microphone and other sensors. The native will be

in the vehicle where the server will be the web server which utilizes a web compiler to execute the corresponding features. The presence of external GPS within the vehicle assists in providing accurate location including the position of the vehicle in the lane based on the GPS module present within the system. The Signal from raspberry pi, external GPS will be transmitted to the server to execute additional features apart from what has been executed in native.

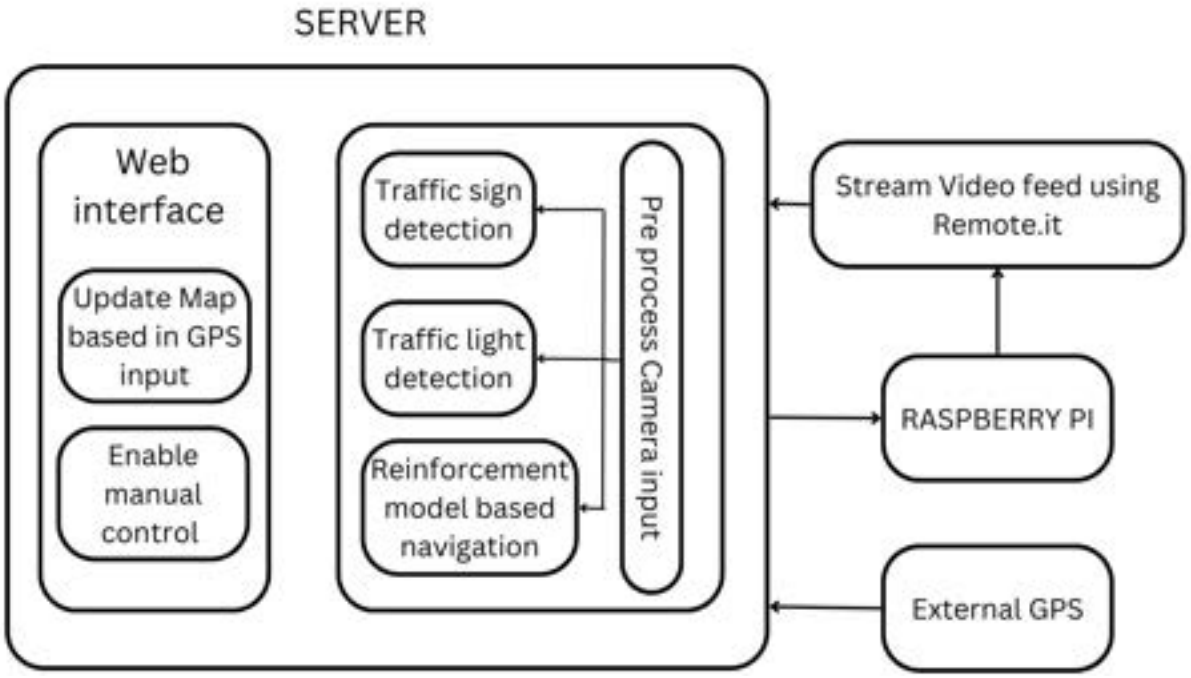


Fig 4.2 Server architecture

The server takes on computationally intensive tasks and handles those with lower priority. This includes critical functions such as traffic sign detection, which identifies and interprets road signs, and traffic signal detection, which recognizes and understands traffic lights. Additionally, the server is responsible for remote control functionalities, allowing for remote operation and control of the vehicle, as well as map routing, which determines the optimal route based on destination information. To facilitate map routing, the server receives GPS data through a dedicated GPS module, ensuring accurate and reliable navigation. The video stream from raspberry pi is transmitted to the server by using remote.it web interface to establish connection

between raspberry pi and server. Upon receiving the image it is processed in parallel using the server GPU module to obtain the results in shorter computation time.

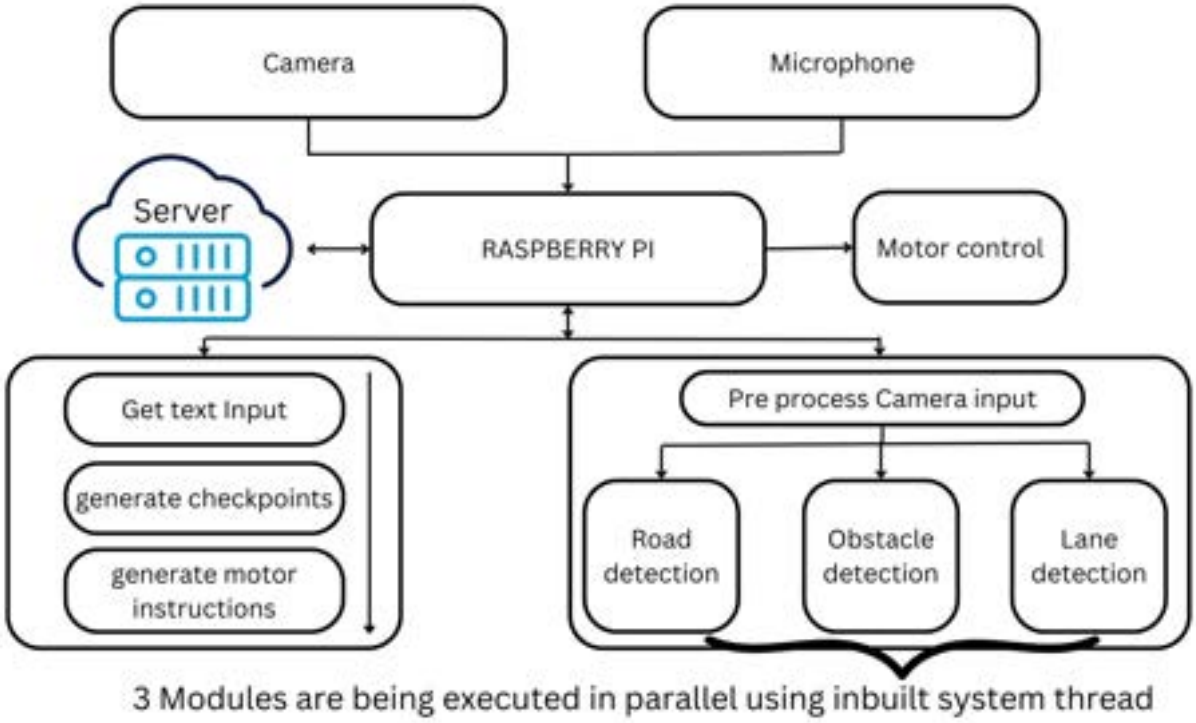


Fig 4.3 Native architecture (Raspberry Pi)

Meanwhile, the Raspberry Pi (native) focuses on specific analysis tasks that require real-time processing and immediate response. These tasks include lane detection, which identifies and tracks lane markings to ensure proper vehicle positioning, road detection, which distinguishes between different road types and surfaces, and object detection, which recognizes and classifies various objects in the environment that are executed in parallel within the system using system thread modules . The initial process of the vehicle functionality begins at the native where the user gives the initial instructions. Upon receiving the instructions, a sequential process of input to generate the motor instructions is carried out. Further the rerouting happens within the native system. By conducting these analyses on the Raspberry Pi, the system can efficiently

process the data and generate prompt feedback, enabling the vehicle to make timely decisions and adjustments.

Once the analyses are complete, the results from both the Raspberry Pi and the server are consolidated. The server sends its analysis results back to the Raspberry Pi, which integrates them into its own findings. This collaborative approach ensures comprehensive and accurate analysis, taking advantage of the strengths and capabilities of both components.

Based on the collective analysis, the Raspberry Pi generates motor instructions that control the vehicle's movements. These instructions are generated using predefined checkpoints and are designed to ensure safe and precise navigation. By relying on the analysis outcomes, the system can make informed decisions about acceleration, deceleration, and steering, allowing for autonomous or semi-autonomous driving.

Additionally, a display is installed on the vehicle to provide a visual output of the camera's feed. This display serves as a real-time monitoring tool, allowing users to observe the analyzed data and the surrounding environment. It provides valuable situational awareness, enabling users to stay informed about potential obstacles, road conditions, and other relevant information during the analysis and operation of the vehicle.

Vehicle Control:

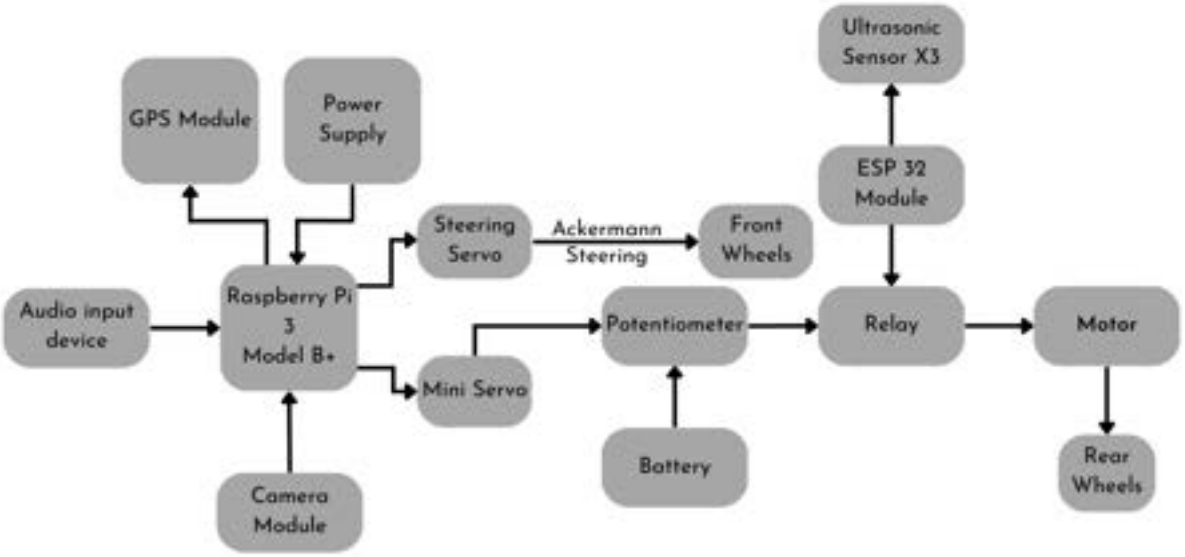


Fig 4.4 Hardware Block diagram

The Raspberry Pi can be used as a client device to communicate with the server over the internet. It can send requests to the server and receive responses that contain the outputs of the neural network.

Based on inputs from the camera module and the server the Raspberry Pi transmits instructions to both the PWM module and high torque servo motor, enabling precise control of speed and steering. Additionally, the ultrasonic sensor continually monitors the surroundings. When an obstacle is detected and its distance from the vehicle falls below a specified threshold, a relay connected to the motor is triggered. This relay promptly interrupts the power supply from the battery to the motor, ensuring a prompt response to potential collisions and enhancing the safety of the vehicle in close proximity to obstacles.

The vehicle's propulsion relies on a single high-torque DC motor, effectively driving its forward and backward motion. The gear connected to this motor is meticulously aligned with the gears positioned on the shaft, which serves as the link between the

motor and the rear wheels. This precise alignment ensures a smooth power transmission, enabling the vehicle to move seamlessly in both directions.

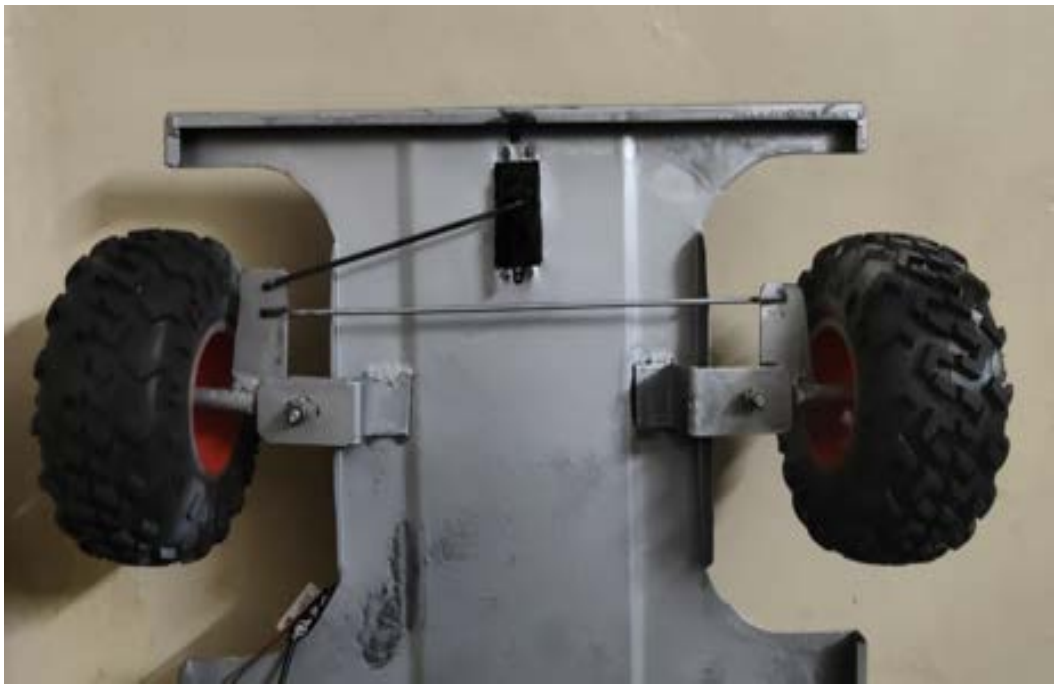


Fig 4.5 Ackermann Steering mechanism

To facilitate steering, the vehicle incorporates an Ackermann steering mechanism. This mechanism allows for accurate and responsive left and right steering by utilizing a high torque servo motor. The high torque servo motor serves as the controlling force behind the Ackermann steering mechanism, enabling precise control and maneuverability of the vehicle.

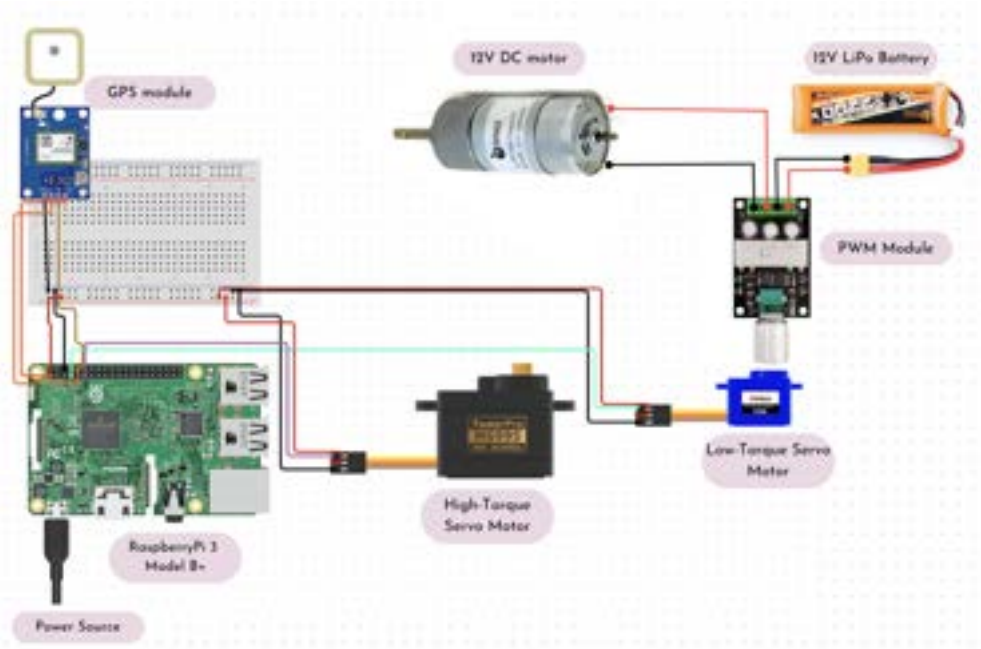


Fig 4.6 Circuit Diagram

The speed control of the vehicle is entrusted to a mini servo, working in conjunction with a PWM (Pulse Width Modulation) module. The mini servo is securely mounted and connected to the knob of the PWM module. Through the rotation of the mini servo, the position of the knob on the PWM module is adjusted. This interaction between the mini servo and the PWM module effectively modulates the speed of the vehicle.

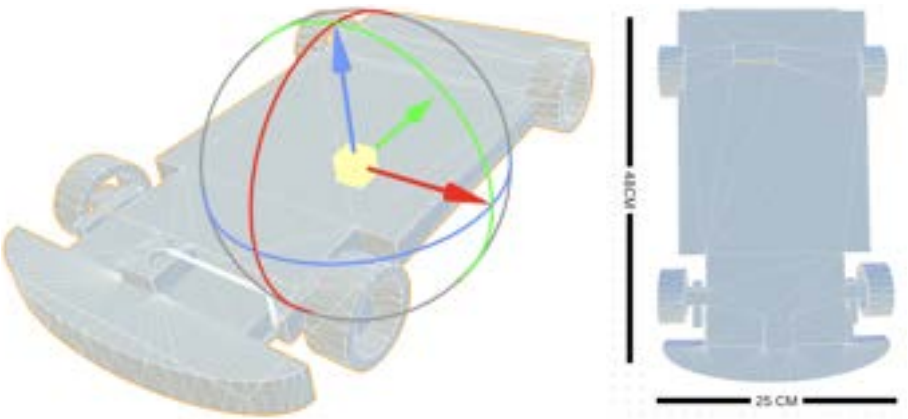


Fig 4.7 CAD model design for chassis

The hardware chassis of the vehicle was meticulously designed and constructed to optimize its aerodynamic stability and accommodate a range of crucial components. This custom-built approach aimed to minimize air resistance and maximize the vehicle's efficiency in traversing different terrains. By tailoring the chassis specifically to the vehicle's needs, engineers sought to enhance its overall performance and energy efficiency.

A key aspect of the custom-built chassis lies in its spacious configuration, which allows for the seamless integration of essential components. Considerations were made to ensure the seamless integration of wheels, gears, battery, mini servo, high torque motor, ESP8266, three ultrasonic sensors, a PWM module, and a Raspberry Pi 4 within the chassis. great care is taken to ensure that these components fit perfectly within the allocated space, allowing for their efficient placement and optimal functionality within the chassis.

Another critical consideration was the accommodation of the battery within the chassis. As a primary power source, the battery needed to be securely housed and easily accessible. The spacious design of the chassis allows for safe installation and convenient replacement when required, without compromising the vehicle's stability and structural integrity.

CHAPTER 5
TESTING AND RESULTS

To perform the testing with respect to software implementation GPS joystick have been used. This application is able to virtually update the live location of the native location and thereby replicate the motions of the vehicle to monitor map navigation and rerouting aspects.

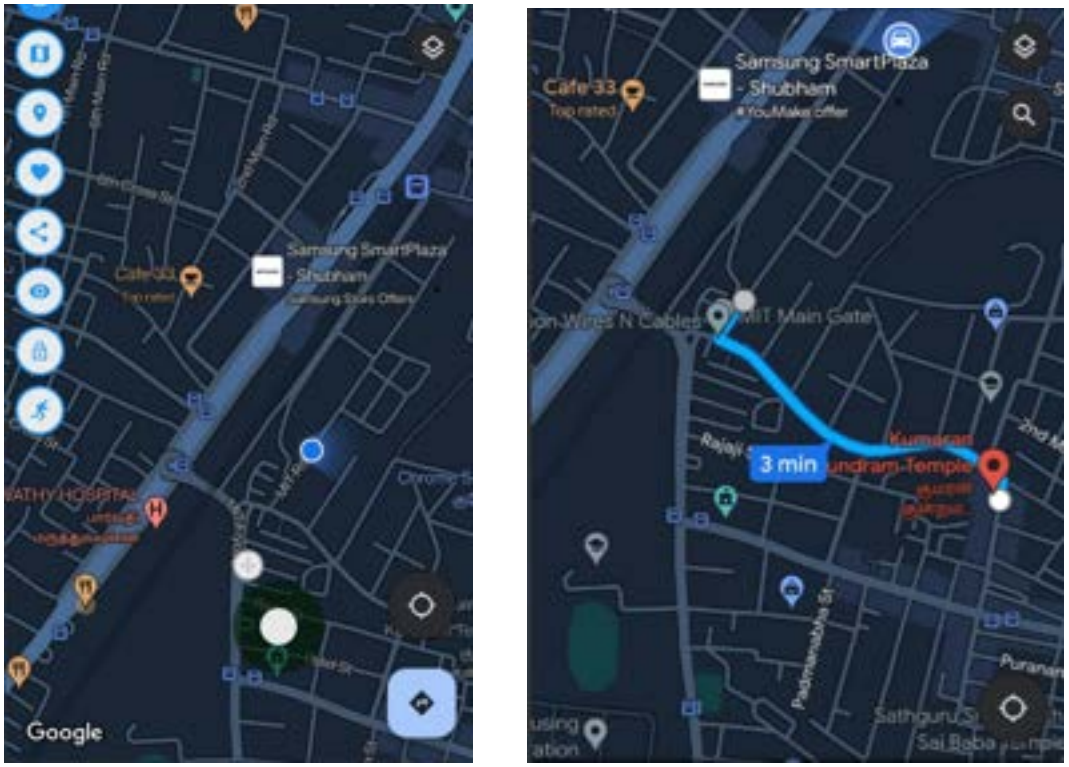


Fig 5.1 GPS joystick interface-Initial location

In this scenario, the starting point is located near the main gate of MIT, while the destination point is represented by a red marker. The path finding algorithm employed in this context provides an optimal travel path from the starting point to the designated destination

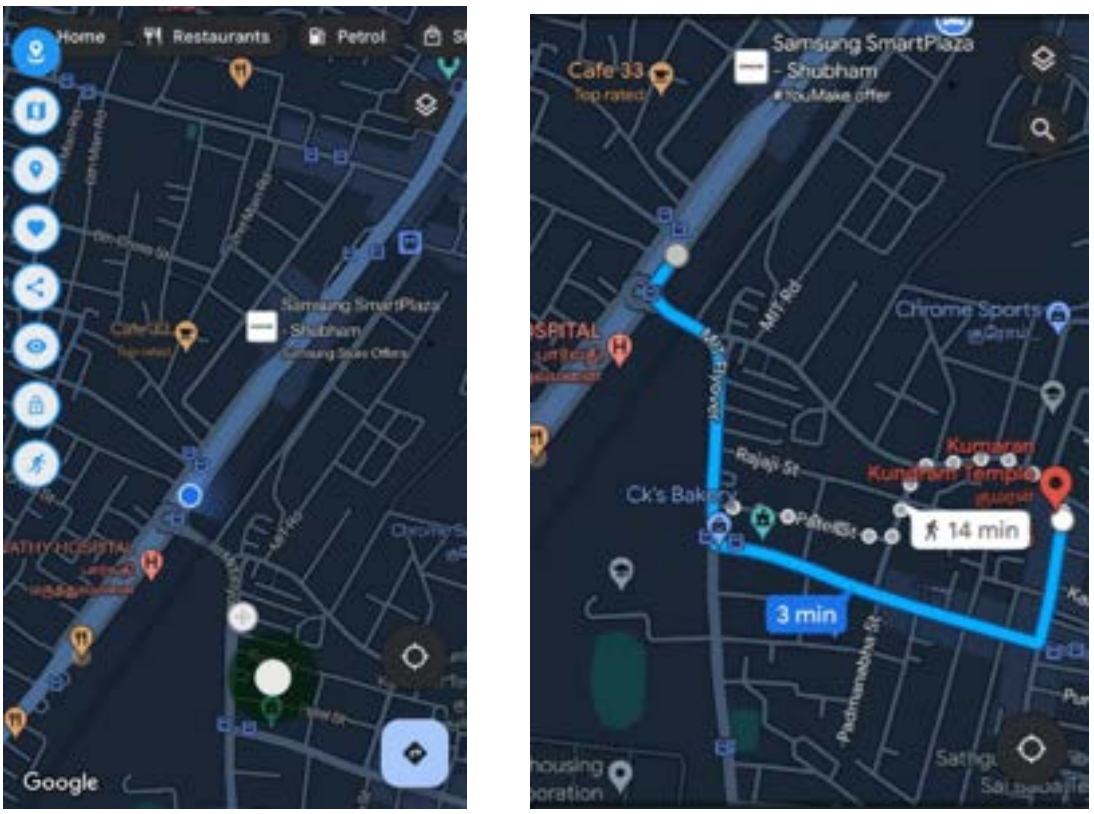


Fig 5.2 GPS joystick interface-Mimicked location

These two pictures represent the sample images which were obtained from using the GPS joystick. Features such as motor instruction set verification and rerouting algorithm have been tested using that software.

Further to test the remaining algorithms and features the accuracy and computational time has been performed for individual features post the implementation of the vehicle.

Feature\time	**Computational time**	**location**
Traffic Sign Detection	0.905 seconds	Server
Traffic Light Detection	0.114 seconds	Server
Road Detection	0.0135 seconds	Native

Lane Detection	0.289 seconds	Native
Importing dependency	43.4 seconds	Both Native and Server
Declaring functions and Global Parameters	47.7 seconds	Both Native and Server
Initial input processing time rerouting	10.1 seconds	Native

Table 5.1 Computational Analysis

With respect to hardware testing performance analysis has been carried out in various terrains and verification of algorithm implementation has been done. The response of the vehicle towards interventions of instant obstacles, traffic signs etc. have been verified.

User given commands for motor navigation using speech recognition library the output for the mentioned feature is depicted below.

```
input from the user : take me from madras institute of technology to
college of engineering guindy
```

Fig 5.3 Getting user input

Extracting needful information from raw user input is done and the obtained results are shown below.

```
starting user location:   madras institute of technology
destination location:   college of engineering guindy
```

Fig 5.4 Extracting input

Based on the information corresponding latitude longitude of the location are primarily extracted.

```
madras institute of technology  :  [12.948048, 80.139742]
college of engineering guindy  :  [13.01094, 80.2354462]
```

Fig 5.5 Latitude and Longitude details

Further implementation of user interactable instruction for easy understanding and its corresponding metrics have been analyzed.

The instruction set to reach the destination from the current position is given below:

10.030854190100001 miles
0.3503613914 hours

directions

	distance	duration	type	instruction	name	way_points	exit_number
0	11.0	2.0	11	Head southeast	-	[0, 1]	NaN
1	232.5	41.8	1	Turn right	-	[1, 6]	NaN
2	71.1	5.7	1	Turn right onto Hastthinapuram Main Road	Hastthinapuram Main Road	[6, 8]	NaN
3	306.4	24.5	0	Turn left	-	[8, 14]	NaN
4	431.5	37.9	3	Turn sharp right onto MIT Gate Bridge	MIT Gate Bridge	[14, 22]	NaN
5	1505.0	149.2	7	Enter the roundabout and take the 2nd exit	-	[22, 45]	2.0
6	38.7	4.0	12	Keep left	-	[45, 47]	NaN
7	250.5	23.8	13	Keep right	-	[47, 50]	NaN
8	595.3	54.3	7	Enter the roundabout and take the 1st exit	-	[50, 62]	1.0
9	1783.6	139.0	12	Keep left	-	[62, 89]	NaN
10	6241.9	390.5	13	Keep right onto Grand Southern Trunk Road, SH88	Grand Southern Trunk Road, SH88	[89, 159]	NaN
11	999.4	76.4	12	Keep left onto Grand Southern Trunk Road, SH88	Grand Southern Trunk Road, SH88	[159, 185]	NaN
12	151.9	13.7	6	Continue straight onto Anna Salai (Mount Road)...	Anna Salai (Mount Road), SH88	[185, 188]	NaN
13	1091.5	83.8	6	Continue straight onto Anna Salai (Mount Road)...	Anna Salai (Mount Road), SH88	[188, 202]	NaN
14	404.9	37.9	12	Keep left onto Anna Salai (Mount Road), SH88	Anna Salai (Mount Road), SH88	[202, 211]	NaN
15	181.8	19.8	13	Keep right onto Anna Salai (Mount Road), SH88	Anna Salai (Mount Road), SH88	[211, 216]	NaN
16	251.2	17.6	5	Turn slight right onto Towards Adayar, SH88	Towards Adayar, SH88	[216, 224]	NaN
17	149.9	6.4	13	Keep right	-	[224, 225]	NaN
18	7.7	2.8	0	Turn left	-	[225, 226]	NaN
19	1087.4	67.4	1	Turn right onto Taluk Office Road	Taluk Office Road	[226, 238]	NaN

Fig 5.6 User Interpretable Instructions

Representation of the 2 layered map along with the travel path has been depicted below

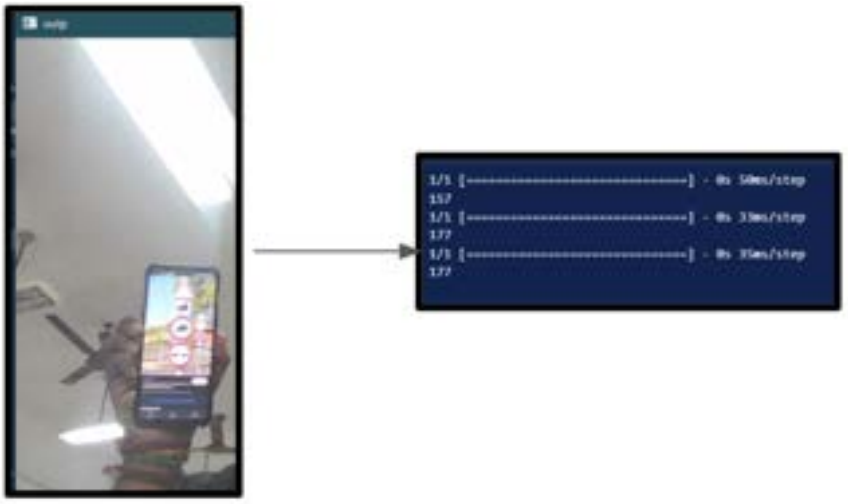
Fig 5.7 Map Checkpoints

Based on the latitude longitude details motor instruction have been developed

- ➔ Distance to be moved is 6.8322981004048255 in STRAIGHT direction with 0 degree rotation
- ➔ Distance to be moved is 26.784095694879593 in LEFT direction with 0.6352863 degree rotation

Similar instructions have been developed for the entire travel path.

Based on the camera visual the following functionalities have been implemented. The first feature of traffic sign detection has been developed with 98% accuracy.



```
Traffic sign detection time consumed for computation: 0.9056813716888428
1/1 [==============================] - 0s 39ms/step
General caution
1/1 [==============================] - 0s 42ms/step
General caution
```

Fig 5.8 Result for Traffic Sign Detection

With a frame rate of 0.4 seconds it is processing the input visuals. The rural road detection algorithm accurately calculated the regions of road and the obtained results are shown below.

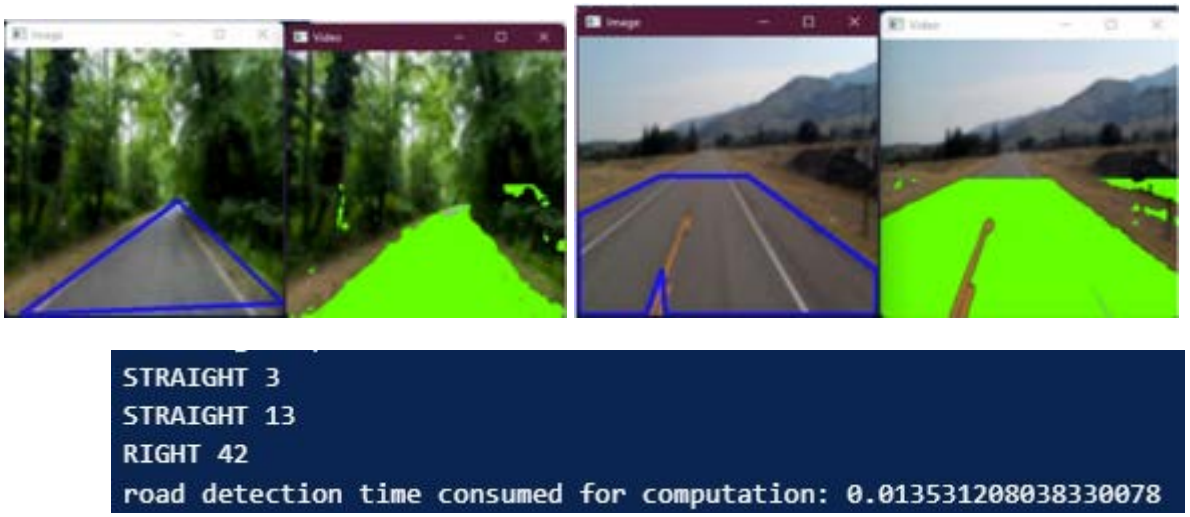


Fig 5.9 Rural Road Detection results

Further overall implementation of features such as lane detection, object detection and road detection has been implemented in both daylight and nightlight and the results obtained as shown below.

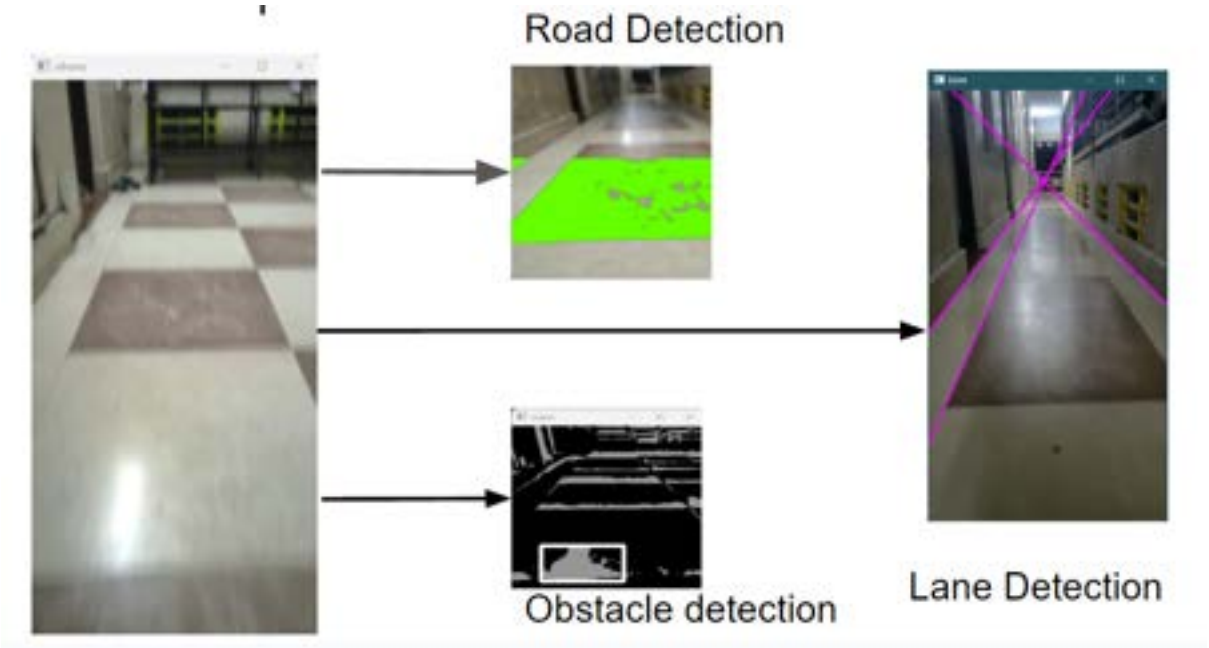


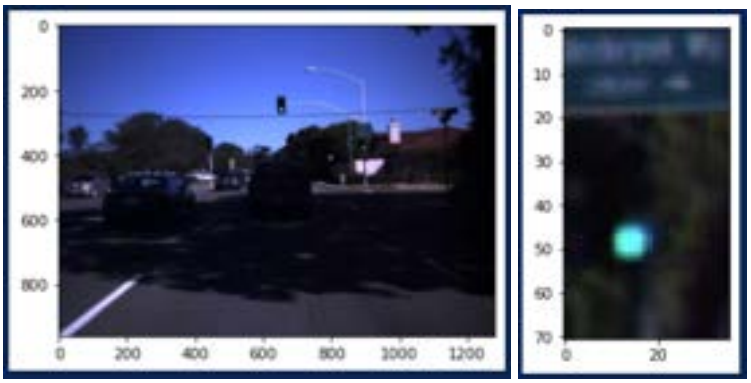
Fig 5.10 simultaneous output for software implementation

```
STRAIGHT 3
STRAIGHT 13
OBSTACLE DETECTED
Obstacle detection time consumed for computation: 0.001995086669921875
```

```
starting loop
STRAIGHT 3
Lane detection time consumed for computation: 0.325128555297851 56
Lane detection time consumed for computation: 0.2893030643463135
```

Fig 5.11 computation time for each feature

Detection of the traffic lights in both daylight and nightlight have been analyzed using varied images and the obtained results are as shown in Fig 5.8.



```
STRAIGHT 3
{'RED': 0, 'YELLOW': 0, 'GREEN': 0}
{'RED': 0, 'YELLOW': 0, 'GREEN': 0}
{'RED': 4, 'YELLOW': 0, 'GREEN': 0}
{'RED': 0, 'YELLOW': 0, 'GREEN': 0}
{'RED': 53, 'YELLOW': 0, 'GREEN': 0}
{'RED': 73, 'YELLOW': 0, 'GREEN': 0}
{'RED': 68, 'YELLOW': 0, 'GREEN': 0}
{'RED': 198, 'YELLOW': 0, 'GREEN': 0}
SIGNAL DETECTED : RED
Traffic Light detection time consumed for computation: 0.11425352096557617
{'RED': 411, 'YELLOW': 0, 'GREEN': 0}
```

Fig 5.12 Traffic Light Detection results

The complete hardware implementation of the project was successfully completed and the result is shown below.

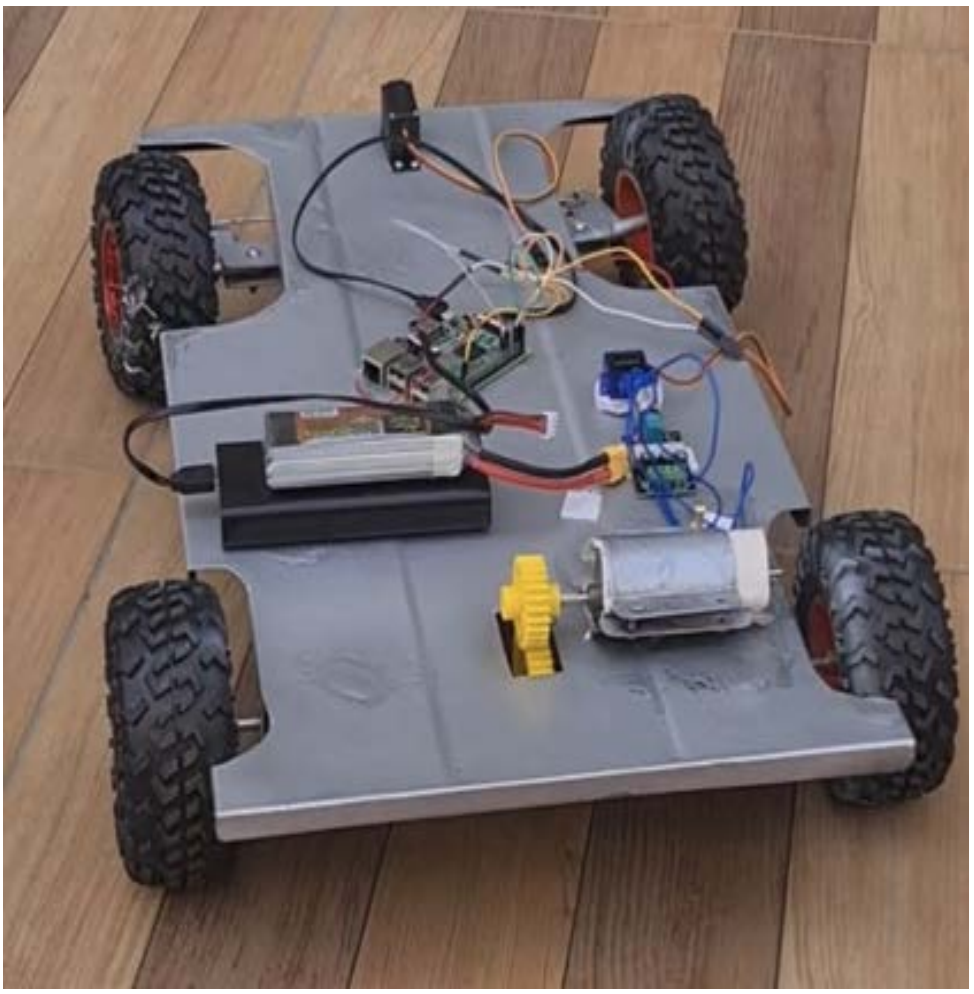


Fig 5.13 Working Model

In the Phase 2 for Autonomous vehicle a level 4 based automation is achieved with above mentioned features with model training and visualization on a laptop with specifications:

Processor: Intel(R) Core(TM) i5-10300H CPU @ 2.50GHz, 2496 Mhz

RAM: 8.00 GB

Operating System: Windows 11

CHAPTER 6
CONCLUSION AND FUTURE SCOPE

In conclusion, the project on autonomous vehicles has been successfully implemented and tested using various software and hardware components to achieve a level 4 automation system. Extensive testing was conducted to verify the accuracy and computational time of individual features, including traffic sign detection, traffic light detection, obstacle detection, road detection, and lane detection. The results demonstrated the effectiveness of the implemented algorithms in various scenarios.

The project also integrated user input through speech recognition and developed user-interpretable instructions for seamless navigation to the desired destination. By extracting latitude and longitude details, motor instructions were generated to guide the vehicle along the designated travel path. Computer vision functionalities played a crucial role in the project, with successful implementations of traffic sign detection, rural road detection, traffic light detection, and lane detection. These features demonstrated high accuracy and real-time processing capabilities, enabling the vehicle to make informed decisions based on visual cues. The hardware implementation of the project resulted in a fully functional autonomous vehicle model. The system performed well in response to instant obstacles, traffic signs, and varying lighting conditions, showcasing its robustness and reliability.

As future scope complete implementation of level 5 automation with implementations of reflex features to the vehicle and giving the vehicle the robust nature for it to travel.

REFERENCES

[1] C. Howden, “U.S department of transportation issues advance notice of proposed rulemaking to begin implementation of vehicle to vehicle communications technology.”https://www.nhtsa.gov/pressreleases/us-department -transportation-issues-advance-noticeproposed-rulemaking-begin , 2014.

[2] Con Cronin, Andrew Conway, Joseph Walsh, State-of-Art Review of Autonomous Intelligent Vehicle, ISSC (June 2019)

[3] M. V. Rajasekhar and A. K. Jaswal, "Autonomous vehicles: The future of automobiles," 2015 IEEE International Transportation Electrification Conference (ITEC), 2015, pp. 1-6, doi: 10.1109/ITEC-India.2015.7386874.

[4] Autonomous Car Market By Type (Semi-Autonomous, Fully Autonomous), By Vehicle Type (Commercial Vehicles, Passenger Cars), By Application (Civil, Defense, Transportation & Logistics, Construction), By Component (Camera Unit, LiDAR, Radar Sensor, Ultrasonic Sensor, Infrared Sensor), By Level of Automation (Level 1, Level 2, Level 3, Level 4, Level 5), Global Report by Size, Industry Analysis & Forecast, 2021 - 2030., 2022

[5] Todd Litman, Autonomous Vehicle Implementation Predictions Implications for Transport Planning, Victoria Transport Policy Institute, 2022

[6] Rasheed Hussain, and Sherali Zeadally, “Autonomous Cars: Research Results, Issues and Future Challenges” IEEE Communications, September 2018.

[7] Saeed Asadi Bagloee, Madjid Tavana, Mohsen Asadi and Tracey Oliver, “Autonomous vehicles : challenges, opportunities, and future implications

for transportation policies", https://link.springer.com/article/10.1007/s40534-016-0117-3 , Journal of Modern Transportation , (2016)

[8] Richard Sun, Michael Nehmad, Jenn Hu,Why autonomous vehicles need a large-systems approach to safety, https://www.weforum.org/agenda/2021/06/autonomous-vehicles-safety-large-systems-approach/ , 2021

[9] Rasheed Hussain, and Sherali Zeadally, "Autonomous Cars: Research Results, Issues and Future Challenges" IEEE Communications, September 2018.

[10] Hiral Thadeshwar, Vinit Shah, Mahek Jain, Prof. Rujata Chaudhari, Prof. Vishal Badgujar "Artificial Intelligence based Self-Driving Car" 4th International Conference on Computer, Communication and Signal Processing (ICCCSP) 2020.

[11] Abdur R. Fayjie, Sabir Hossain, Doukhi Oualid, and Deok-Jin Lee, "Driverless Car: Autonomous Driving Using Deep Reinforcement Learning In Urban Environment" 15th International Conference on Ubiquitous Robots (UR) Hawaii Convention Center, Hawai'i, USA June 2018.

[12] Qi-Chao Mao, Hong-Mei Sun, Yan-Bo Liu, Rui-Sheng Jia, "Mini-YOLOv3: Real-Time Object Detector for Embedded Applications", IEEE Access , 2019

[13] A. Vidyavani , K. Dheeraj, M. Rama Mohan Reddy, KH. Naveen Kumar, "Object Detection Method Based on YOLOv3 using Deep Learning Networks ",International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN: 2278-3075 (Online), Volume-9 Issue-1, November 2019.

[14] Manjiri Bichkar, Suyasha Bobhate, Prof. Sonal Chaudhari, "Traffic Sign Classification and Detection of Indian Traffic Signs using Deep Learning", International Journal of Scientific Research in Computer Science Engineering and Information Technology,2021

[15] Kolachalama Venkata Naga Sreya, “Traffic Sign Classification Using CNN”, International Journal for Research in Applied Science and Engineering Technology 10.22214/ijraset.2021.37700, 2021

[16] Shao, F., Wang, X., Meng, F., Rui, T., Wang, D., & Tang, J. (2018). “Real-time traffic sign detection and recognition method based on simplified Gabor wavelets and CNNs”, Sensors, 18(10), 3192, 2018.

[17] Der-Horng Lee, Hao Wang, Ruey Long Cheu, & Siew Hoon Teo, "A Taxi Dispatch System Based on Current Demand and Real-Time Traffic Conditions", Submitted to TRB Committee A1C05 -Transportation Network Modeling for presentation at the 82nd Annual meeting of the Transportation Research Board and consideration of publication in Transportation Research Record (2003).

[18] A planetary-scale platform for Earth science data and analysis https://earthengine.google.com/

[19] Yi-zhou Chen, Shi-fei Shen, Tao Chen, Rui Yang, “Path Optimization Study for Vehicles Evacuation Based on Dijkstra algorithm” International Conference on Performance-based Fire and Fire Protection Engineering, Wuhan (ICPFFPE 2013), 2013.

[20] Raper. J, Gartner. G, Karimi. H, Rizos. C, “A critical evaluation of location based services and their potential”, J. Locat. Based Se.rv. 2007

[21] Wang S, Zhong E, Li K, Song G, Cai W, “A Novel Dynamic Physical Storage Model for Vehicle Navigation Maps”, ISPRS Int. J. Geo-Inf, 2016.

[22] A diagram illustrating great-circle distance between two antipodal points on a sphere, https://en.wikipedia.org/wiki/Great-circle_distance.

[23] Angélique A. Scharine, Kara D. Cave, Tomasz R. Letowski, Auditory Perception and Cognitive Performance, U.S. Army Aeromedical Research Laboratory, 2009

[24] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie. "Feature pyramid networks for object detection", In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2117–2125, 2017.

[25] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. arXiv preprint arXiv:1506.01497, 2015.

[26] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning, 2017.

[27] Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. YOLOv3: An Incremental Improvement. arXiv 2018, arXiv:1804.02767.

[28] Rong Guo, Shixin Li and Kun Wang, "Research on YOLOv3 algorithm based on darknet framework", 2020 2nd International Conference on Applied Machine Learning and Data Science (ICAMLDS 2020) 21-23 August 2020, Chengdu, China, 2020

[29] Redmon, J. Divvala, S. Girshick, R. Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 2016.

[30] Soumali Roychowdhury, Michelangelo Diligenti, Marco Gori, “Image Classification Using Deep Learning and Prior Knowledge”, The Workshops of the Thirty-Second AAAI Conference on Artificial Intelligence, 2018

[31] Rahul Chauhan, Kamal Kumar Ghanshala, R.C Joshi, Convolutional Neural Network (CNN) for Image Detection and Recognition, 2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC).

[32] Engineering Applications of Bio-Inspired Artificial Neural Networks, International Work-Conference on Artificial and Natural Neural Networks, IWANN '99, Alicante, Spain, June 2-4, 1999, Proceedings, Volume II.

[33] G. Bharath Kumar, N. Anupama Rani, “TRAFFIC SIGN DETECTION USING CONVOLUTION NEURAL NETWORK A NOVEL DEEP LEARNING APPROACH”, International Journal of Creative Research Thoughts (IJCRT), ISSN:2320-2882, Vol.8, Issue 5, May 2020.

[34] Ouyang, Z.; Niu, J.; Liu, Y.; Guizani, M. Deep CNN-based Real-time Traffic Light Detector for Self-driving Vehicles. IEEE Trans. Mob. Comput. 2019, 19, 300–313

[35] Ravikiran A S, A Guide to Speech Recognition in Python: Everything You Should Know, 2022

[36] Yuan Wu, Keke Geng, Peiling Xue, Guodong Yin, Ning Zhang, Yiheng Lin, “Traffic Lights Detection and Recognition Algorithm Based on Multi-feature Fusion”, 2019 IEEE 4th International Conference on Image, Vision and Computing (ICIVC),2019

[37] Mehdi Masmoudi, Hakim Ghazzai, Mounir Frikha, and Yehia Massoud “Autonomous CarFollowing Approach Based on Real-time Video Frames

Processing" IEEE International Conference on Vehicular Electronics and Safety (ICVES), 2019.

[38] Irfan, Karunakar Pothuganti "Design & implementation of real time autonomous car by using image processing & IoT " International Conference on Smart Systems and Inventive Technology (ICSSIT 2020).

[39] JIN Zhao-zhao and ZHENG Yu-fu, "Research on Application of Improved YOLO V3 Algorithm in Road Target Detection", The 2020 International Conference on Mechatronics Technology and Intelligent Manufacturing (ICMTIM) 2020 28-30 August 2020, Xi'an, China, 2020

[40] Kim, J.; Cho, H.; Hwangbo, M.; Choi, J.; Canny, J.; Kwon, Y.P. Deep Traffic Light Detection for Self-driving Cars from a Large-scale Dataset. In Proceedings of the 2018 21st International Conference on Intelligent Transportation Systems (ITSC), Maui, HI, USA, 4–7 November 2018; pp. 280–285.

[41] D. Draheim, C. Lutteroth, G. Weber, "A source code independent reverse engineering tool for dynamic Web sites", IEEE Ninth European Conference on Software Maintenance and Reengineering, 2005.

[42] Abhishek Singh, Raspberry Pi 4 Specifications Pin Diagram and Description, https://www.hackatronic.com/raspberry-pi-4-specifications-pin-diagram-and-description/,2021

[43] Hyun-Koo Kim, Young-Nam Shin, Sa-gong Kuk, Ju H. Park, and Ho-Youl Jung, "Night-Time Traffic Light DetectionBased On SVM with Geometric MomentFeatures", World Academy of Science, Engineering and Technology International Journal of Computer and Information Engineering, Vol:7, No:4, 2013

[44] Ahmed Amirou, Djaffar Ould Abdeslam, Zahia Zidelmala, Mohammed Aidene, Jean Merckle. Optimizing the classification cost using SVMs with a double hinge loss. . Informatica, Slovene Society Informatika, Ljubljana, 2014.

[45] D. R. Niranjan, B. C. VinayKarthik and Mohana, "Deep Learning based Object Detection Model for Autonomous Driving Research using CARLA Simulator," 2021 2nd International Conference on Smart Electronics and Communication (ICOSEC), Trichy, India, 2021

[46] S. B. Prathiba, G. Raja, K. Dev, N. Kumar and M. Guizani, "A Hybrid Deep Reinforcement Learning For Autonomous Vehicles Smart-Platooning," in IEEE Transactions on Vehicular Technology, Dec. 2021,

[47] https://www.kaggle.com/datasets/manideep1108/culane

[48] ***Jiwoong Choi, Dayoung Chun, Hyun Kim, Hyuk-Jae Lee***; Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 2019, pp. 502-511