SELF DRIVING ROBOT CAR

Submitted in partial fulfilment of the requirements

of the Degree of

Bachelors in Engineering

By

Nidhi Mundhada	**D19B_58**
Shrutika Patel	**D19B_60**
Aditi Patil	**D19B_61**
Vaishnavi Shetty	**D19B_66**

Supervisor

Dr. (Mrs.) Saylee Gharge

Department of Electronics and Telecommunication Engineering

Vivekanand Education Society's Institute of Technology

University of Mumbai

Academic Year 2022-2023

Project Approval for B.E

Project entitled **Self Driving Robot Car** by Nidhi Roshan Mundhada (58), Shrutika Saket Patel (60), Aditi Kishor Patil (61), Vaishnavi Harish Shetty (66) is approved for the degree of Bachelor of Engineering.

Examiners

1. _____

2. _____

Supervisors

1. _____

2. _____

Date: 14th April, 2023

Place: Chembur, Mumbai

Certificate

This is to certify that Nidhi Roshan Mundhada (58), Shrutika Saket Patel (60), Aditi Kishor Patil (61), Vaishnavi Harish Shetty (66) have completed the project report on the topic **Self Driving Robot Car** satisfactorily in partial fulfilment of the requirements for the Bachelor's Degree of Engineering in Electronics and Telecommunication under the guidance of Dr. (Mrs.) Saylee Gharge during the **year 2022-2023** as prescribed by University of Mumbai.

_____ _____

Guide **Head of the Department**

Dr. (Mrs.) Saylee Gharge **Dr. Chandansingh Rawat**

_____

Principal

Dr. (Mrs.) Jayalekshmi Nair

_____ _____

Examiner 1 **Examiner 2**

Declaration

We declare that this written submission represents our ideas in our words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misinterpreted or fabricated or falsified any idea/data/fact/source in my submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Nidhi Mundhada

Shrutika Patel

Aditi Patil

Vaishnavi Shetty

Date: 14th April, 2023

Table of Contents

Abstract

Self-driving cars are automobiles that do not require human operation to navigate to a destination. There are several advantages claimed about driverless vehicles, though they are still being proven out. A few advantages brought forth are safer streets, greater environmental benefits and better access for people with disabilities. They use cameras, sensors, and advanced software to interpret and respond to traffic, pedestrians, and other surroundings on the road. To qualify as fully autonomous, a vehicle must be able to navigate without human intervention to a predetermined destination over roads that have not been adapted for its use. In this project, the design of Self Driving Robot Car has been implemented. This Robot Car will be designed so as to detect the lane lines. Along with the detection of lane lines, stop signs, traffic lights and Obstacle detection will be accomplished. The methodology to detect lane lines is discussed thoroughly as well as the steps taken to implement the above-mentioned techniques are mentioned in detail.

List of Figures

List of Tables

Abbreviations

CNN	Convolutional Neural Network
CSI	Camera Serial Interface
DC	Direct Current
FPS	Frames Per Second
GPU	Graphics processing units
I/O	Input / Output
IDE	Integrated Development Environment
IEEE	Institute of Electrical and Electronics Engineers
ROI	Region Of Interest
RPI	Raspberry Pi
RPM	Rotations Per Minute
UART	Universal Asynchronous Receiver/Transmitter

Chapter 1

Introduction

Introduction

1.1 Need for Self-Driving Cars

1.2 Proposed Solution

Introduction

Globally speaking, nearly 1.3 million people die in road crashes each year, on average of 3,287 deaths a day and talking about India the number of people who were killed in road accidents in 2013 alone was 1, 37,000 [1]. Speeding, talking over the phone, drunk driving, and breaking traffic rules are the root causes behind these accidents and the statistics are rising daily, which is now becoming a significant concern. No matter how hard we try to create awareness regarding traffic rules and safety that has to be followed while driving, accidents are still occurring and aren't showing a sign to stop. Though human errors can never be eliminated, accidents can definitely be prevented. Human-driven cars use technologies to provide safety and detect obstacles and auto stops in various high-end cars but none of them works completely driverless. The existing cars do not contain the feature of automation to the extent that cars can drive autonomously. There is a constant need for drivers, without it the car becomes unavailable but with self-driving cars, we can make the availability of cars constant on roads. In traditional cars, the driver constantly needs to keep a check on the signals, road safety signs, obstacles, and lanes and needs to make decisions accordingly.

1.1 Need for Self-Driving Cars

In an ideal world, automated vehicles will operate on the roads without human assistance and increase their safety by reducing road accidents. One of the major benefits of self-driving technology is making transportation more accessible for those who cannot drive. Many people cannot drive for various reasons, including age, disability, or lack of experience. Automated vehicles can be a much safer mode of transportation for these people, allowing them greater independence. Many people cannot drive for various reasons, including age, disability, or lack of experience. Automated vehicles can be a much safer mode of transportation for these people, allowing them greater independence. Also, traffic congestion is a huge problem in cities around the world. It wastes time and fuel, and it's a significant contributor to air pollution. Fully autonomous vehicles could help reduce traffic congestion and various other problems.

1.2 Proposed Solution

To build the robot car, a Raspberry Pi as the master device and an Arduino Uno as the slave device were used. The image/video processing will be handled by Raspberry Pi, and the controlling commands will be sent to the Arduino via Raspberry pi. To make the car move, wheels (motors) will be attached to it, which will be controlled by Arduino Uno using Raspberry pi commands. The project aims to build a car that, when given power, can detect lanes using video processing and further stop signs, traffic signs, and objects on its own using neural network training (machine learning methodologies).

Chapter 2
Review of Literature

2.1 Referred Papers

2.1.1. Real-time traffic sign detection and recognition using Raspberry Pi.

2.1.2. Real-time Lane detection and Motion Planning in Raspberry Pi and Arduino for an Autonomous Vehicle Prototype

2.1.3. Artificial Intelligence based Self-Driving Car

2.1.4. Convolution Neural Network based Working Model of Self-Driving Car - A study

2.1.5. Object detection in autonomous driving - from large to small datasets

2.1.6. Road Sign Recognition System for Autonomous Vehicle using Raspberry Pi

2.1.7. Working model of Self-driving car using Convolutional Neural Network Raspberry Pi and Arduino

2.1.8. Self-Driving Robot using Neural Network

2.2 Summary of Literature Review

Review of Literature

The necessity of Self Driving Robot Cars was discussed in the previous chapter. In this chapter, implementation of Self Driving Robot Car done by various researchers will be discussed thoroughly. Following are the papers which were referred during the implementation of the project – Self Driving Robot Car.

2.1 Referred Papers

2.1.1. Real-time traffic sign detection and recognition using Raspberry Pi [2]

The paper describes the implementation of road sign detection and notifies the driver about the type of sign detected via a speaker using TensorFlow Machine Learning algorithm. In this work, a Raspberry Pi 3 module is used to run the TensorFlow machine learning algorithm and is connected to a Raspberry Pi camera NoIR to record the real-time video, a Raspberry Pi Display module to display the related information of the detected traffic sign and a speaker to give an alert sound to alert the driver. The system consists of Raspberry Pi 3 processor and web camera which automatically captures the video data and converts them into a number of frames which are processed by the proposed algorithm in OpenCV to detect the road sign and control the vehicle. Based on the detected sign, the vehicle is controlled by two DC motors interfaced with Raspberry Pi.

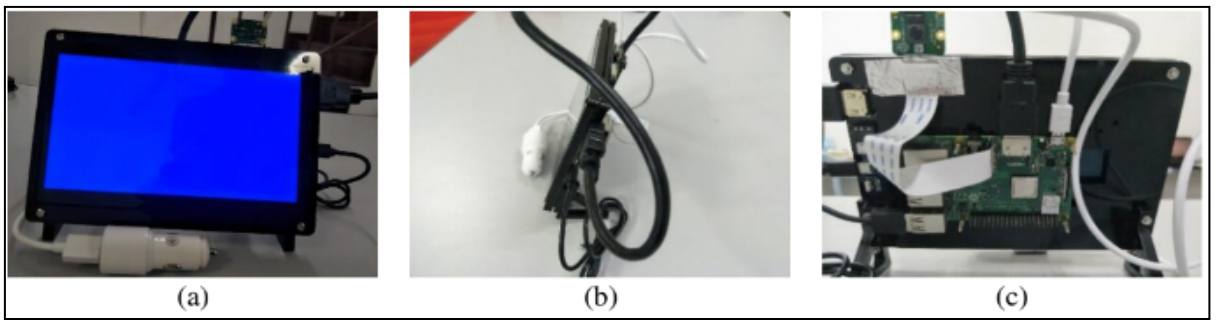

Fig.2.1 The developed real-time traffic sign detection and recognition system; (a) front view, (b) side view and (c) rear view [2]

Fig.2.1 shows the hardware implementation of the proposed system. The front view, side view and the rear view is shown. The camera will be attached to the car such that it will capture the outside environment and detect proper signs to notify the driver. The working principles of the developed real-time traffic sign recognition system are as shown in Fig.2.2 First, the Raspberry Pi 3 and Raspberry Pi camera NoIR are in stand-by mode once the system is powered on. Then, the Raspberry Pi camera NoIR will start recording the real-time video and share the video with the Raspberry Pi 3 synchronously. The Raspberry Pi 3 will process the video to detect the type of traffic sign. If a traffic sign is detected, the results will be sent to the Raspberry Pi Display Module to display the type of the detected traffic sign and

activate the speaker. Otherwise, the camera will keep on recording and sending the data to the Raspberry Pi 3 module for processing until the traffic sign is detected.

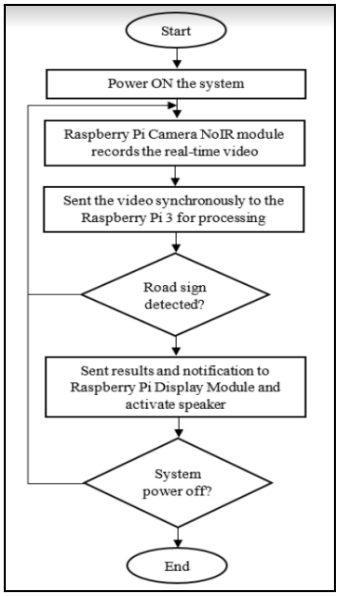

Fig.2.2 The flow system of the real-time traffic sign detection and recognition system algorithm [2]

The authors of the paper have used labeling. Labeling is also known as label image, which is an open-source image labelling tool that is used to label the size of the traffic sign image dataset. Each traffic sign image needs to be labelled before starting the training process of the traffic sign images with TensorFlow machine learning software. The purpose of labelling the traffic sign images is to determine the location of the traffic signs. Besides, labeling carries out the process of segmentation of traffic sign images and then continues with the process of annotation and interpretation for the traffic sign images. Annotation of traffic sign images is important as the bounding outside boxes will be shown on the traffic sign images, thus the images will be easily recognized. The coding used for Labeling is Python programming language. After Labeling annotates the sample set of traffic sign images, it saves the images as 'XML' file format, and the images are ready to be trained and tested by TensorFlow.

In this work, the TensorFlow machine learning algorithm is used to train a dataset consisting of five different classes of traffic signs, including the Speed Bump, Stop, Give Way, No U-Turn and Chevron Alignment. These five classes of traffic signs are considered as they are commonly found at the roadside. For each class, there are 100 sample images with different angles, and size hence give a total of 500 sample images. The training process is important to prepare the pre-trained model before the implementation of the real-time recognition system. The pre-trained model that contained the size, colour, shape and boundary of the sample traffic sign images is used for the recognition. The performance of the developed real-time traffic sign detection and recognition system in terms of accuracy,

reliability and delay has been evaluated considering five different classes of traffic signs, the state of the environment and the condition of the traffic sign. The developed system is installed in a car, and the speed of the car considered in this experiment is 50 kilometres per hour (50 km/h). Twenty traffic sign images from five different classes of traffic sign have been considered for the real-time testbed implementation. The results show that the average accuracy of detecting and identifying traffic sign images from the five traffic sign classes is above 90%. Meanwhile, the results also show that the maximum average delay in determining the type of traffic sign in the system is 3.44 seconds when the car moves at 50 km/h. This shows that the system is suitable to be used for a city speed limit of the car. In terms of reliability, the developed system is tested considering the condition of the traffic sign such as broken and blurred traffic sign and the state of the environment (i.e., during night-time). The results indicate that the developed system is reliable to detect the type of traffic sign with 90% accuracy for all considered conditions and send out an alert to the drivers.

2.1.2. Real-time Lane detection and Motion Planning in Raspberry Pi and Arduino for an Autonomous Vehicle Prototype [3]

This paper discusses a vehicle prototype that recognizes streets' lanes and plans its motion accordingly without any human input. Pi Camera 1.3 captures real-time video, which is then processed by Raspberry-Pi 3.0 Model B. The image processing algorithms are written in Python 3.7.4 with OpenCV 4.2. Arduino Uno is utilized to control the PID algorithm that controls the motor controller, which in turn controls the wheels. Algorithms that are used to detect the lanes are the Canny edge detection algorithm and Hough transformation. Elementary algebra is used to draw the detected lanes. After detection, the lanes are tracked using the Kalman filter prediction method. Then the midpoint of the two lanes is found, which is the initial steering direction. The initial steering direction is further smoothed by using the Past Accumulation Average Method and Kalman Filter Prediction Method. The prototype was tested in a controlled environment in real-time. Results from comprehensive testing suggest that this prototype can detect road lanes and plan its motion successfully.

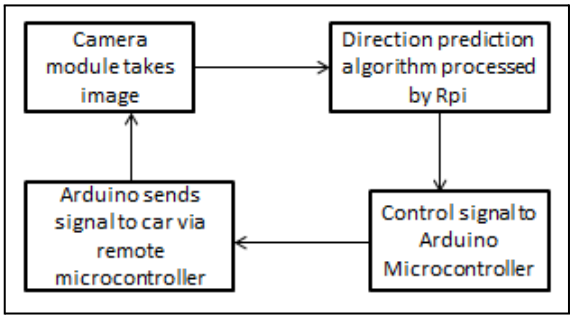

Fig.2.3 Block Diagram of Self Driving Car [3]

Fig.2.3 shows the block diagram of the Self Driving Car. Raspberry pi is the primary processing unit of their system. Image frames captured by Pi cam go through different algorithms which helped them to find out the lane markers. The python OpenCV library is used to implement the image processing algorithms. Then, the current position of the prototype is determined, and the steering direction is planned. PID algorithm controls the motor speed and consequently controls the direction of motion.

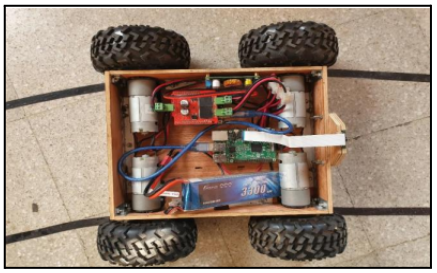

Fig.2.4 Hardware Implementation for Real-time Lane detection and Motion Planning in Raspberry Pi and Arduino for an Autonomous Vehicle Prototype [3]

Fig.2.4 shows the hardware implementation for Real-time Lane detection and Motion Planning in Raspberry Pi and Arduino for an Autonomous Vehicle Prototype. The video captured by Pi Camera at the resolution 640 x 480 is resized to half of its width and height; thus, the resolution is converted to 320 x 240. The image frames of the video must be converted to a grayscale image for the upcoming stages of the presented algorithm, namely edge detection and Hough transformation. A Gaussian filter is implemented for image blurring, which removes noise. Then the canny edge detector is used for edge detection. As it can safely be assumed that the lanes will be situated at the bottom half of the frames, the top half of every frame is masked out so that no unnecessary lines in that region gets detected. The untouched region down below is the region of interest (ROI).

Fig.2.5 Actual Frame [3]

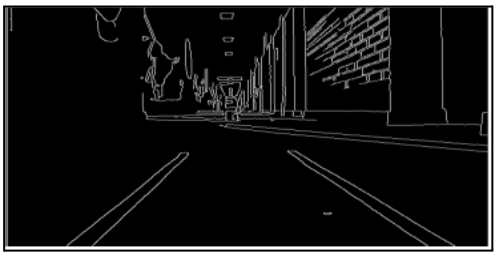

Fig.2.6 Edge Detected Frame [3]

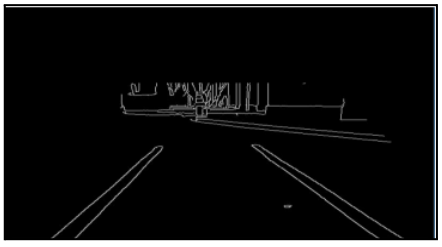

Fig.2.7 Segmented Frame [3]

Fig 2.5, 2.6, 2.7 shows the actual frame, the canny edge detected frame and the segmented frame respectively. By averaging the slope and y-intercept values found from Hough transform, all the lines of the left part are converted to a single left lane and all the lines of the right part are converted to a single right lane. The lane trackers orientation parameter is predicted by Kalman filter. Since the Pi Cam is mounted precisely at the centre of the prototype, the prototype's desired position is at the centre of the obtained frame. The setpoint is the 160th pixel of the X-axis 320 pixels, while the Y coordinate is fixed at the ceiling of ROI. This coordinate should always be the desired position of the vehicle. Initially it is also the prototype's position. When the prototype moves, the current position gets away from the setpoint. However, there are some shortcomings in finding the current position. Depending on the weather and light condition, there can be some inconsistency in this process. For example, sometimes, the current position can be determined without any complications, and sometimes the current position cannot be readily determined. Two different algorithms were used to overcome this: Past Accumulated Average Method and Kalman Filter Algorithm.

Fig.2.8 Current Position Based on Kalman Filter Prediction Method and Past Accumulated Average Method [3]

Fig.2.8 illustrates the current position based on both methods, the Kalman Filter Method and the Past Accumulated Method. The PID algorithm is implemented on the Arduino Uno. The algorithm calculates the error by subtracting the obtained current position value from time to time from the set point. Depending on the PID output, the drive function adds the error value of PID to the base value, which is 100 PWM for our case, of the left motor and subtracts it from the right or vice versa depending on the necessity. The highest PWM value is predefined and would not be exceeded. Thus, the vehicle moves smoothly without any sharp turns or extraordinarily high or low PWM and moves autonomously, maintaining the lanes. This paper implemented a very cost-efficient autonomous vehicle where we used Canny edge detection and Hough Transform to detect road lanes and Kalman filter for tracking our lane markers. The position information was fed from the Raspberry Pi to Arduino via UART communication, which controlled the motors' speed. Several tracks were made on our university campus, mimicking original streets, and our prototype completed all of those. The processing rate was six frames per second.

2.1.3. Artificial Intelligence based Self-Driving Car [4]

The paper proposes a self-driving car model also called autonomous, robotic or driver-less car that operates and navigates using its intelligence. The Pi camera and ultrasonic sensor are attached to the raspberry pi board to collect input images along with sensor data to stream these data to the server which in our case is the laptop. The (CNN) convolutional neural network running on the server is used to enable lane detection to provide steering predictions that are left, right, forward based on the input image. The Haar cascade classifier is used to detect signals and stop signs and monocular vision algorithms to calculate distance from them. The ultrasonic sensor is used for front collision avoidance by stopping the car at a certain distance before the obstacle ahead. The navigation commands such as right, left, forward, stop are sent to the car through Arduino which is connected to the RC car's remote, this makes the car drive autonomously based on neural network predictions and some hard coded rules.

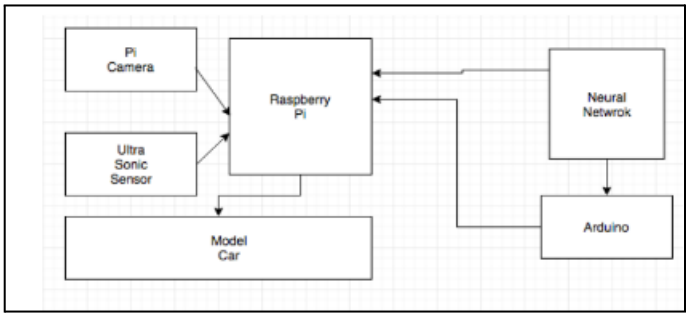

Fig. 2.9 Block Diagram of the Proposed System [4]

Fig.2.9 consists of three units. First is the input unit containing a pi camera and ultrasonic sensor, the second is the processing unit which is our laptop that will act as a server, the neural network is running over here and third, is RC control unit which is Arduino. Firstly, in the input unit, the raspberry pi board of the B+ model is connected with the raspberry pi camera and an ultrasonic sensor to stream input data. There are two client programs running on raspberry pi one to stream images collected by the pi camera and another to send sensor data through a local wi-fi connection.

The braking distance between the car and obstacle is hardcoded and set to 5.5 centimetres and the threshold for stopping is 30 centimetres. The RC car used in the prototype has an on/off or high/low switch type controller. Therefore, the Arduino board is used to imitate button press actions. There are four Arduino pins used to connect four chip pins on the rc remote controller, corresponding to forward, reverse, left and right actions respectively. The Arduino is connected to the computer through USB and the computer send the output

commands and write out low or high signals, simulating button press actions to drive the car autonomously

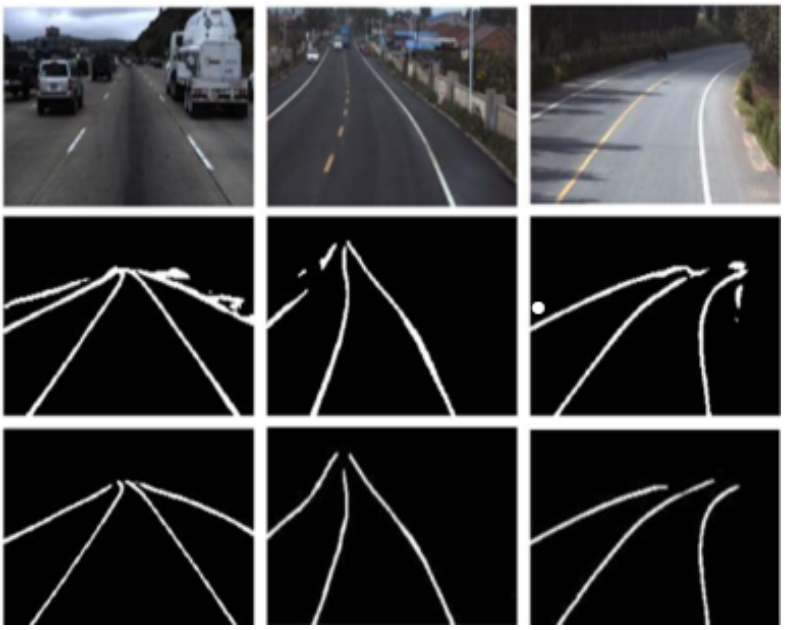

Fig. 2.10 Lane line detection using OpenCV [4]

Fig.2.10 shows the implemented results of the model. The lane lines (white lines) are detected when the model moves straight, taking a left and right turn. Using a rpi camera the images of the surroundings are also captured. Furthermore, the authors of this paper mentioned; automation in cars in real world is a very big field, where many sensors come into action, their idea is to solve many of those into two sensors by detecting distance and objects like stop sign, signals and other obstacles with a single method of monocular vision which can be enhanced in future from a scaled car to an actual car.

2.1.4. Convolution Neural Network based Working Model of Self-Driving Car - A study [5]

In this paper, four functionalities are used, namely: 1) Lane Detection, 2) Object Detection, 3) Traffic Signal Detection and 4) Signboard Detection. Convolution Neural Network (CNN) is for efficient image classification. Deep learning techniques namely Convolution Neural Networks, YOLO algorithm, Hough Transform Algorithms, Transfer Learning and Canny Edge Detection algorithm were used. CNN is used for efficient image classification for all detection purposes. Software components such as Arduino IDE, Raspberry Pi Camera Interface, Open CV, Tensor Flow, Carla simulators and hardware components such as Raspberry Pi 3, Arduino UNO, Pi Camera, sensors like radar, lidar are used to build a prototype of a self-driving car.

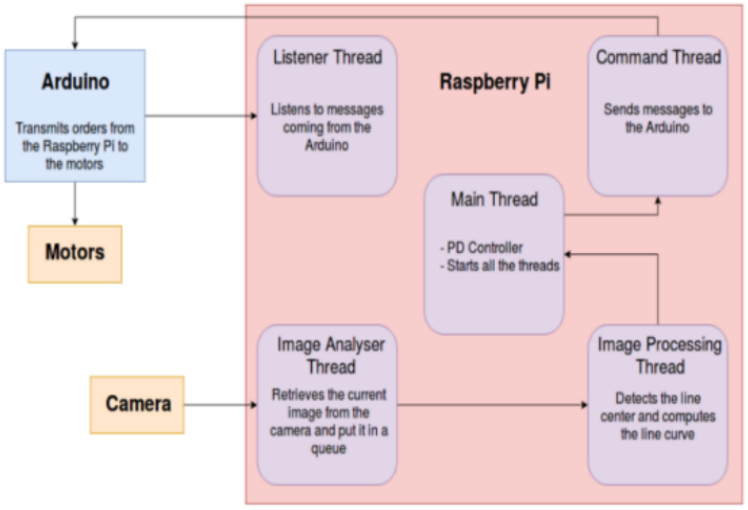

Fig.2.11 Proposed Model's Block Diagram [5]

Fig. 2.11 shows the block diagram of the proposed model. A pi camera is used for capturing live images which is sent to Raspberry pi 3 for image processing and Arduino is used to move the car around by sending signals to the motors of the cars. So, Raspberry pi 3 acts as the main device and Arduino acts as the slave device.

The main advantage of CNN compared to its predecessors is that it automatically detects the important features without any human supervision. CNN is also a very powerful and computationally efficient model which performs automatic feature extraction to achieve accuracy. But the drawback here is that to use deep learning technologies like CNN, a large amount of training data is needed. If the computer does not have a good GPU, then if CNN is made up of multiple layers, the training process could take a particularly long time.

Furthermore, Deep learning algorithms like CNN (Convolution neural networks) are employed to make immediate decisions for the self-driving car. Detection of objects, lane detection and detection of traffic signs and signals have been studied in this paper and algorithms like Faster R-CNN, YOLO and canny edge detection algorithms have been used for the same.

2.1.5. Object detection in autonomous driving - from large to small datasets [6]

In this paper, the current capacity of pedestrian and vehicle detection was compared and analyzed of four deep learning technologies namely Yolo, SSD, Faster R-CNN and Retina Net on a big dataset like Berkeley Deep Drive (BDD100K) and small dataset collected through the campus. Precision and recall were done on this big data and small data using Yolo, SSD, Retina Net and Faster R-CNN to see which is the best object detector and what improvements could be made.

Table 2.1: Precision on BDD100K dataset [6]

	DC	DO	DR	DS	DSC	DSO	DSR	DSS	NC	NO	NR	NS
Yolo AP@.50IOU	0.66	0.65	0.64	0.65	0.65	0.65	0.62	0.65	0.63	0.64	0.61	0.64
SSD AP@.50IOU	0.92	0.93	0.92	0.92	0.94	0.93	0.93	0.93	0.90	0.93	0.90	0.91
Faster R-CNN AP@.50IOU	0.84	0.86	0.86	0.88	0.86	0.86	0.87	0.89	0.82	0.86	0.82	0.83
RetinaNet AP@.50IOU	0.28	0.27	0.32	0.32	0.27	0.26	0.29	0.30	0.31	0.33	0.32	0.34
Yolo MAP	0.56	0.55	0.55	0.55	0.55	0.55	0.53	0.55	0.54	0.55	0.53	0.55
SSD MAP	0.79	0.80	0.79	0.79	0.80	0.79	0.79	0.80	0.74	0.79	0.75	0.75
Faster R-CNN MAP	0.67	0.69	0.68	0.70	0.68	0.68	0.68	0.71	0.64	0.68	0.63	0.65
RetinaNet MAP	0.34	0.34	0.37	0.36	0.34	0.33	0.35	0.36	0.37	0.39	0.37	0.38

Table 2.2: Recall on BDD100K dataset [6]

	DC	DO	DR	DS	DSC	DSO	DSR	DSS	NC	NO	NR	NS
Yolo AR@.50IOU	0.36	0.37	0.36	0.36	0.35	0.37	0.34	0.35	0.20	0.33	0.18	0.22
SSD AR@.50IOU	0.17	0.17	0.19	0.19	0.16	0.17	0.18	0.19	0.12	0.16	0.11	0.14
Faster R-CNN AR@.50IOU	0.14	0.14	0.13	0.14	0.12	0.13	0.12	0.13	0.05	0.11	0.04	0.06
RetinaNet AR@.50IOU	0.09	0.08	0.10	0.10	0.08	0.08	0.09	0.09	0.06	0.09	0.06	0.07
Yolo MAR	0.30	0.31	0.30	0.30	0.29	0.31	0.29	0.30	0.17	0.28	0.15	0.19
SSD MAR	0.14	0.15	0.16	0.16	0.14	0.14	0.15	0.16	0.10	0.14	0.09	0.11
Faster R-CNN MAR	0.11	0.11	0.10	0.11	0.09	0.10	0.09	0.10	0.04	0.09	0.03	0.05
RetinaNet MAR	0.11	0.11	0.11	0.12	0.10	0.10	0.10	0.11	0.07	0.11	0.07	0.08

Table 2.3: Precision on BDD100K dataset - only car and person [6]

	DC	DO	DR	DS	DSC	DSO	DSR	DSS	NC	NO	NR	NS
Yolo AP@.50IOU	0.72	0.72	0.74	0.72	0.72	0.72	0.73	0.73	0.72	0.74	0.73	0.76
SSD AP@.50IOU	0.92	0.93	0.92	0.92	0.94	0.93	0.93	0.93	0.90	0.93	0.90	0.91
Faster R-CNN AP@.50IOU	0.87	0.87	0.87	0.88	0.88	0.87	0.89	0.89	0.85	0.88	0.85	0.85
RetinaNet AP@.50IOU	0.28	0.27	0.32	0.32	0.27	0.26	0.29	0.30	0.31	0.33	0.32	0.34
Yolo MAP	0.60	0.60	0.62	0.60	0.60	0.61	0.61	0.61	0.60	0.62	0.60	0.62
SSD MAP	0.79	0.80	0.79	0.79	0.80	0.79	0.79	0.80	0.74	0.79	0.75	0.75
Faster R-CNN MAP	0.69	0.70	0.69	0.70	0.70	0.69	0.69	0.71	0.66	0.70	0.66	0.67
RetinaNet MAP	0.34	0.34	0.37	0.36	0.34	0.33	0.35	0.36	0.37	0.39	0.37	0.38

Table 2.4: Recall on BDD100K dataset - only car and person [6]

	DC	DO	DR	DS	DSC	DSO	DSR	DSS	NC	NO	NR	NS
Yolo AR@.50IOU	0.48	0.50	0.50	0.50	0.47	0.50	0.47	0.50	0.30	0.46	0.27	0.34
SSD AR@.50IOU	0.17	0.17	0.19	0.19	0.16	0.17	0.18	0.19	0.12	0.16	0.11	0.14
Faster R-CNN AR@.50IOU	0.20	0.21	0.20	0.21	0.17	0.19	0.17	0.21	0.09	0.16	0.08	0.10
RetinaNet AR@.50IOU	0.09	0.08	0.10	0.10	0.08	0.08	0.09	0.09	0.06	0.09	0.06	0.07
Yolo MAR	0.40	0.43	0.42	0.42	0.39	0.42	0.39	0.42	0.25	0.38	0.22	0.28
SSD MAR	0.14	0.15	0.16	0.16	0.14	0.14	0.15	0.16	0.10	0.14	0.09	0.11
Faster R-CNN MAR	0.16	0.17	0.16	0.17	0.14	0.15	0.14	0.17	0.07	0.13	0.06	0.08
RetinaNet MAR	0.11	0.11	0.11	0.12	0.10	0.10	0.10	0.11	0.07	0.11	0.07	0.08

Table 2.1 and 2.2 depicts the data of Precision and Recall done on the big data and shows which algorithm would be faster. Table 2.3 and 2.4 depicts the data of Precision and Recall done only on car and person that is the small dataset.

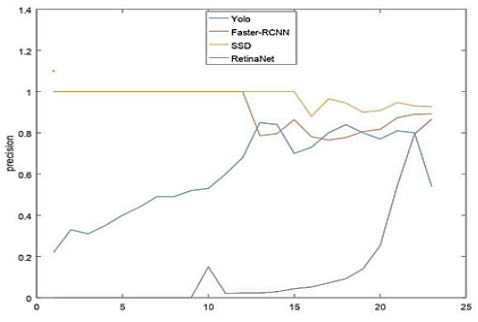

Fig.2.12 Precision comparison regarding object size [6]

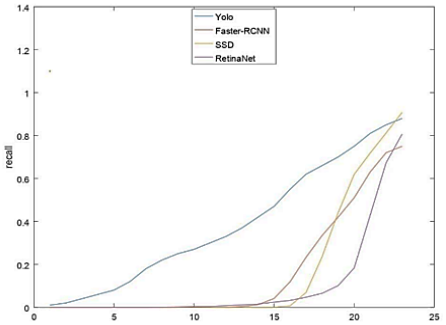

Fig.2.13 Recall comparison regarding object size [6]

Fig.2.12 and 2.13 shows the result of Precision and Recall done on object size and compares the algorithms used. After analysis it was found that Yolo has the best recall and SSD the best precision, close to Faster R-CNN but it is only useful in detecting bigger objects

like another car in the front or a person. Retina Net was outperformed by all of the detectors in both precision and recall.

Regarding the BDD dataset, the precision doesn't change too much regarding the weather and the day time changes, but it is seen that there are worse results in terms of recall at night and in the rain. The results obtained from the small dataset were similar in terms of precision but the recall was better than for the BDD100k dataset because of a lesser number of objects to be detected. SSD is only useful for detecting bigger objects. Recall on big data gives the worst result.

2.1.6. Road Sign Recognition System for Autonomous Vehicle using Raspberry Pi [7]

The project aims at implementation of road sign detection and control of an autonomous vehicle using Haar Cascade Classifier algorithm. In this proposed work, the system automatically detects the road signs, controls the vehicle and commands certain actions. The system consists of Raspberry Pi 3 processor and web camera which automatically captures the video data and converts them into a number of frames which are processed by the proposed algorithm in OpenCV to detect the road sign and control the vehicle. Based on the detected sign, the vehicle is controlled by two DC motors interfaced with Raspberry Pi.

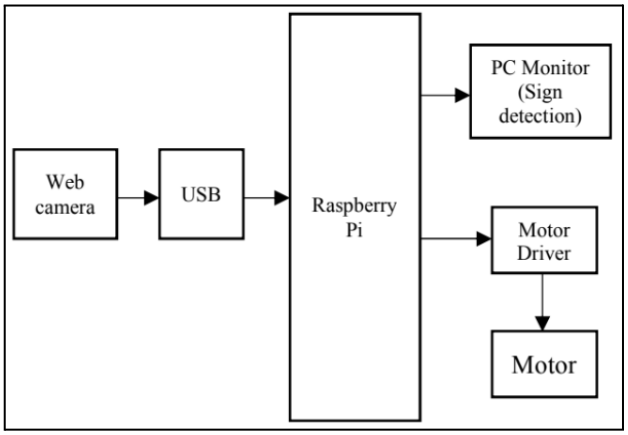

Fig. 2.14 Proposed Solution's Block Diagram [7]

Fig.2.14 shows the block diagram of Road sign detection and steering control of autonomous vehicles. Raspberry Pi 3 is the main controlling unit in the system. The system consists of a web camera that automatically captures the video data, converts them into a number of frames. The images are processed with the Haar cascade classifier algorithm for sign detection and classification. Based on the detected sign, the vehicle is controlled by a DC motor interfaced with Raspberry Pi. The motor can be driven using an L298 motor driver. The power to the system is given by a 12V battery.

The system consists of two modules: detection stage and classification stage. In the image acquisition step the road scene images were taken in a good camera with 5 megapixels, so that there will be a good distribution of images taken under different lighting conditions and at different angles. The images in each category were randomly selected.

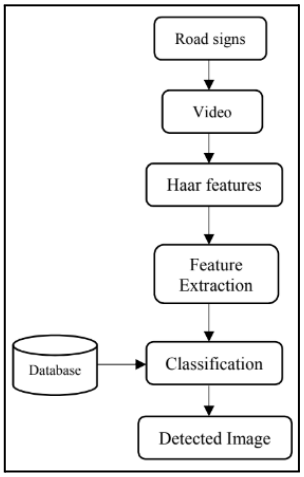

Fig. 2.15 Flowchart for detection of images [7]

From Fig.2.15 it is observed that the captured images are taken to the detection stage. The main work of the detection module is to segment the captured input image and extract the areas in which it contains the road sign pattern and then give it to the classification stage. The proposed algorithm uses the haar features of the road sign image in order to identify the interested region that is the actual road sign. The main aim of the classification stage is to classify the extracted regions to its input to the road sign database. The final reduced and normalized regions of the road sign image is detected.

Around 10 datasets for different road sign images have been taken and it is processed in the proposed work. The algorithms were implemented both in MATLAB and ARM processors. Various parameters have been measured and tabulated for around five images. The peak signal to noise ratio of the image and its minimum error value is calculated.

Images	PSNR	MSE	Execution Time(μs)
	30.92	0.063	0.37
	32.20	0.052	0.35
	31.32	0.051	0.22
	31.91	0.036	0.29
	32.61	0.056	0.31

Table 2.5: Performance metrics in ARM Processor [7]

From Table 2.5, it is inferred that the proposed algorithm implemented in ARM processors gives much better results by producing higher PSNR value with less error. Haar-like features are more robust to lightning or brightness changes than colour histogram.

This is because, instead of validating several features of image pixels at a time, the cascade is made on all the features for several stages using a cascade classifier during the detection stage. Therefore, it is computationally simple and fast with a higher detection rate. Also, it is clear that the haar cascade classifier has shown better performance for the images which contain the complex background. The overall execution time is very much lesser when processed using ARM Processor than in MATLAB.

Furthermore, the authors said that the system is trained to detect only a few image datasets due to less memory capacity of the SD card in the Raspberry pi processor. For training more datasets, the GPU-like processor can be used because they are capable of performing parallel processing and it provides high-end computation with less processing time. It can be trained and store huge datasets efficiently.

2.1.7. Working model of Self-driving car using Convolutional Neural Network, Raspberry Pi and Arduino [8]

This paper proposes a working model of self-driving car which is capable of driving from one location to the other or to stay on different types of tracks such as curved tracks, straight tracks and straight followed by curved tracks. A camera module is mounted over the top of the car along with Raspberry Pi sends the images from real world to the Convolutional Neural Network which then predicts one of the following directions, right, left, forward or stop which is then followed by sending a signal from the Arduino to the controller of the remote-controlled car and as a result of it the car moves in the desired direction without any human intervention.

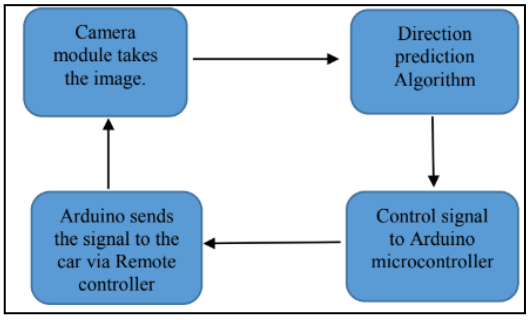

Fig.2.16 Proposed self-driving model [8]

From Fig.2.16, the model takes an image with the help of Pi cam attached with Raspberry Pi on the car. The Raspberry-Pi and the laptop are connected to the same network, the Raspberry Pi sends the image captured which serves as the input image to the Convolutional Neural Network. The image is Gray-scaled before passing it to the Neural Network. Upon prediction the model gives one of the four outputs i.e., left, right, forward or

stop. When the result is predicted, a corresponding Arduino signal is triggered which in turn helps the car to move in a particular direction with the help of its controller**.**

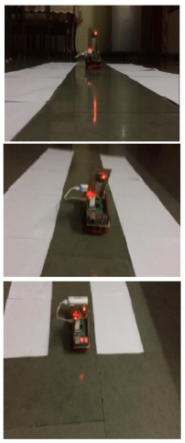

Fig.2.17 Car moving on straight track (top to bottom sequence) [8]

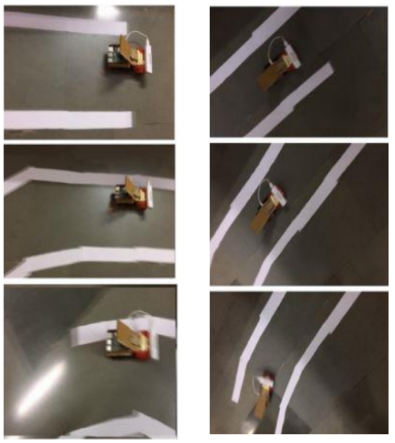

Fig. 2.18 Car moving on straight followed by a curve (top to bottom sequence) [8]

Fig. 2.17 and Fig.2.18 shows the results of the implemented model moving forward and taking a curve left respectively. In this paper, a method to make a model of a self-driving car is presented. With the help of Image Processing and Machine Learning a successful model was developed which worked as per expectation. Thus, the model was successfully designed, implemented and tested. Though there were some drawbacks obtained by the authors of the paper. When the car took a steep turn, it drifted out of the lanes. An advanced system should be developed which would take care of this issue, it would not only make the system reliable but at the same time it would make the overall design attractive and risk-free from accidents.

2.1.8. Self-Driving Robot using Neural Network [9]

This paper introduces the autonomous robot which is a scaled down version of an actual self-driving vehicle and designed with the help of a neural network. The main focus is on building an autonomous robot and training it on a designed track with the help of a neural network so that it can run autonomously without a controller or driver on that specific track. The robot will stream the video to a laptop which will then take decisions and send the data to raspberry pi which will then control the robot using a motor driver. This motor driver will move the robot in required directions. Neural Network is used to train the model by first driving the robot on the specially designed track by labelling the images with the directions to be taken. After the model is trained it can make accurate predictions by processing the images on the computer. This approach is better than the conventional method which is done by extracting specific features from images.

Raspberry pi will work as a brain of the robot, which takes all the decisions of the robot and live streams video to the laptop. Ultrasonic sensor is used to calculate the distance of the obstacles ahead. Motor control IC is used to control the motors.

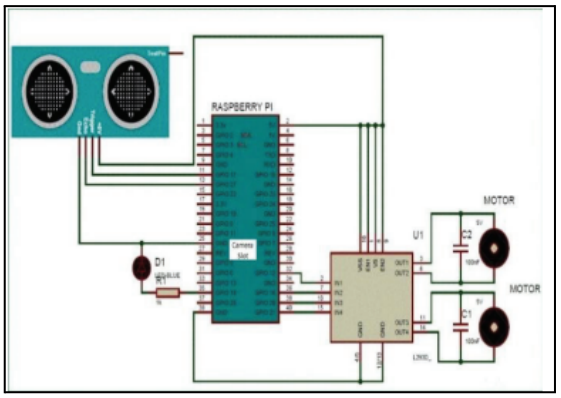

Fig.2.19 Circuit Diagram of Self Driving Robot using Neural Network [9]

Fig 2.19 shows the circuit diagram of a Self-Driving Robot using Neural Network. OpenCV, TensorFlow and NumPy libraries are used in this model.

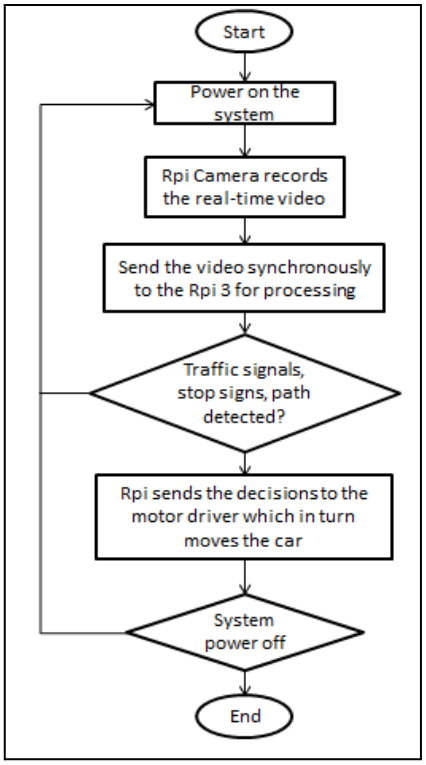

Fig.2.20 Block Diagram of Self Driving Robot [9]

Fig.2.20 shows the block diagram of Self Driving Robot. The Pi Camera mounted on raspberry pi captures the images and transmits to the computer by using SOCKET programming. Raspberry pi is made as a client and computer as a server. Pygame, a library of python which is used to label the image which gets called when you press a specific key. Hence, when the key is pressed pygame is called which saves the corresponding label to the image. The images are saved to a respective folder when a key is pressed. These images stored are used to view the data being trained. The images along with labels stored in NumPy

format which are used to train the data. TensorFlow is used to train the model with these datasets. After the model is trained, it is saved. Then it can make predictions on the new data. The other pre-trained models of traffic signals and stop signals are used.

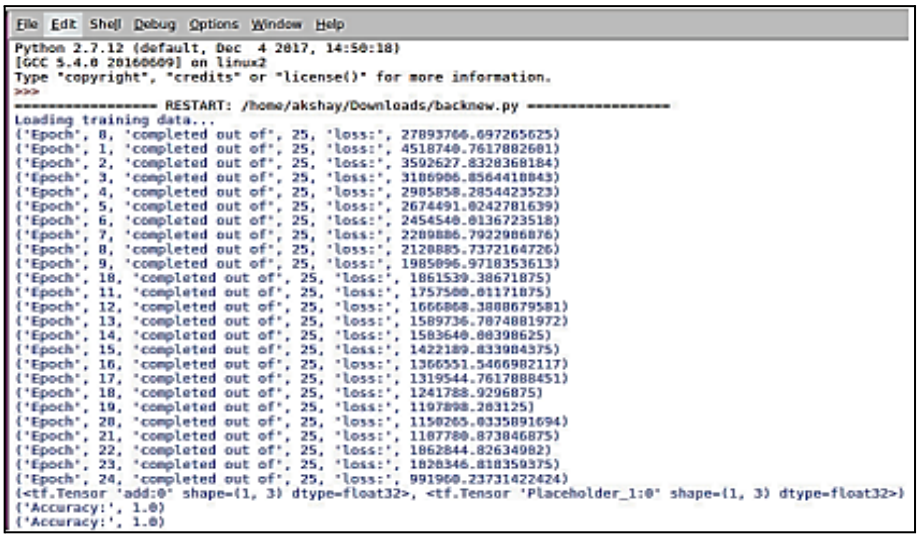

Fig. 2.21 Model Training of Self Driving Car using Neural Network [9]

Fig.2.21 illustrates the model training of the Self Driving Car Model. The live feed will be sent to the laptop from Raspberry pi. First the Raspberry pi will check if all the conditions are fulfilled like minimum distance and obstacle ahead and many more. Socket programming will be used for client and server communication. Multithreading will be used to receive the live feed and send the predictions simultaneously. The prediction made by this model is accurate and the distance is measured to prevent any collision. '0', '1' and '2' are the predictions for forward, right and left direction. When the robot sees a straight path, the model gives a prediction as '0' and the robot moves in the forward direction. When the robot sees a right turn in the path, the model gives a prediction as '1' and the robot turns in the right direction. When the robot sees a left turn in the path, the model gives a prediction as '2' and the robot turns in the leftward direction. In this paper, an autonomous robot is designed and trained with the help of a neural network. It is observed that, once trained, the robot worked autonomously without any intervention of human beings and performance of this robot is very satisfactory.

2.2 Summary of Literature Review

This section gives the summary of the research papers described in detail in section 2.1, which is depicted in the form of a table given below.

Sr. no.	Title of Technical paper	Methodology	Results	Drawbacks
1.	Real-time traffic sign detection and recognition using Raspberry Pi. [2]	A Raspberry Pi 3 module is used to run the TensorFlow algorithm and is connected to the Pi camera. A Display module to display the related information of the detected traffic sign and a speaker to give an alert sound to alert the driver.	The performance has been evaluated in terms of accuracy, delay and reliability. The results show that the average accuracy of detecting and identifying traffic sign images from the five traffic sign classes is above 90%.	The maximum average delay in determining the type of traffic sign in the system is 3.44 seconds when the car moves at 50 km/h.
2.	Real-time Lane detection and Motion Planning in RPi and Arduino for an Autonomous Vehicle Prototype [3]	It is a vehicle prototype that recognizes streets' lanes and plans its motion accordingly without any human input. Rpi and Arduino are used for processing and taking the decisions for the vehicle movement.	The robot is designed and trained with the help of Canny Edge Detection, Hough Transformation algorithm and Kalman's filter.	Changes in particular directions are not very smooth which makes the system unstable. Processing rate can be improved
3.	Artificial Intelligence based Self-Driving Car [4]	Implementation of self-driving car by self-navigation using Haar Cascade classifier and Monocular Vision Algorithm.	The paper solved many of those into two sensors by detecting distance and objects like stop signs, signals and other obstacles with a single method of monocular vision which can be enhanced in future from a scaled car to an actual car.	Due to the size of the model the maximum speed limit is 10-15 km.
4.	Convolution Neural Network based Working Model of Self-Driving Car - A study [5]	CNN is used for efficient image classification for all detection purposes. A pi camera is used for capturing live images which is sent to Raspberry pi 3 for image processing and Arduino is used to move the car around by sending signals.	In this paper, Deep learning algorithms like CNN are employed to make immediate decisions for the self-driving car. The algorithms like Faster R-CNN models, YOLO and canny edge detection algorithms have been studied for detecting objects, lanes, Traffic signals and signboards.	For CNN, a large training data is needed and in case of multiple layers is comparatively a slow processor.

5.	Object detection in autonomous driving - from large to small datasets [6]	Precision and recall were done on big data (BDD100K) and small data using Yolo, SSD, Retina Net and Faster R-CNN to see which is the best object detector and what improvements could be made.	In this paper, after analysis it was found that Yolo has the best recall and SSD the best precision, close to Faster R-CNN. The recall on small data is much better than on big data.	SSD is only useful for detecting bigger objects. Recall on big data gives the worst result.
6.	Road Sign Recognition System for Autonomous Vehicle using Raspberry Pi [7]	Implementation of road sign detection and control of an autonomous vehicle using Haar Cascade Classifier algorithm.	Convey information about the vehicle which needs to take left/right turn and start/stop according to road signs.	The system was trained to detect only a few image datasets due to less memory capacity of the SD card in the pi processor.
7.	Working model of Self-driving car using Convolutional Neural Network, Raspberry Pi and Arduino [8]	Paper presents a new model which involves Machine Learning as well as Image Processing. Image Processing helps in preparing the input image.	The car was trained under different combinations of the track i.e., straight, curved, combination of straight and curved etc. Total of 24 videos were recorded out of which 10868 images were extracted.	Since the car slightly moves out of the track which can be a serious issue if it hits nearby objects if we consider a real car.
8.	Self-Driving Robot using Neural Network [9]	A Raspberry Pi 3 module is used to run the TensorFlow machine learning algorithm and is connected to a Rpi camera to record the real-time video, and a neural network is used to train the model.	The model could detect the lane, directions, stop sign, signals and other obstacles which can be enhanced in future from a scaled car to an actual car.	Canny Edge Detection algorithm might have given better results for detection of path to travel

Table 2.6: Literature Review Summary

From Table 2.6, we can observe that all the papers have used Raspberry Pi as the main processor for implementation of the system. This is due to the fact that, to process images / videos we often need a processor and a graphical interface (GUI) to see the output results which can be achieved by using Raspberry Pi as it performs as a central processing unit (CPU). Many types of classifiers were used and compared by the authors. It was suggested that for small data sets algorithms / classifiers like HAAR Cascade, R-CNN should be used to improve the speed of computation whereas for huge dataset algorithms like Yolo, SSD, Retina Net and Faster R-CNN can be used for better performance at a cost of computing complexity.

Chapter 3
Proposed Project Description

3.1 Problem Statement

The aim of this project is to Design and Implement a Self-Driving Robot Car which will detect the pre-defined lanes. It will automatically turn itself without manual intervention according to the turns of the path.

3.1.1 Steps involved

Before starting the physical implementation of this project, a literature survey was carried out. A thorough research was done upon this project. Relevant papers and information on the websites (internet) were analysed and noted down carefully.

The first and foremost step was to build the robot car. A thorough testing of components was done. After testing all the components, the components were assembled on the robot car chassis. The Raspicam Camera was mounted on the robot car chassis with some height (approximately 20 cm) from the ground for better view. After the Robot car was assembled, Installation of Raspbian OS into Raspberry Pi along with all the required softwares and libraries for image processing was done. Testing of Raspicam Camera was performed with the help of the command "raspistill -o name_of_image.jpg". The code for video processing was written in C++ language for faster processing speed. Finally, the functioning of the robot car was tested and troubleshooted for any problems faced.

3.2 Block Diagram

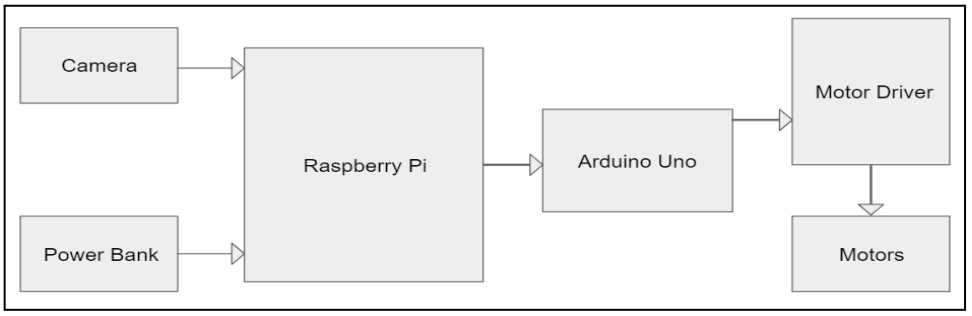

Fig 3.1 Block Diagram of Self Driving Robot Car

As seen from Fig.3.1, to build this entire system, we will be using two microcontrollers. first being the Raspberry Pi 3B+ and other being the Arduino Uno R3. The Raspberry Pi board will act as a master device whereas the Arduino Uno will be used as a slave device. The power to all the components in this project is given through a power bank. A Raspicam V1.3 is attached to the Raspberry Pi for taking the input video and images. To control the motors, a dual-bridge motor driver L298N has been used. There are four wheels attached to the entire chassis which are controlled by Arduino Uno. The Arduino Uno listens to the commands given by Raspberry Pi after processing the videos and images.

3.3 Component Description

3.3.1 Hardware

1) Raspberry Pi Model 3B+ [12]

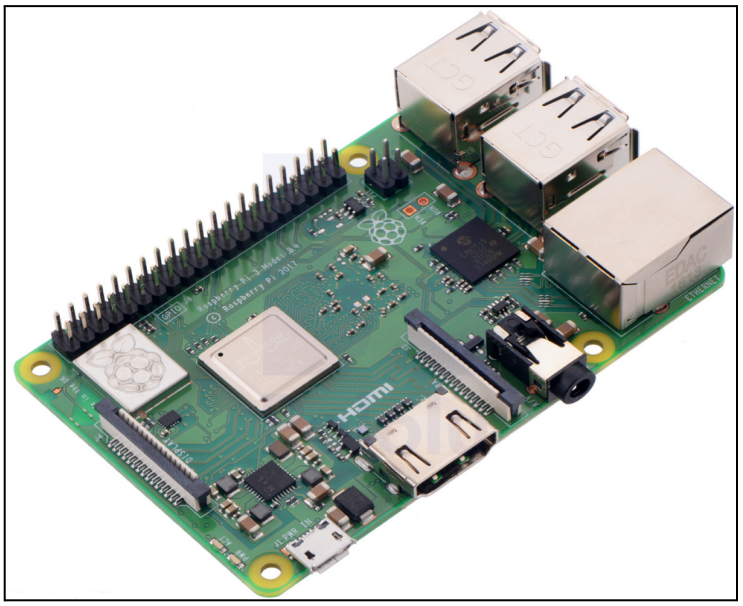

Fig.3.2 Raspberry Pi Model 3B+

The Raspberry Pi is a credit card-sized computer. The Raspberry Pi 3 Model B+ is an improved version of the Raspberry Pi 3 Model B. It is based on the BCM2837B0 system-on-chip (SoC), which includes a 1.4 GHz quad-core ARMv8 64bit processor and a powerful VideoCore IV GPU. The Raspberry Pi supports a wide range of ARM GNU/Linux distributions, including Snappy Ubuntu Core, Raspbian, Fedora, and Arch Linux, as well as Microsoft Windows 10 IoT Core. The Raspberry Pi 3 Model B+ outperforms the Model B in many ways, including a faster CPU clock speed (1.4 GHz vs 1.2 GHz), increased Ethernet throughput, and dual-band WiFi. The dual-band wireless LAN comes with modular compliance certification, allowing the board to be designed into end products. The Raspberry Pi was designed by the Raspberry pi Foundation to provide an affordable platform for experimentation and education in computer programming.

Features:

1) 1.4 GHz quad-core BCM2837B0 ARMv8 64bit CPU
2) 1 GB RAM
3) VideoCore IV 3D graphics core
4) Ethernet port
5) dual-band (2.4 GHz and 5 GHz) IEEE 802.11.b/g/n/ac wireless LAN (WiFi)
6) Bluetooth 4.2
7) Bluetooth Low Energy (BLE)
8) Four USB ports

9) Full-size HDMI output
10) Four-pole 3.5 mm jack with audio output and composite video output
11) Camera interface (CSI)
12) Display interface (DSI)
13) Micro SD card slot

2) Arduino Uno [13]

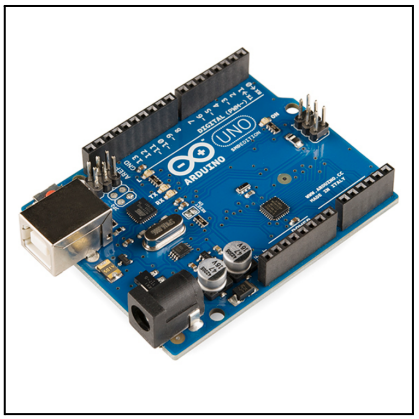

Fig.3.3 Arduino Uno SMD R3

The Arduino Uno is an open-source microcontroller board based on the Microchip ATmega328P microcontroller developed by Arduino.cc and initially released in 2010. The board is equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards (shields) and other circuits. The board has 14 digital I/O pins (six capable of PWM output), 6 analog I/O pins, and is programmable with the Arduino IDE (Integrated Development Environment), via a type B USB cable. It can be powered by the USB cable or by an external 9-volt battery, though it accepts voltages between 7 and 20 volts. It is similar to the Arduino Nano and Leonardo. The Uno board is the first in a series of USB-based Arduino boards; it and version 1.0 of the Arduino IDE were the reference versions of Arduino, which have now evolved to newer releases. The ATmega328 on the board comes pre-programmed with a bootloader that allows uploading new code to it without the use of an external hardware programmer.

Technical Specifications:

1. Microcontroller: Microchip ATmega328
2. Operating Voltage: 5 Volts
3. Input Voltage: 7 to 20 Volts
4. Digital I/O Pins: 14
5. PWM Pins: 6 (Pin # 3, 5, 6, 9, 10 and 11)
6. UART: 1

7. Analog Input Pins: 6
8. DC Current per I/O Pin: 20 mA
9. DC Current for 3.3V Pin: 50 mA
10. Clock Speed: 16 MHz
11. Power Sources: DC Power Jack, USB Port and the VIN pin (+5 volt only)

3) Dual shaft BO motor

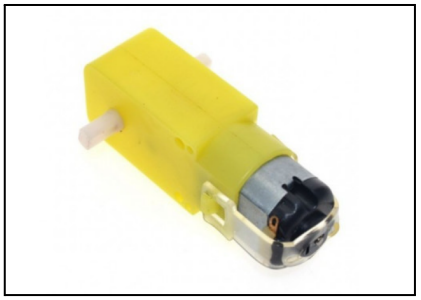

Fig 3.4 200 RPM Dual shaft BO motor

DC Motor – 200RPM – 12Volts geared motors are generally a simple DC motor with a gearbox attached to it. This can be used in all-terrain robots and a variety of robotic applications. These motors have a 3 mm threaded drill hole in the middle of the shaft thus making it simple to connect it to the wheels or any other mechanical assembly. 200 RPM 12V DC geared motors widely used for robotics applications. Very easy to use and available in standard size.mThe most popular L298N H-bridge module with onboard voltage regulator motor driver can be used with this motor that has a voltage of between 5 and 35V DC or you can choose the most precise motor driver module from the wide range available in our Motor divers category as per your specific requirements.

Specifications:

1. RPM: 200.
2. Operating Voltage: 12V DC

4) L293D Motor Driver Module

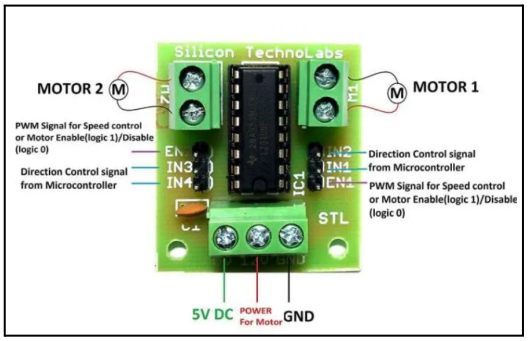

Fig 3.5 L293D Motor Driver Module

L293D Motor Driver Module is a medium-power motor driver perfect for driving DC Motors and Stepper Motors. It uses the popular L293 motor driver IC. It can drive 4 DC motors on and off, or drive 2 DC motors with directional and speed control. The driver greatly simplifies and increases the ease with which you may control motors, relays, etc from micro-controllers. It can drive motors up to 12V with a total DC current of up to 600mA. You can connect the two channels in parallel to double the maximum current or in series to double the maximum input voltage.

Features:

1. Wide supply voltage: 4.5 V to 12 V.
2. Max supply current: 600 mA per motor.

5) Raspberry Pi Camera Module

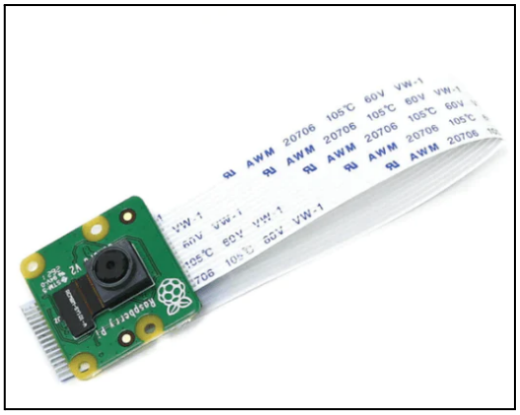

Fig.3.6 Raspberry Pi Camera Module

Pi Camera module is a camera that can be used to take pictures and high-definition video. Raspberry Pi Board has CSI (Camera Serial Interface) interface to which we can attach the Pi Camera module directly. This Pi Camera module can attach to the Raspberry Pi's CSI port using a 15-pin ribbon cable. Features of Pi Camera.

Here, we have used the Pi camera v1.3. Its features are listed below,

1. Resolution – 5 MP
2. HD Video recording – 1080p @30fps, 720p @60fps, 960p @45fps and so on.
3. It Can capture wide, still (motionless) images of resolution 2592x1944 pixels
4. CSI Interface enabled.

3.3.2 Software

1) Geany C++

Geany is a solid editor to use on Raspberry Pi as it's preinstalled with Raspberry Pi OS and perfect to code in Python or C/C++. There is a built-in terminal to compile and run scripts directly in it, and many other settings to save time while coding (like shortcuts and

productivity options). It is very well suited for coding on the Raspberry Pi because it provides a lot of functionality while efficiently using computer resources like CPU and most importantly RAM (system memory). Once Geany is running, you can create a new file by going to File > New. Saving the file as either ".c" or ".py" (or any other common language file extension) will immediately inform Geany what kind of language you're working with, so it can start highlighting your text. Geany can be used with most languages, including the Python and C examples we've just examined. Some tweaks to the default IDE options are necessary, though. [10]

Some of Geany's notable features include:

- Lightweight and Fast
- Split Screen Editing
- Extensibility through Plugins

2) Arduino IDE

The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino hardware to upload programs and communicate with them. The Arduino IDE is an open-source software, which is used to write and upload code to the Arduino boards. The IDE application is suitable for different operating systems such as Windows, Mac OS X, and Linux. It supports the programming languages C and C++. Here, IDE stands for Integrated Development Environment. The program or code written in the Arduino IDE is often called sketching. We need to connect the Genuino and Arduino board with the IDE to upload the sketch written in the Arduino IDE software. The sketch is saved with the extension '.ino.' The Arduino IDE is very simple and this simplicity is probably one of the main reasons Arduino became so popular. [11]

3.4 Working

The proposed project has two parts, lane detection, which is performed solely by video processing, and second is the detection of various variables like stop signs, traffic lights, and obstacles using machine learning algorithms.

3.4.1 Lane Detection

There are various steps to be carried out for detection of Lane Lines. They are as follows:

1) Connect the Raspberry Pi to your Desktop / Laptop with help of an Ethernet Cable or Wi-Fi. Install all the required libraries used for Video or Image processing in OpenCV. Update the path location for all of the installed libraries in the programming editor, here, Geany Programming Editor for C++ is used.

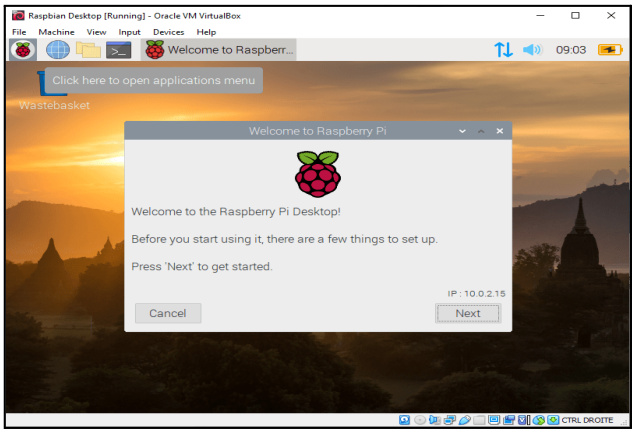

Fig 3.7 Raspberry Pi - Raspbian Desktop

2) Start capturing the video using camera.grab() function available in OpenCV. Adjust the Brightness, Contrast, Saturation, Hue and other parameters to achieve proper video.
3) After capturing the video, Image / Video Processing steps need to be implemented. They are as follows:
   a) Change the colour space of the images from RGB to grayscale.
   b) Define a Region of Interest (ROI).
   c) Apply Bird Eye View Transformation also known as Perspective Wrapping.

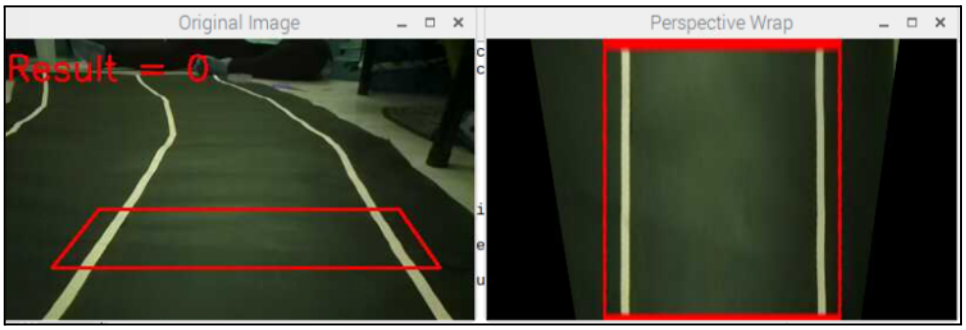

Fig 3.8 Defining the ROI and Perspective Wrapping

   d) Apply Thresholding to the Wrapped Image.

e) Apply Canny Edge Detection to the Perspective Wrapped Image for smoothening the lane line. Add the thresholded image and edge detected image for perfectly visible lane lines.

f) Detect the Lane Centre and the Frame Centre and take the difference between them as a parameter to detect the position (direction) of the car on the lane and move accordingly. To detect the lane centre first we need to detect both the lane lines. To detect the lane lines, we create a dynamic array, wherein the sum of the pixel intensities of each column of the frame is stored. For example, the first column of the frame will result in '0' as a sum, because all the pixels are black for the entire column. Refer to Perspective Wrapped Image in Fig. 3.8. After all the sum of pixels intensities for each column are stored in the array, we can scan the first half of the array to get the position of the first lanes where the sum of pixel intensities will be greater than zero for a particular column in the entire frame and similarly, we can scan the second half of the obtained array for the second lane

line. The Frame centre on the other hand will simply be the line drawn from the centre of the width of the Frame (Video Frame).

g) Use this difference as the information from the Rpi to Arduino Uno Digital pins to turn the wheels of the car for proper movement.

4) Test and troubleshoot the model.

3.4.2 Stop Sign Detection

The steps for the detection of the stop sign are as follows:

1) Creating / Making the stop sign for taking positive samples.

Fig 3.9 Stop Sign

2) Taking the Positive and the Negative samples for training purposes.

- Around 100 - 150 positive samples of stop signs must be taken from different angles under varying light conditions. Negative Samples are the samples which do not contain the stop sign. They are the images consisting of surroundings excluding the stop sign.

Fig 3.10 Positive Sample

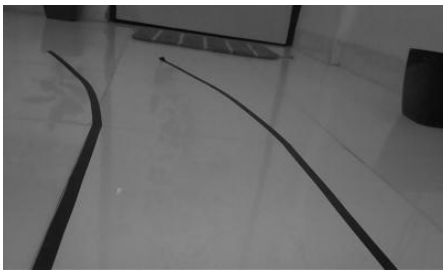

Fig 3.11 Negative Sample

3) Training of the Stop sign model using HAAR cascade model.
4) Loading the trained file (.xml file) in the code writing and testing the detection.
5) Creating and solving the linear equations to calculate the distance from the objects.
   - Using the below formula, the distance between the stop sign and the robot car is found with the help of solving the equation **y = mx + c** where y is the distance between the stop sign and car; x is the difference between the end pixel values. Two readings of x are taken by varying y. For example, first place the car at a distance of 15 cm (y =15) with respect to the stop sign and note down the value of x. Again note the value of x when the distance between the stop sign and robot car is 30 cm (y = 30). Arrange both the values in the form of y = mx + c and obtain the value of m and c. Now substitute the value of m and c in the equation y = mx + c to get correct values of x and y. The value of y can be used to put a threshold for the car to stop at a certain distance.
6) Finally, having a parameter/command from Rpi to Arduino Uno to control the wheels of the robot car to go to the desired directions or take desired actions.

3.4.3 Traffic Lights Detection

The steps for the detection of the stop sign are as follows:

1) Creating / Making the stop sign for taking positive samples.

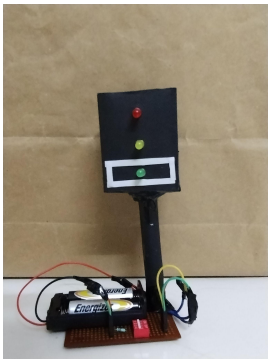

Fig. 3.12 Traffic Light

2) Taking the Positive and the Negative samples for training purposes.
   - Around 100 - 150 positive samples of traffic lights must be taken from different

angles under varying light conditions. Positive samples are the samples which have the Red light and the Yellow lights. Negative Samples are the samples which contain the green light and the environment. They are the images consisting of surroundings excluding the Red and Yellow lights.

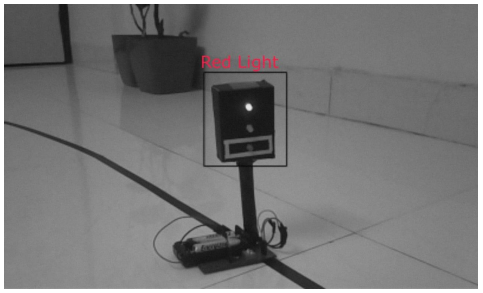

Fig 3.13 Positive sample

Fig. 3.14 Negative Sample

3) Training of the Traffic Lights model using HAAR cascade model.
4) Loading the trained file (.xml file) in the code writing and testing the detection.
5) Creating and solving the linear equations to calculate the distance from the traffic lights pole..
   - Using the below formula, the distance between the traffic light and the robot car is found with the help of solving the equation **y = mx + c** where y is the distance between the traffic light and car; x is the difference between the end pixel values. Two readings of x are taken by varying y. For example, first place the car at a distance of 15 cm (y =15) with respect to the traffic light and note down the value of x. Again note the value of x when the distance between traffic light and robot car is 30 cm (y = 30). Arrange both the values in the form of y = mx + c and obtain the value of m and c. Now substitute the value of m and c in the equation y = mx + c to get correct values of x and y. The value of y can be used to put a threshold for the car to stop at a certain distance.
   - Incase of the Red light, the car should stop at a particular distance from the traffic light. Incase of the Yellow light, it should continue its motion. Incase of the green light, the car ignores it and continues to move.
6) Finally, having a parameter/command from Rpi to Arduino Uno to control the wheels of the robot car to go to the desired directions or take desired actions.

3.4.4 Object Detection

The steps for the detection of an object are as follows:

1) Creating / Making the object for taking positive samples.

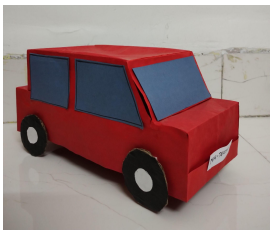

Fig 3.15 Object (dummy car)

7) Taking the Positive and the Negative samples for training purposes.
   - Around 100 - 150 positive samples of an object must be taken from different angles under varying light conditions. Negative Samples are the samples which do not contain the object. They are the images consisting of surroundings excluding the object.

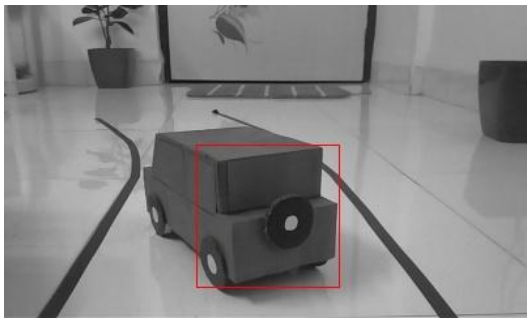

Fig 3.16 Positive Sample

Fig 3.17 Negative Sample

8) Training of the object detection model using HAAR cascade model.
9) Loading the trained file (.xml file) in the code writing and testing the detection.
10) Creating and solving the linear equations to calculate the distance from the object and the car.
    - Using the below formula, the distance between the object(we have used a dummy car) and the robot car is found with the help of solving the equation **y = mx + c** where y is the distance between the object and car; x is the difference between the end pixel values. Two readings of x are taken by varying y. For example, first place the car at a distance of 20 cm (y =20) with respect to the object(dummy car) and note down the value of x. Again note the value of x when the distance between the object and robot car is 40 cm (y = 40). Arrange both the values in the form of y = mx + c and obtain the value of m and c. Now substitute the value of m and c in the equation y = mx + c to get correct values of x and y. The value of y can be used to put a threshold for the car to stop at a certain distance.
11) Finally, having a parameter/command from Rpi to Arduino Uno to control the wheels of the robot car to go to the desired directions or take desired actions.

Chapter 4

Results

4.1 Lane Detection Using Raspberry Pi and Arduino Uno

4.2 Simulation Results of Stop Sign Detection

4.3 Simulation Results of Traffic Light Detection

4.4 Simulation Results of Obstacle Detection

4.5 Hardware Implementation

4.1 Lane Detection Using Raspberry Pi and Arduino Uno

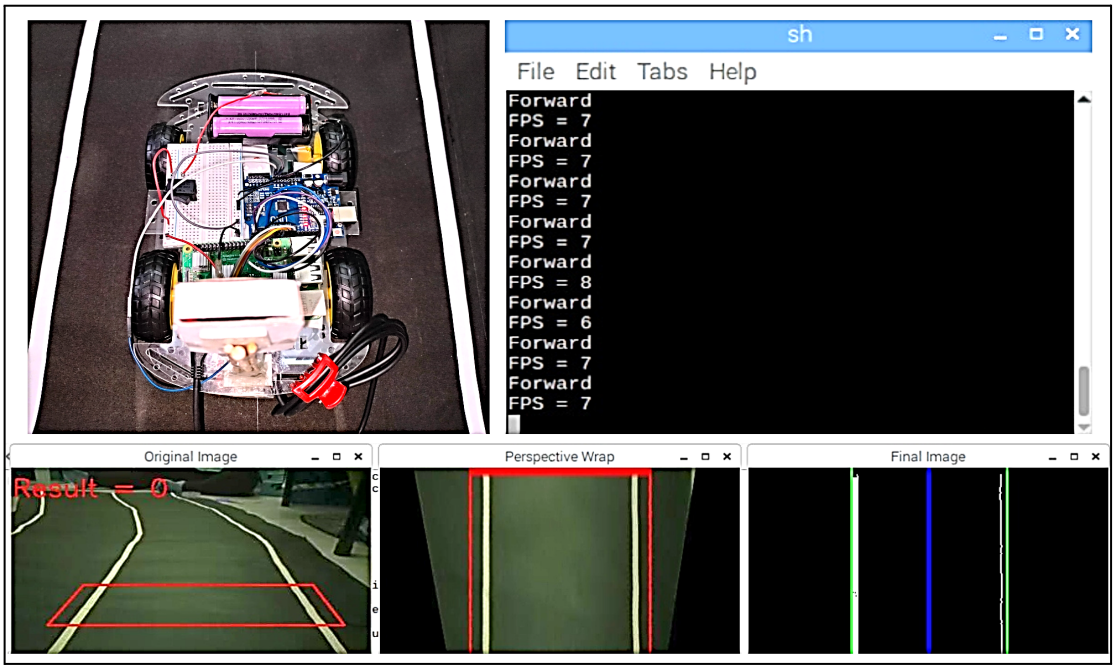

Fig.4.1 Simulation Results of Robot Car when in the centre of lane

Fig.4.1 is the screenshot taken from the Raspberry Pi's Desktop. It depicts that when the result is 0 (Result = 0, written in red colour), the lane centre (green line) and the frame centre (blue centre) overlaps each other indicating the car is placed at the centre of the lane. In this picture, we cannot see the centre green line as it is overlapped by the blue line. When these lines overlap each other, it indicates that the car is exactly at the centre and it needs to move forward. This can be seen in the shell window (sh window).

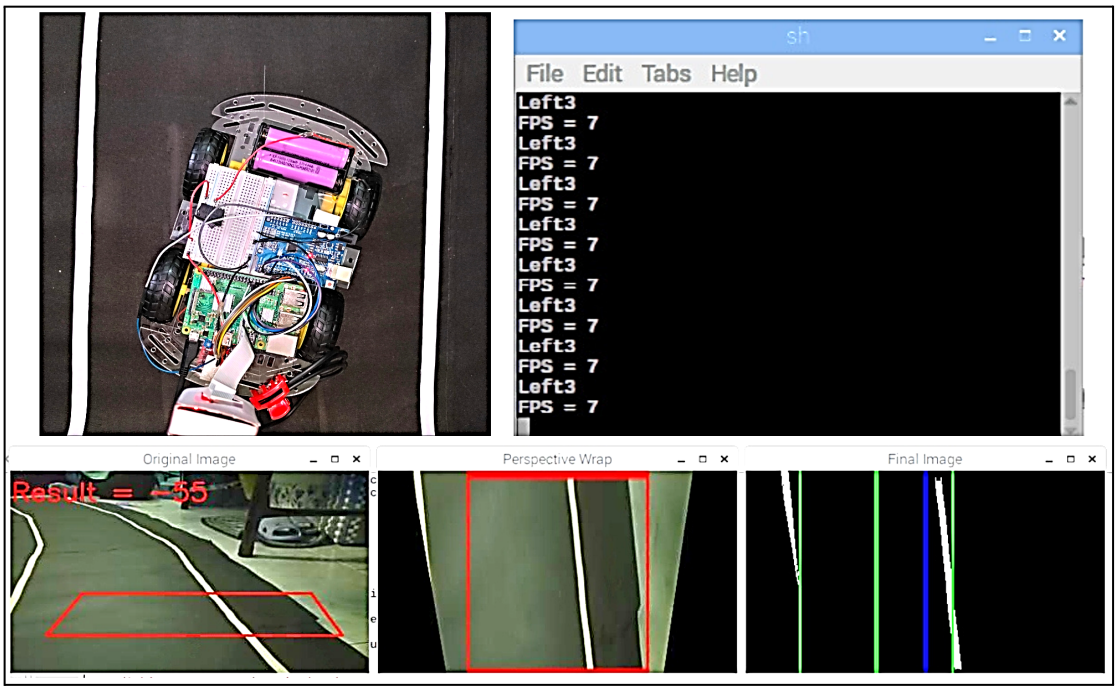

Fig. 4.2 Simulation Results of Robot Car when turned towards right side of lane

Fig.4.2 is also a screenshot taken from the Raspberry Pi's Desktop. The car is now facing at the right side, which means it needs to move to the left to get back on the centre of the lane. Here, the result is negative (Result = -55, written in red colour), the lane centre (green line) and the frame centre (blue centre) does not overlap each other indicating the car is not placed at the centre of the lane anymore. The centre green line is before the blue line indicating the difference is negative and it needs to move towards the left. This can be seen in the shell window (sh window). Left 3 is written indicating a left turn to be taken by the car.

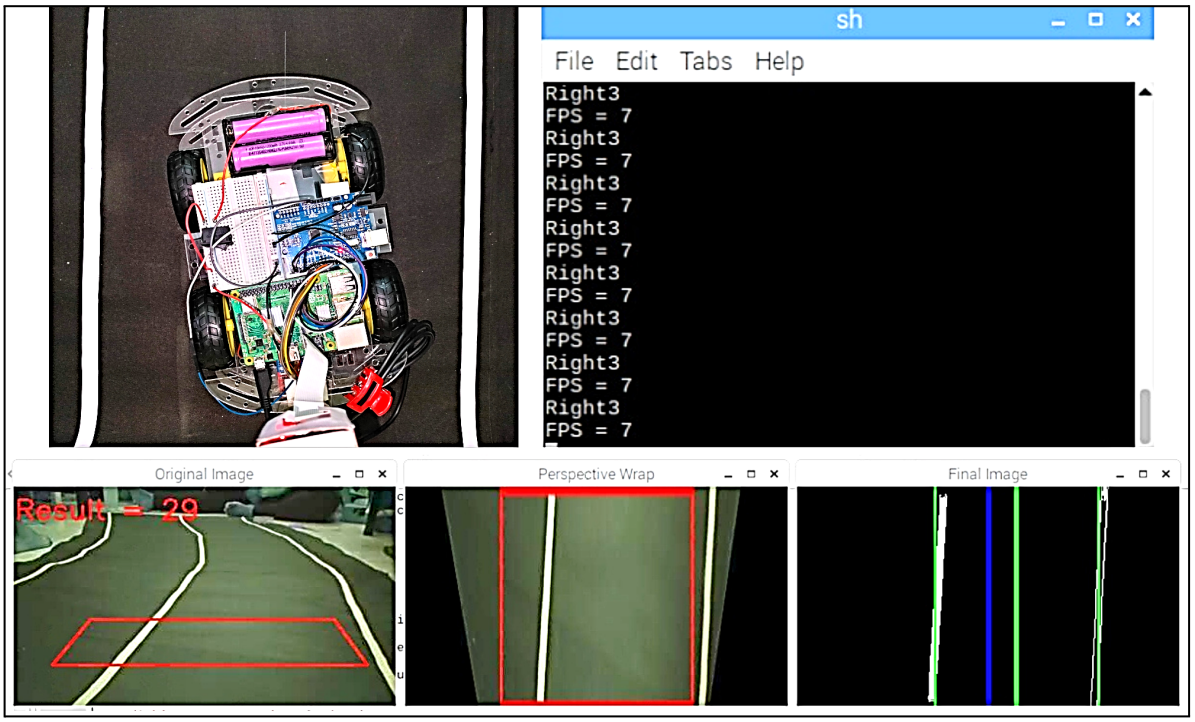

Fig. 4.3 Simulation Results of Robot Car when turned towards left side of lane

Fig.4.3 the car is facing at the left side, which means it needs to move to the right to get back on the centre of the lane. Here, the result is positive (Result = 29, written in red colour), the lane centre (green line) and the frame centre (blue centre) do not overlap each other indicating the car is not placed at the centre of the lane. The centre green line is after the blue line indicating the difference is positive and it needs to move towards the right. This can be seen in the shell window (sh window). Right 3 is written indicating a right turn to be taken by the car.

4.2 Simulation Results of Stop Sign Detection

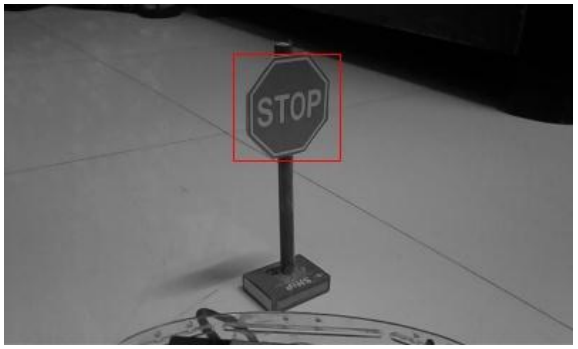

Fig. 4.4 Detected Stop Sign by HAAR Model

Fig. 4.5 Stop Sign Detected by Raspberry Pi

Figure 4.4 is one of the results given by the GUI trainer after testing the stop sign for 100 test images of the stop sign. Fig. 4.5 is the stop sign detected by the raspberry pi. It is also showing the distance between the stop sign and the robot car as D = 54 cm.

4.3 Simulation Results of Traffic Light Detection

Fig. 4.6 Detected traffic light by HAAR Model

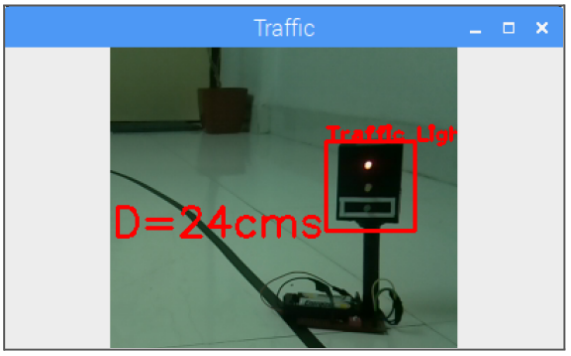

Fig. 4.7 Traffic light Detected by Raspberry Pi

Figure 4.6 is one of the results given by the GUI trainer after testing the traffic lights for 100 test images of the traffic lights. Fig. 4.7 is the traffic lights detected by the raspberry pi. It is also showing the distance between the traffic lights and the robot car as D = 24 cm.

4.4 Simulation Results of Object Detection

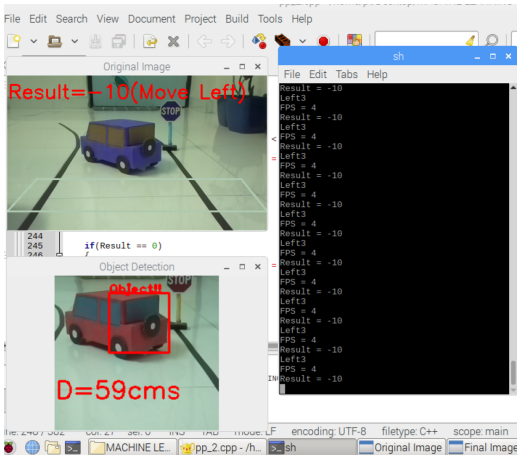

Fig. 4.8 Detected Object by HAAR Model

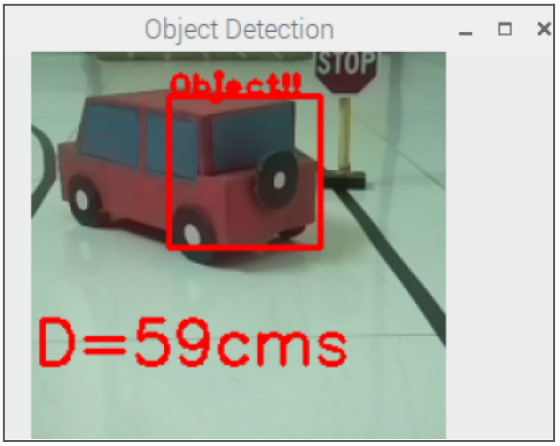

Fig. 4.9 Object Detected by Raspberry Pi

Figure 4.8 is one of the results given by the GUI trainer after testing the object for 100 test images of the object, i.e., the dummy car. Fig. 4.9 is the object detected by the raspberry pi. It is also showing the distance between object (the dummy car) and the robot car as D = 59 cm.

4.5 Hardware Implementation

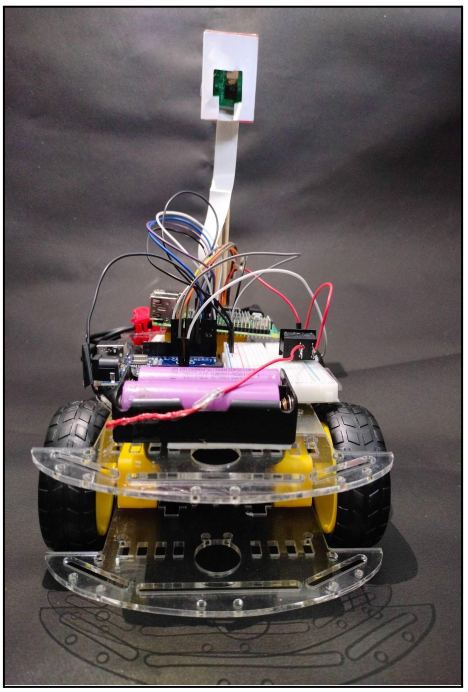

Fig 4.10 Front View of Self Driving Robot Car

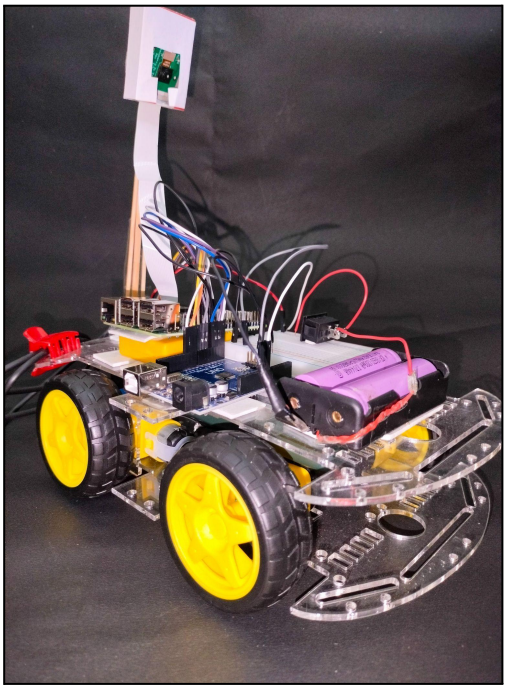

Fig 4.11 Side View of Self Driving Robot Car

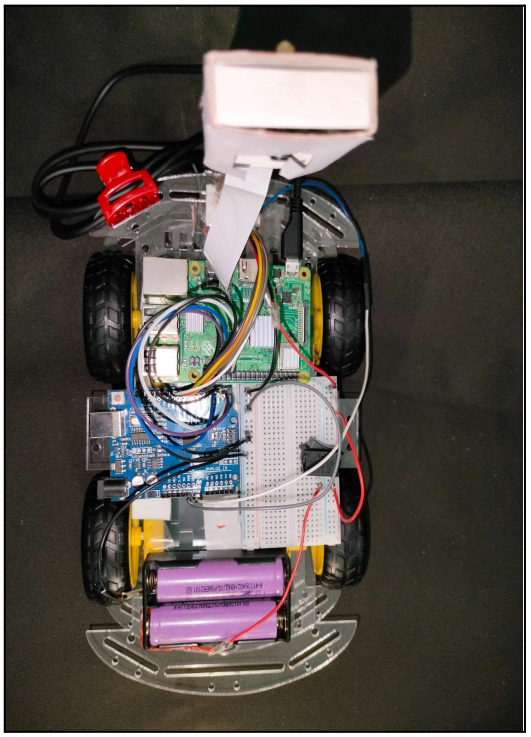

Fig 4.12 Top View of Self Driving Robot Car

Fig. 4.10, 4.11 and 4.12 are the pictures of hardware implementation of the robot car from different views i.e., Front view, Side view and Top view respectively.

Chapter 5

Conclusion and Future Scope

5.1 Conclusion

5.2 Future Scope

5.1 Conclusion

This paper outlines a methodology for detecting lane lines in video processing without utilizing machine learning techniques. By employing adaptive thresholding with Canny edge detection and a constrained search region, lane detection can be achieved even in varying lighting conditions. This approach is more efficient and faster than machine learning-based lane detection carried out by several other authors, as it requires fewer CPU resources. Therefore, it is suitable for situations where the master device's specifications are not optimal, such as in a robot car.

This robot car can detect stop sign, traffic light and obstacles (car). The HAAR cascade algorithm is utilized as a training model. The HAAR Cascade model is used as it has lower computational complexity and higher accuracy than the LBP classifier and faster than HOG classifier. However, the robot car cannot make sharp turns because lane identification fails when one of the lanes ventures outside the Region of Interest. Moreover, the performance of the robot car depends on its speed since it takes time to capture and process each image frame. These drawbacks can be overcome by using a camera with a wider angle and a faster processing unit (CPU).

5.2 Future Scope

In the future it is possible to use this robot in a real time environment in the form of an autonomous vehicle with some modifications. In future, various algorithms like YOLO, TensorFlow, Hough Transform and CNN can be implemented for smoother control on street corners. Using one Raspberry Pi for pre-processing and feeding the pre-processed data to another Raspberry Pi will improve the processing rate. For training more datasets, the GPU-like processor can be used because they are capable of performing parallel processing and it provides high-end computation with less processing time. It can be trained and store huge datasets efficiently. If we could develop an advanced system which would take care of this issue, it would not only make the system reliable but at the same time it would make the overall design attractive and risk-free from accidents.

References

[1] https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries, Last Accessed On: 17th August 2022

[2] Ida Syafiza Binti Md Isa, Choy Ja Yeong, Nur Latif Azyze bin Mohd Shaari Azyze, Real- time traffic sign detection and recognition using Raspberry Pi", 2022 International Journal of Electrical and Computer Engineering (IJECE) Vol. 12, ISSN: 2088-8708, DOI: 10.11591/ijece.v12i1.pp331-338.

[3] Rossi, A., Ahmed, N., Salehin, S., Choudhury, T.H., & Sarowar, G. (2020). "Real-time Lane detection and Motion Planning in Raspberry Pi and Arduino for an Autonomous Vehicle Prototype". ArXiv, abs/2009.09391.

[4] H. Thadeshwar, V. Shah, M. Jain, R. Chaudhari and V. Badgujar, "Artificial Intelligence based Self-Driving Car," 2020 4th International Conference on Computer, Communication and Signal Processing (ICCCSP), 2020, pp. 1-5, doi: 10.1109/ICCCSP49186.2020.9315223.

[5] Chaitra P G, Deepthi V, Gautami S, Suraj H M, Prof. Naveen Kumar, "Convolution Neural Network based Working Model of Self-Driving Car - A study," 2020 International Conference on Electronics and Sustainable Communication Systems (ICESC), 2020, pp. 645 - 650, DOI: 10.1109/ICESC48915.2020.9155826.

[6] David-Traian Iancu, Alexandru Sorici, Adina Magda Florea, "Object detection in autonomous driving - from large to small datasets," 2019 11th International Conference on Electronics, Computers and Artificial Intelligence (ECAI), 2019, pp. 1 - 6, DOI: 10.1109/ECAI46879.2019.9041976.

[7] K. Vinothini and S. Jayant, "Road Sign Recognition System for Autonomous Vehicle using Raspberry Pi", 2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS), 2019, pp.78-83, doi: 10.1109/ICACCS.2019.8728463.

[8] A. K. Jain, "Working model of Self-driving car using Convolutional Neural Network, Raspberry Pi and Arduino," 2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA), 2018, pp. 1630-1635, doi: 10.1109/ICECA.2018.8474620.

[9] A. Mogaveera, R. Giri, M. Mahadik and A. Patil, "Self Driving Robot using Neural Network," 2018 International Conference on Information , Communication, Engineering and Technology (ICICET), 2018, pp. 1-6, doi: 10.1109/ICICET.2018.8533870.

[10] https://geany.org/manual/dev/, Last Accessed On: 18th August 2022

[11] https://docs.arduino.cc/software/ide-v1/tutorials/arduino-ide-v1-basics, Last Accessed On: 18th August 2022

[12] https://static.raspberrypi.org/files/product-briefs/Raspberry-Pi-Model-Bplus-Product-Brief.pdf, Last Accessed On: 20th September 2022

[13] https://robu.in/what-is-arduino-uno/, Last Accessed On: 20th September 2022