PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)
100-ft Ring Road, Bengaluru – 560 085, Karnataka, India

Report on

Advanced Driver Assistance system and Tracking Device using Machine Learning

Submitted by

Aditya V Kakarambi - PES1UG19EC017

Sadir Irfan - PES1UG19EC250

Sanjan IS - PES1UG19EC265

Sathvik Prakash - PES1UG19EC269

January-May 2022

Under the guidance of

Prof. Karpagavalli S

Assistant Professor

Department of Electronics and Communication Engineering

PES University

Bengaluru - 560085

CERTIFICATE

This is to certify that the Project entitled

Advanced Driver Assistance system and Tracking Device using Machine Learning

is a Bonafede work carried out by

Aditya V Kakarambi - PES1UG19EC017

Sadir Irfan - PES1UG19EC250

Sanjan IS - PES1UG19EC265

Sathvik Prakash - PES1UG19EC269

In partial fulfillment for the completion of the Program of Study B. Tech in Electronics and Communication Engineering under rules and regulations of PES University, Bengaluru during the period August–December 2022. It is certified that all corrections/suggestions indicated for internal assessment have been incorporated in the report. The project report has been approved as it satisfies the 6th-semester academic requirements in respect of project work.

Signature with date & seal	*Signature with date & seal*	*Signature with date & seal*
Prof. Karpagavalli S	*Dr. Anuradha M*	*Dr. Keshavan B K*
Internal Guide	*Chairperson*	*Dean of Faculty*

Name and Signature of Examiners:

NAME: **SIGNATURE:**

DECLARATION

We, **Aditya V Kakarambi, Sadir Irfan, Sanjan IS, Sathvik Prakash,** hereby declare that the project entitled, **"Advanced Driver Assistance System and Tracking Device",** is an original work done by us under the guidance of **Prof. Karpagavalli S**, **Assistant Professor, Department of Electronic and Communication Engineering, PES UNIVERSITY**, and is being submitted in partial fulfilment of the requirements for completion of 6th semester course work in the Program of Study B. Tech in **Electronics and Communication Engineering.**

PLACE: BENGALURU

DATE:14/12/2022

NAME AND SIGNATURE OF THE CANDIDATE

Aditya V Kakarambi - PES1UG19EC017

Sadir Irfan - PES1UG19EC250

Sanjan IS - PES1UG19EC265

Sathvik Prakash - PES1UG19EC269

ACKNOWLEDGEMENT

We would like to take this opportunity to thank **Prof. KARPAGAVALLI S**, Professor, Department of Electronics and Communication Engineering, PES University, for his persistent guidance, suggestions, assistance, and encouragement throughout the development of this project. It was an honor for me to work on the project under his supervision and understand the importance of the project at various stages.

We are grateful to the project coordinator **Prof. R M**, Department of Electronics and Communication Engineering for, organizing, managing, and helping with the entire process.

We would like to thank **Dr. ANURADHA M**, Professor and Chairperson, Department of Electronics and Communication Engineering, PES University, for her invaluable support and thankful for the continuous support given by the department. I am also very grateful to all the professors and non-teaching staff of the Department who have directly or indirectly contributed to our researchtowards enriching work.

Chancellor **Dr. M R DORESWAMY**, Pro-Chancellor **Prof. JAWAHAR DORESWAMY**, Vice Chancellor **Dr. SURYAPRASAD J**, the Registrar **Dr. K S SRIDHAR,** and Dean of Engineering **Dr. KESHAVAN B K** of PES University for giving us this wonderful opportunity to complete this project by providing all the necessaries required to do so. It has motivated us and supported our work thoroughly.

We are grateful and obliged to our family and friends for providing the energy and encouragement when we need them the most.

Abstract

By improving mobility, road safety, freight productivity, and reducing traffic congestion, autonomous vehicle development has the potential to fundamentally revolutionize both transportation and society. We focused on creating an algorithm that will use machine learning to do path following, obstacle recognition, and collision avoidance. After which, implementing that algorithm in the robot car model. The Raspberry Pi served as the master device in our arrangement, while the Arduino served as the slave. We will send instructions to the slave device, which will use the H Bridge to control the motion of the models' motors, by taking an image with the master device and processing it using an algorithm in the master device. We will program left, right, forward, and backward motions for an automobile model on Arduino Uno. To execute the algorithm, various degrees of left and right turns will be used. Depending on the discrepancy between the frame centre and lane centre, Raspberry will communicate the decimal number to the Arduino Uno. The Arduino Uno will receive the decimal number as input and use that to determine how the automobile model moves. The stop sign will be detected by an algorithm, and the automobile model will halt for a set period. If an obstacle is found, the vehicle will try to change lanes, if not possible it will stop. Additionally, it complies with traffic signals; a red light indicates that a halt is to be made, and a green light indicates that a move is to be made. We can demonstrate the robot automobile that will use our algorithm.

Table of Contents

CONTENTS **PAGE NUMBER**

List of Tables

List of Figures

1. Introduction

The development of completely autonomous vehicles is currently a trend and area of focus in the automotive industry. According to GIDAS, human error is to be blamed for 93.5% of automobile-related accidents. We have received interest in this area as fully autonomous vehicles can greatly reduce accidents that are caused by automobiles. Regarding vehicles' potential safety and comfort benefits, it will be a worthwhile investment to invest our time and resources.

The primary goal of this initiative is to offer a solution to the growing number of traffic accidents and reduce fuel usage by offering drivers a vehicle that can drive independently, recognize the signal, start, and stop when there is a green, or reutilize the appropriate lightingand the ultrasonic sensor, which also benefits the user of not just observing movement but also possess motion-control capabilities for the vehicle using remote mode, and instructions. To achieve the above endeavor, we wish to concentrate on designing an algorithm that does so. We will be able to see the algorithms' results by implementing them on a mechanical platform, which will be useful for upcoming teams working on the continuation of our project. Our main goal is to create an algorithm that will help us drive cars and then further develop it into a completely autonomous driving system.

2. Problem Statement

- Autonomous car is a driverless car that is capable of navigating without human intervention by sensing its environment. They have no driver manually to steer.
- Collision detection and avoidance systems are designed for the safety of autonomous cars. The autonomous car performs these activities without human input.
- These cars use various technologies like RADAR and LIDAR for detecting the surroundings. They also have controlled systems which analyse the collected data. This analysis is useful in planning a path the vehicle should take to reach the desired location.
- For performing these analytical tasks, the data is processed to produce the required output and the car-controlled systems use various machine learning algorithms. Machine learning algorithms train the system to behave in a certain way.
- Hence, our motive is to design an algorithm that performs path following, edge detection, and collision avoidance system considering the surrounding environment. Additionally, to also design an algorithm that assists a Robot Car to obey traffic signals.

3. Literature Survey

3.1 Deep Learning based Image Recognition for Autonomous Driving

(International Association of Traffic and Safety Sciences, 2019)

- This paper enlightens us about how deep learning can be applied to image recognitionand how it is solved. It describes the trend of deep learning based autonomous drivingand related problems.
- The tasks performed task are –
  - ✓ Image Verification: The distance between the feature vector of the reference pattern and the feature vector of the input image is calculated. If this distance isless than a threshold value, images are said to be identical.
  - ✓ Object Detection: Using deep learning, only one network is required to achievemulti class object detection. Face detection uses a pair of Haar and AdaBoost features. Pedestrian detection uses HOG features and support vector machine (SVM).
  - ✓ Image Classification: Deep learning achieves an accuracy better than human recognition performance in the 1000 class image classification task.
- A type of deep learning, known as Convolution Neural Network (CNN), is an approach for learning classification and feature extraction from training samples. This paper focuses on object detection, semantic segmentation, and Advanced Driving AssistanceSystem (ADAS) using CNN.

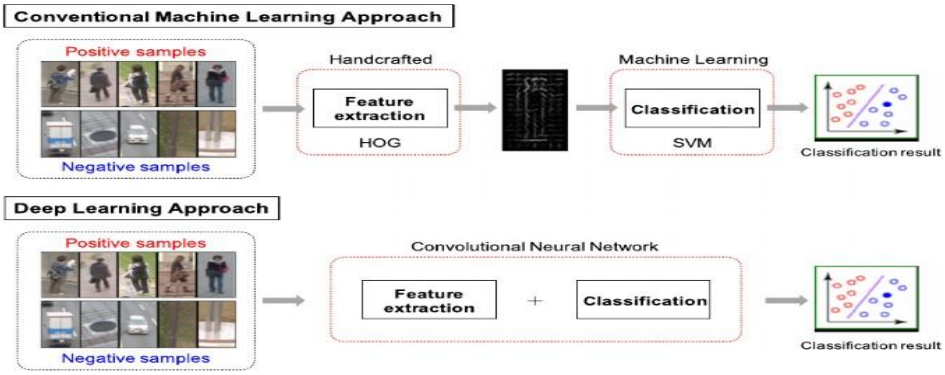


Fig: 3.1.1 – Comparison between Machine Learning and Deep Learning

- CNN is more advantageous than conventional machine learning. CNN computes the feature map corresponding to the kernel by convoluting the kernel on the input image.Feature maps corresponding to the kernel types can be computed as there are multiplekernels. The size of feature map is minimized. Hence, it allows absorption of geometrical variations and rotation of the input image.
- In conclusion, this paper introduces the latest image recognition technology andaccomplishes image recognition tasks using deep learning. In the future, recognition ofinput images as well as high expectation of the end-to-end learning and deep reinforcement learning technologies for control and judgement of the autonomous vehicle can be achieved.

3.2 Design and Implementation of Autonomous Car using Raspberry Pi

(International Journal of Computer Applications, 2015)

- This paper focuses on designing and modelling a prototype of a car that can reach the desired destination safely, ruling out the factor of human error.
- Human Errors are on the rise today, and considering the number of advancements intechnology, humans cannot resist the urge to use their cell phones while driving.
- Hence, needing the requirement of developing a car that can reach the destination allby itself, and providing the leverage for the passengers to continue with their work.
- Hardware used in this paper are –
  - ✓ Raspberry Pi – Model B Rev 2, with 512 MB RAM, two USB ports and 10/100Ethernet Controller.
  - ✓ Pi Camera – It can click HD photographs or videos of the roads for performinglane detection.
  - ✓ Ultrasonic Sensors – Used to detect the distance of obstacles from the car. Helpsin determining when to stop the car or deviate to another lane.
- Other hardware components used here are – Wi-Fi 802.1 dongle to remotely connect the Pi, Motor driver IC L293D to control the two motors, Batteries to

supply power, Jumper wires to make connections, L shaped aluminium bar to support and hold the camera, Servo motor to make the camera flexible.

- Software used are – Python, RPi.GPIO Python Library, Open CV.
- The model performs Lane Detection by extracting the colour range of the road. Further,the region of interest (ROI), as shown in the figure, is defined from the nearest to the farthest region by moving upwards in the image created. Height of ROI is generally not more than half of the height of the image

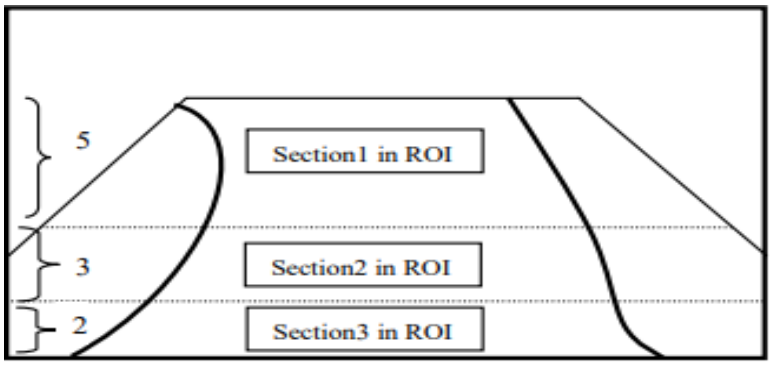


Fig: 3.2.1 – Sections of Region of Interest

- The complex region of interest is further converted into a simple shape. This step is vital in determining the borders of the road. The largest contour which contains the central part of the bottom region is the road.
- The shape of the road is then determined by drawing several Hough lines on the manipulated contour, along the left and right edges of the road. Only few lines, out of many, represent the actual edge. The other lines exist due to noise, which needs to be filtered out. A line with a small angle or having least x-intercept or y-intercept is chosenas both the edges of the road.
- For the car, the change in direction of the road cannot be abrupt. The edges of the roadcannot be changed discretely. Hence, a line that is far away from the previous frames line will be discarded.
- Thus, it determines the turns in the road and gives direction to the car by dividing the ROI in the ratio 2:3:5 and the lines obtained in all three parts are compared with the possible shape of the road.

- Obstacle Detection using an Ultrasonic Sensor is performed as shown:

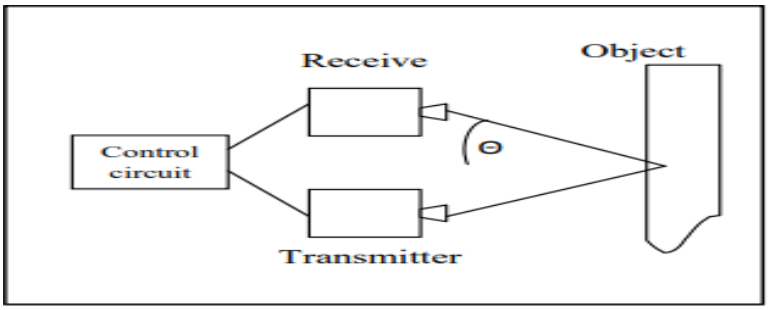


Fig: 3.2.2 – Working of Ultrasonic Sensor

- In conclusion, a novel method of determining uneven, marked, or unmarked road edgesis elaborated relying on OpenCV. Using ultrasonic sensors, the task of collision avoidance is accomplished. The algorithm used in this paper has been successfully implemented in a small autonomous robot car.
- The work can be enhanced by adding machine learning to it. In this algorithm, operations are performed on all the frames. It is sufficiently accurate, but the efficiency can be improved if it is trained and hence, avoid unnecessary calculations of the regionsthat are familiar. The distance between the nodes, dimensions of the obstacle, the number of speed breakers and static obstacles are among the few that can also be determined.

3.3 Path Planning for Autonomous Vehicles with Dynamic Lane Mapping and Obstacle Avoidance

(13th International Conference on Agents and AI, 2021)

- This paper presents a waypoint-based path planning architecture with LiDAR based localization, dynamic lane mapping and static/moving obstacle avoidance.
- They utilize road width information at each waypoint from real-time drivable area datato dynamically map the positions of lanes according to a well-defined width of the lane.
- A novel path planning system for autonomous vehicles, taking into consideration the position of surrounding obstacles and position and trajectory

of moving obstacles are taken into consideration and it generates a path and trajectory plan for the vehicle validfor the next few seconds, making the vehicle safely manoeuvre away from the obstacle.

- The overview of the system architecture used in this paper is as shown:

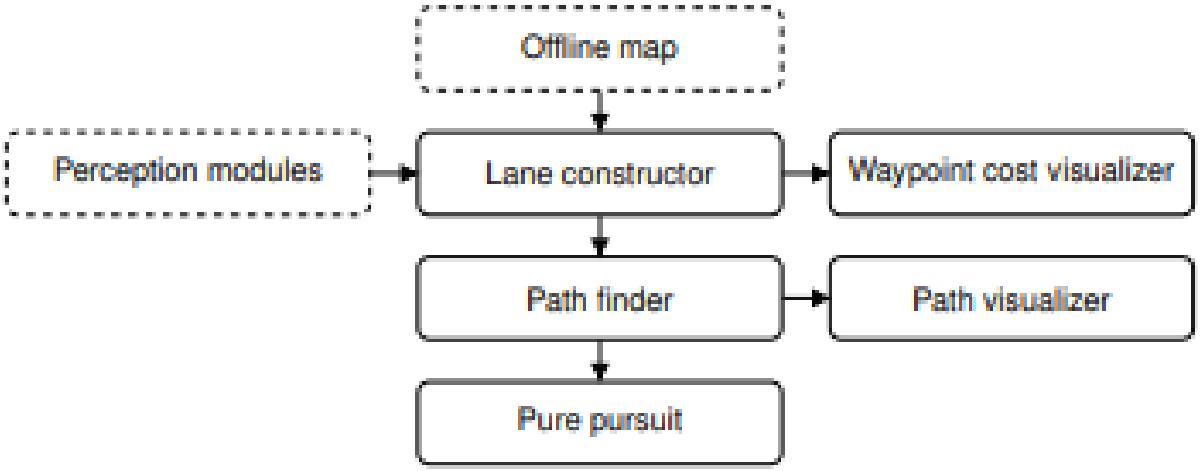


Fig: 3.3.1 – System Architecture of Autonomous Vehicle

- ✓ Offline Map: It is a directed graph of GPS coordinates called waypoints, used to mark a section of the road. The lateral position of the vehicle does not dependon the lateral position of the waypoint on the road. Hence, they cannot be usedto mark lanes.
- ✓ Lane Construction: Since waypoints do not carry any information about the road dimensions or position, they construct their own lane position for the vehicle. Since waypoints have a heading, they construct a line at the waypoint position normal to the waypoint heading.
- ✓ Cost Visualizer: The cost value in the range [0,1] is assigned to each lane waypoint. A waypoint having a cost closer to 1 is likely to be in close vicinity of the obstacle. For safety purposes, the speed of the vehicle is linked to the costi.e., as cost approaches 1, the driving speed approaches 0.
- ✓ Path Finding: Here, the best path in a lane waypoint grid is sculpted as a shortestpath problem. Lane waypoints are a directed graph where each node is connected to the next node in the same lane. Simultaneously, the nodes in the left and right lanes signify a lane change.

- ✓ Vehicle Trajectory: The pure pursuit algorithm is used to steer the vehicle alongthe generated path. It finds a curvature along which the vehicle moves from itscurrent position to the final position.

- Thus, the performance of the autonomous vehicle is evaluated using two methods, Simulation Test and Field Test.

  The LGSVL simulator provides a 3D environment for testing the functionalities of theautonomous vehicle. The simulator integrates with the ROS platform, thus providing a vehicle model which is controlled using ROS commands and simulates sensors like camera, LiDAR, and GPS.

- A live demo of the path planning modules is performed using a modified electric golf cart. Data processing and vehicle control is done on a computer running the ROS platform. The steering column and throttle actuators of the vehicle are connected electrically to an Arduino, which receives commands from a ROS node running on thevehicle computer through a serial connection.

3.4 Raspberry Pi based Autonomous Car

(Lokesh M. Giripunje, Manjot Singh, Surabhi Wandhare, Ankita Yallawar, Department of Electronics and Telecommunication Engineering, Dr. D. Y. Patil Institute of Engineering, Management and Research, Akurdi, Pune)

- The primary purpose of this paper is to provide a solution to increasing road accidentsby providing an autonomous car that can drive itself, detect the traffic lights, in turn stopping the car if there is the red signal and moving the car if there is the green signal.It also performs the task of collision avoidance.
- It also makes the user capable of not just monitoring the movements but also able tocontrol the vehicles' movements and directions using a remote mode.
- The flowchart of the working model is as shown:

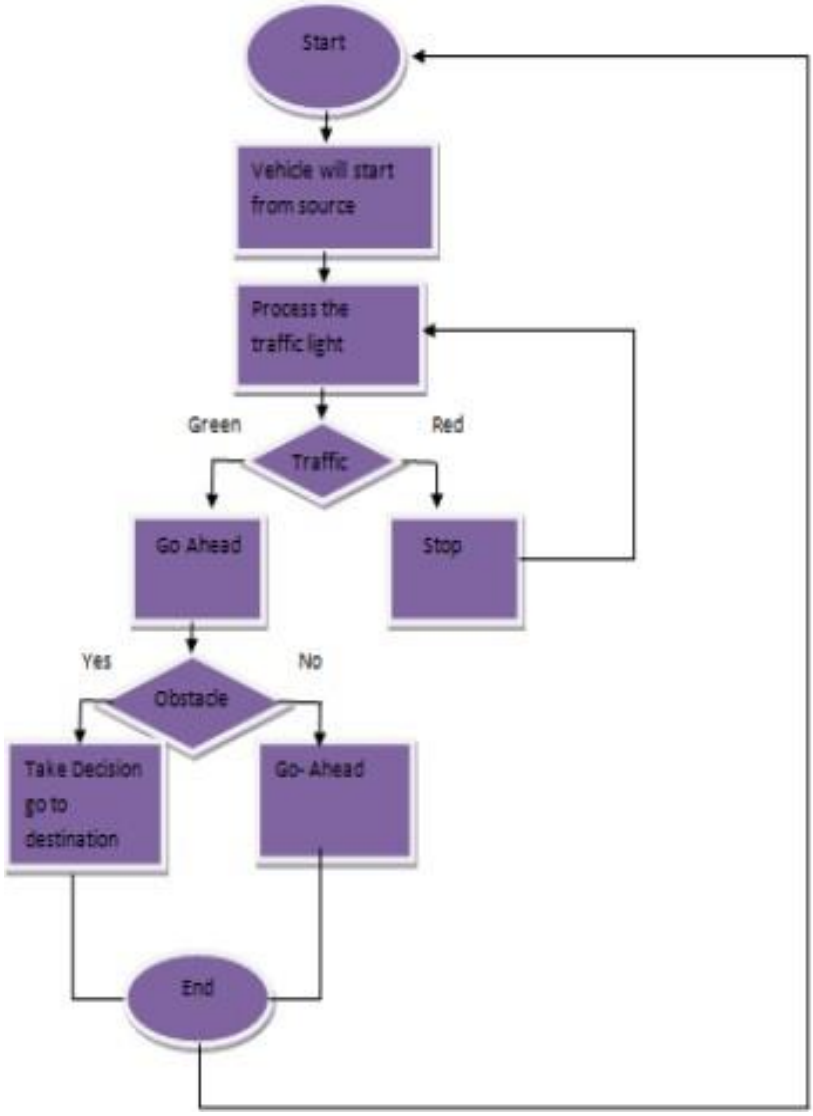


Fig: 3.4.1 – Working of Raspberry Pi based Autonomous Car

- First, the car starts from the source. On detection of traffic lights, the camera will capture the image of the signal and process it. If the traffic signal is red, it will stop thecar for a while and wait for the signal to become green. Once it turns green, the car starts moving again.
- If there is any obstacle while moving along the path, then it will stop and change its path and reach the desired destination. If there are no obstacles, it will continue along the same path and reach the destination.
- The Hardware components used in this project are:
  - ✓ Power Supply – Additional two pins are provided for connecting a transformer which will be used to charge the battery. It will also provide DC voltage to the

bridge rectifier.

- ✓ Raspberry Pi – This is the main component of the project, which is trained to identify images, compare, and finally implement the algorithm. It will take

  immediate action most suitable for the input image.

- ✓ Motor Drivers – The circuit is used to drive the two DC motors, which in turn move the model. The motor driver used here is L293D, which can drive both the motors simultaneously.

- ✓ Ultrasonic Sensor – There are two pins namely, Trig and Echo. It is used to measure the distance between the model and the obstacle.

- ✓ Camera – It is connected to the Raspberry Pi through a USB wire. It will captureimages from the surroundings and input it to the Raspberry Pi for further processing.

- The software used is OpenCV (Open-Source Computer Vision). It is used in this project to train the processor to distinguish between positive and negative images. Thealgorithms can be used to identify objects, track camera movements.

- The system architecture used in this paper is as follows:

3.5 Autonomous Mode –

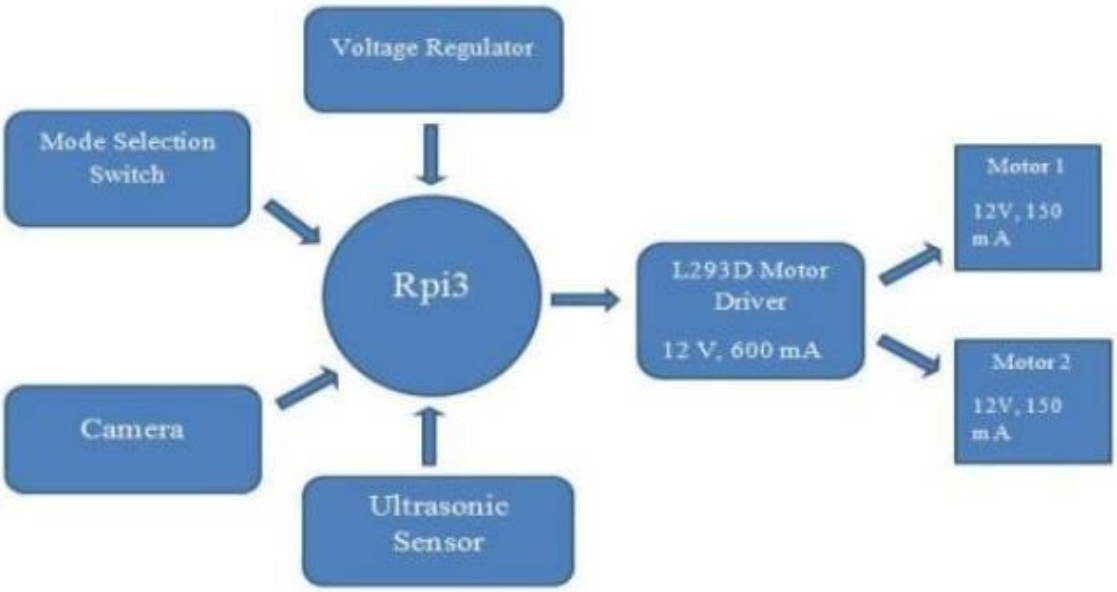


Fig: 3.4.2 – System Architecture of Autonomous Mode

- In the Autonomous Mode, the raspberry pi is used mainly to process images of the signal lights that will be captured by the camera. It also measures the distance from obstacles using ultrasonic sensors and thus reduces movement of the motor driver according to the program written.
- Here, if the distance between the vehicle and the obstacle is less than 2m, then the program is coded in such a way as to stop the vehicle immediately.
- They are using LM7805 as a voltage regulator to provide 5V to the Raspberry Pi.

3.6 Remote Mode –

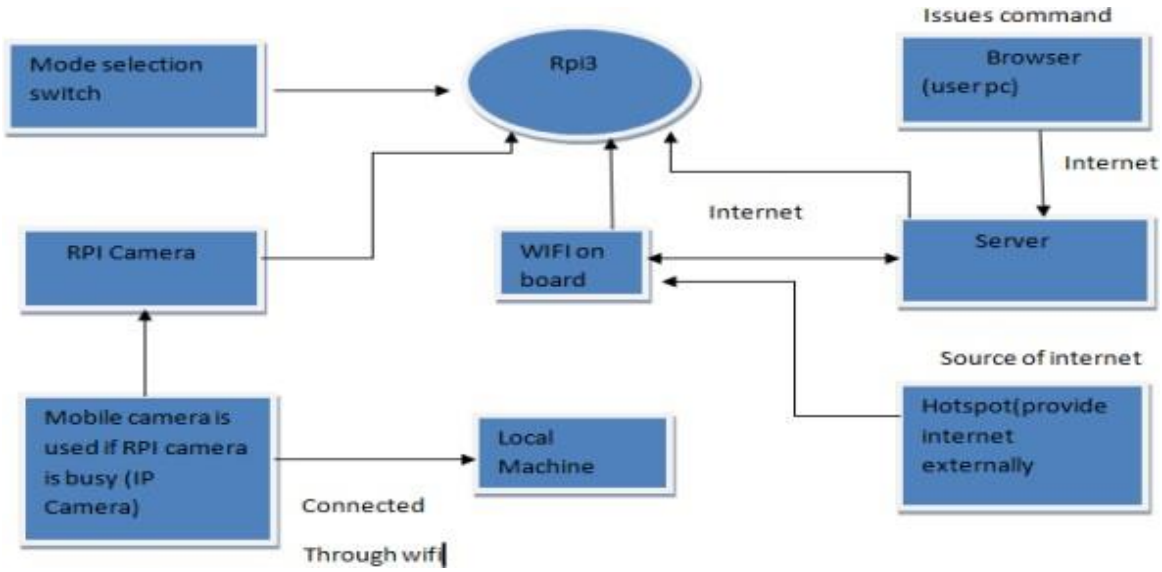


Fig: 3.4.3 – System Architecture of Remote Mode

- In this mode, they can control the operation of the car and its movements using remotemode of operation, where all the collected information will be sent to a device like mobile or laptop using IoT. It also enables them to control motions like left, right, front,back and stop.
- Here, the Internet is provided using hotspot externally. If the Raspberry Pi camera is busy processing images, then the person can use his mobile camera for processing.
- In summary, the Autonomous Car in this project can achieve the goal of transportation capabilities of a conventional car, without any input from the human, and hence, reducing human error and preventing accidents.
- Any senior citizens, disabled humans, or children willing to travel can make use of thisto commute safely to different places, without depending on anyone else.
- This project uses convenient and affordable ways to provide a safe, driverless

journey without any hindrance. It also provides mode selection provision to the humans througha switch, where they can choose if they want to drive the car by themselves or otherwise.

- Even a small number of autonomous cars can have a major impact in eliminating wavesand reducing fuel consumption.
- The future can be enhanced by using machine learning which helps to improve the performance as efficiency increases if the algorithm starts learning by itself and hence,avoid wastage of time in doing calculations.
- Additionally, the vehicle can keep track of obstacles encountered while travelling andsave the data for future references or similar circumstances.

4. Implementation

4.1 – Hardware Architecture

The block diagram is as follows:

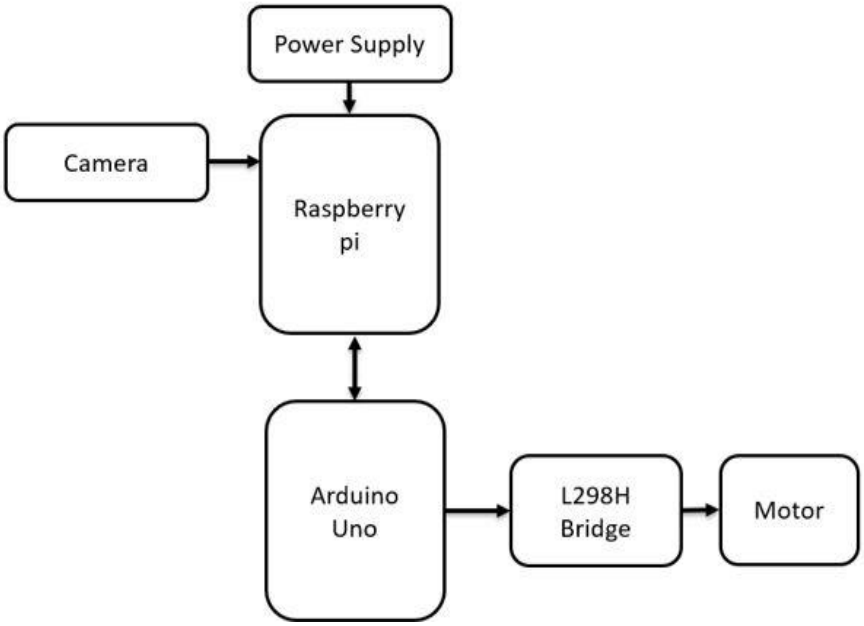


Fig: 4.1 – Block Diagram

4.1.1 – Hardware Components Used:

- **4-Wheel Robot Chassis:**

Structural component for the prototype car which offers a large space with predrilled holes for mounting sensors and electronics. It gives the mechanical platform required for buildingthe prototype car.

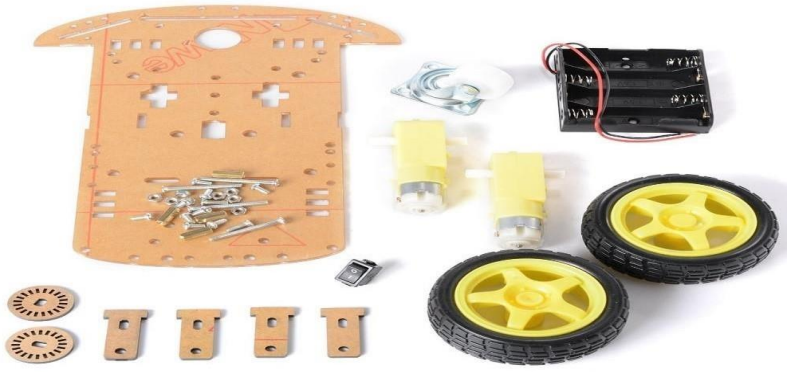


Fig: 4.1.1.1 – 4 Wheel Robot Chassis

Specifications:

Operating Voltage	3 - 6 V
Shaft length	8.5 mm
Shaft Diameter	5.4mm and 3.5mm
Speed	200 rpm
Torque	0.8 Kg-cm
Dimension	70 x 19 x 22 mm
Weight	250 gms

Tab: 4.1.1.1 – Specifications of Robot Chassis

L298 H Bridge

It is a dual H-Bridge used for driving a load such as a brushed DC motor connected to the wheels of a prototype car. It also allows for controlling the speed and direction of the motorat the same time. The module can drive a DC motor that has voltage between 5 and 35V with a peak current of 2A.

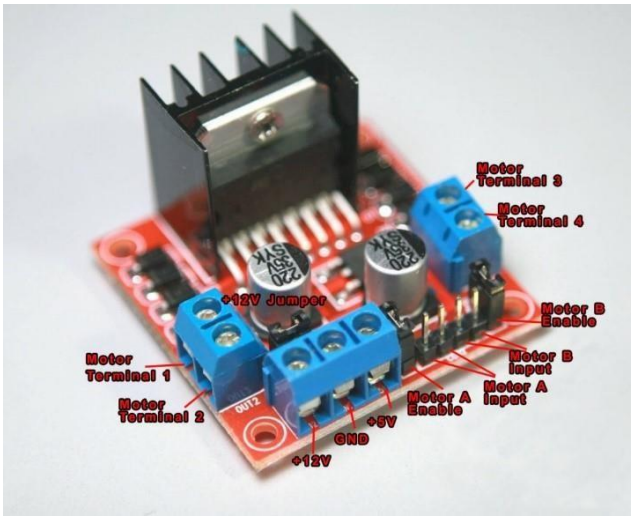


Fig: 4.1.1.2 – L298 H Bridge

➢ Arduino Uno

It is a microcontroller board that is used to issue commands to L298 H Bridge. It is responsible for turning off the robot based on PWM i.e., maintaining the RPM of the robot motors.

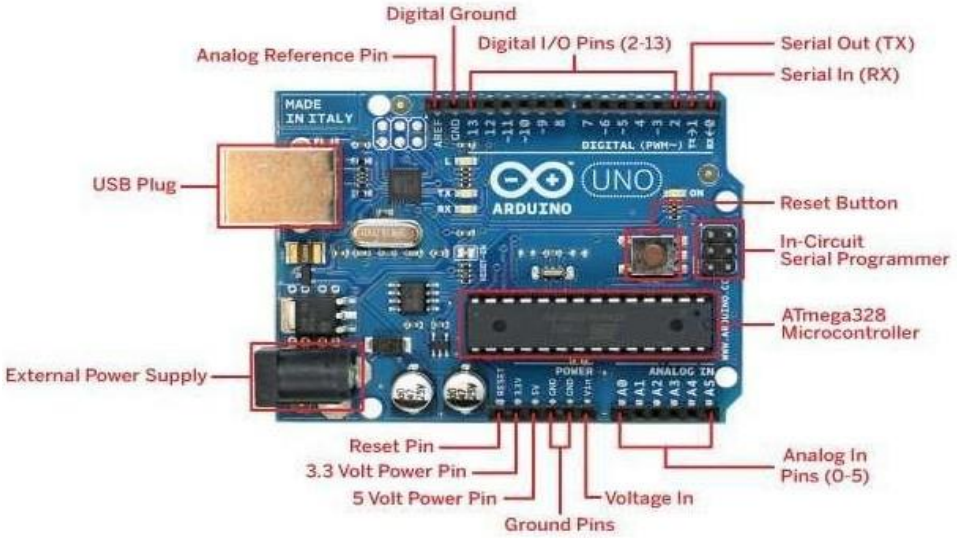


Fig: 4.1.1.3 – Arduino Uno

Specifications:

Microcontroller	Atmega328p
Operating Voltage	5V
Input Voltage	7-12V
Digital I/O Pins	14 (6 provide PWM output)
Max Current Rating	40mA
USB	Mini
Flash Memory	16 KB
SRAM	2KB
EEPROM	1KB
Clock Speed	16MHz
LED_BUILTIN	13

Tab: 4.1.1.2 – Specifications of Arduino Uno

➢ Raspberry Pi 3 B+

It is a 64-bit quad-core processor. It is a microcomputer which acts as a master device for a prototype car and processes the information in real-time. It issues the instruction for Arduinoto control the prototype car. It is the main processing unit of the robot. The specifications are as follows:

CPU Type	ARM Cortex-A53
Speed	1.4GHz
RAM Size	1GB
Integrated Wi-Fi	2.4GHz and 5GHz
Ethernet Speed	300Mbps
PoE	Yes
Bluetooth	4.2
Input power	5V/2.5A DC via micro-USB connector

Tab: 4.1.1.3 – Specifications of Raspberry Pi 3B+

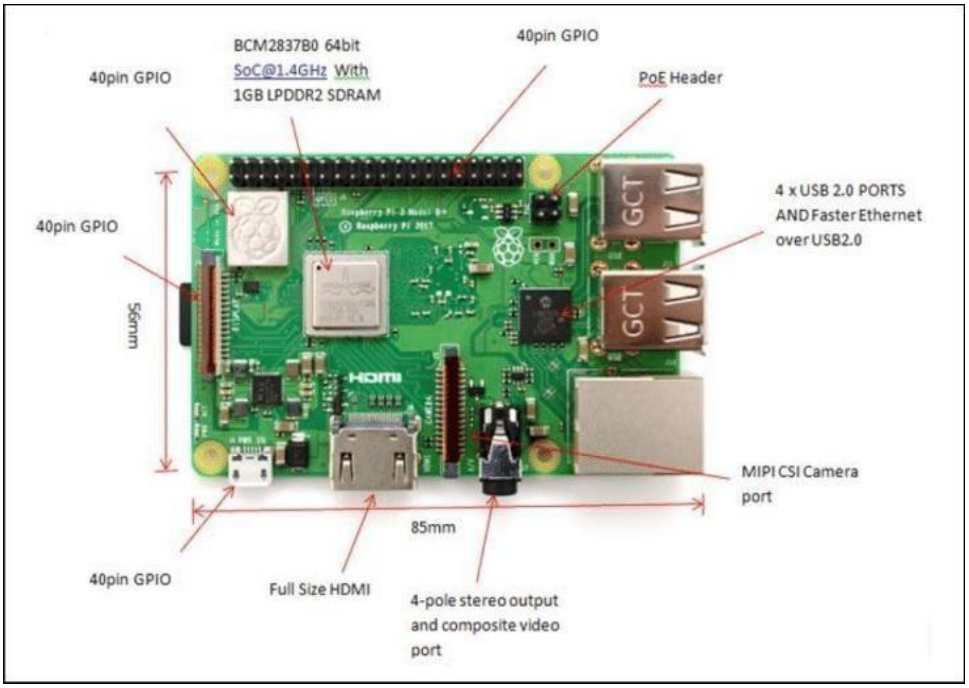


Fig: 4.1.1.4 – Raspberry Pi 3B+

- **Raspberry Pi Camera**

This camera module has a 5-megapixel Omni Vision OV5647 sensor for capturing photographs and videos. It comes with a cable to connect the MIPI CSI Camera port presenton Raspberry Pi.

Fig: 4.1.1.5 – Raspberry Pi Camera

- **SD Card –** 32 GB portable storage device is used as a storage device for Raspberry Pi.
- **Power Bank –** A 10000 mAh power bank is used as the power for all the components present in the prototype car model.
- **Other Components Used –** Jumper wires, USB cables, LEDs, Traffic Light Indicator, Copper wires.

4.1.2 – Hardware Design

To implement the algorithm of advanced self-driving assistance, we planned to build a prototype car model. To process real-time information, we need a microcomputer aa s centralprocessor unit. It will process the information taken from the camera using an algorithm. Here we will use a master-slave device setup between the microcomputer and microcontroller to make the circuit edge sensitive which helps to control and enhance theperformance of the prototype of the car model.

DC motor is used to rotate the wheels present in the robot car chassis. To have freedom of rotation and control the speed of both wheels at the same time, DC motors will be connected to H Bridge. H Bridge will act according to a command issued by the microcontroller. The role of uno will be to act as a mediator between sensors and the commanding machine. It will be a slave device which will act according to the commands instructed by the master device to control the movement of the prototype.

The sensors present in the prototype car model will make a large impact on the result of the algorithm implemented in it as the quality of information obtained by sensors will influence the performance of algorithms. The materials used to build a prototype model also make adifference. Its weight must be minimal to avoid the stress on motors. The power source usedon the model should be able to handle the power requirement required by the prototype car model.

The requirement for the prototype car model is shown in the table below:

Microcomputer	Raspberry Pi 3B+
Microcontroller	Arduino Uno
H Bridge	L298 H Bridge
Camera	Raspberry Pi Camera
Structural Component	Robot Car Chassis
Power Source	10000mAh Power bank

Tab: 4.1.2.1 – Requirements of Prototype Car

4.1.3 – Construction of the Prototype Model

The model is designed to show the performance of the algorithm that we developed for advanced driving assistance. To process the real-time information with efficiency, RaspberryPi 3B+ is required. To increase the circuit performance, we utilized a master-slave device setup. Arduino Uno is a microcontroller to coordinate the speed and direction of DC motors.

1. **Assembling the Robot Car Chassis:**

   Using the materials that are provided in the kit, we assembled the platform for hosting the components that are required to build a prototype car model. DC motors terminals will be soldered for connecting copper wires.

2. **Connecting DC motors to L298 H bridge:**

   We made the two sets of cross-coupled motors placed in parallel with each other and mounted the L298 H Bridge in the chassis. The two parallel motors are connected to each port on the bridge.

3. **Mounting the Power Bank:**

   We mounted the 10000mAh Power bank in Chassis. Two USB cable connectors have been inserted in the ports available in Power Bank.

4. **Powering up L298 H bridge:**

We connected one USB Connector to the 5V and GND port in L298 H Bridge. As the connection between the power bank and L298 H Bridge is established LED present in theL298 H bridge will switch on.

5. **Mounting Arduino Uno and connecting with L298 H Bridge:**

We mounted the Arduino Uno on the chassis. Using six female-three male jumper wires, where the female side of the jumper wire is connected to L298 H Bridge and male side ofjumper wire is connected to Arduino Uno.

6. **Mounting Raspberry Pi 3B+ in the chassis:**

Raspberry Pi is mounted on the chassis along with the case with a heat sink.

7. **Powering Up Raspberry Pi 3B+:**

Micro USB cable from PCB is connected to micro-USB port of Raspberry Pi.

8. **Mounting camera on the model:**

The camera is connected to CSI camera port of Raspberry Pi.

4.1.4 – Master Device Setup

In our project, we used Raspberry Pi 3B+ as a master device. We try to connect this masterdevice to our Personal computer.

Firstly, we need to have an operating system. Thus, we flashed the Raspbian OS on Raspberry Pi. We connect Raspberry Pi to our Personal computer using an Ethernet cableand a micro-USB cable which helps in the power supply. The ethernet cable helps in communication with our Personal computer and resource sharing.

We must install an Advanced IP scanner application to find Raspberry Pi in the network and allocate it with an IP address. Then using Remote Desktop Connection, we can cast the Raspberry Pi on to our Personal Computer.

4.1.5 – Slave Device Setup

Pin	Left Motor	Right Motor
Enable	9th Digital Pin	10th Digital Pin
High	7th Digital Pin	5th Digital Pin
Low	6th Digital Pin	4th Digital Pin

Tab: 4.1.5.1 – Slave Device Pin Assignments

The speed of motors will be controlled by using pulse width modulation. Hence, we used PWM pins. For forward and backward movements of motors, we used the Arduino Nano forcontrolling the voltage supplied to the motor using logic. To change the direction of motors we will control the speed of motors. For instance, to make the car turn left, we would lower the speed of the left motor, and to have the car turn right, we would lower the speed of the right motor.

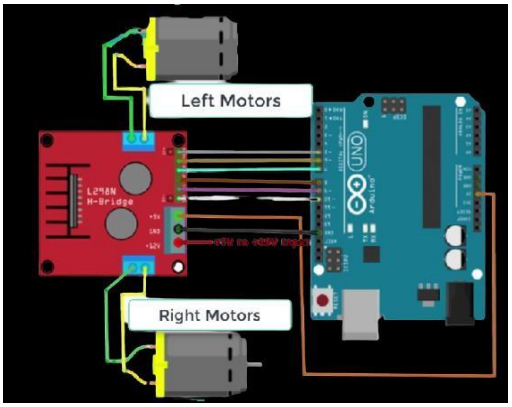


Fig: 4.1.5.1 – Slave Setup

4.2 – Lane Detection

The steps involved in Lane Detection are:

1. Capturing Images and Videos:

To obtain real-time information needed for the execution of algorithms wewill capture information from the surrounding environment of the robot car.

We will use the raspicam module in C++ to capture the images from a camera mounted on Raspberry Pi. We will use Raspicam_CV class in the code.We will set the value for frame width, frame height, brightness, contrast, saturation, gain, and frame per second, thus we can focus on what we needed from the surroundings to increase efficiency.

To capture the video, we will use the while loop in C++ for continuously taking theimages and forming a video.

Calculating Frame Per Second (FPS): We are using the chrono library in C++ to calculate the time taken for capturing one image

T = time after grabbing and retrieving the image – time just before grabbing the image

FPS = 1/T

2. Converting Image Signature:

BGR displays tend to make text look blurry or fuzzy, which may haveadverse effect on display of the surroundings on the output screen. We needed RGB image format. We used the OpenCV library's function cvtcolor to convert BGR color space into RGB color space in the output frame.

3. Creating Region of Interest:

Creating a region in the environment that we need as input for the prototype carmodel's algorithm to work is called creating a region of interest.

Many items from the environment are caught by the camera in a frame. However, we only need to concentrate on the lane that is visible in the frame. So, by setting up a setof locations inside the frame, we will obtain the lane-required region.

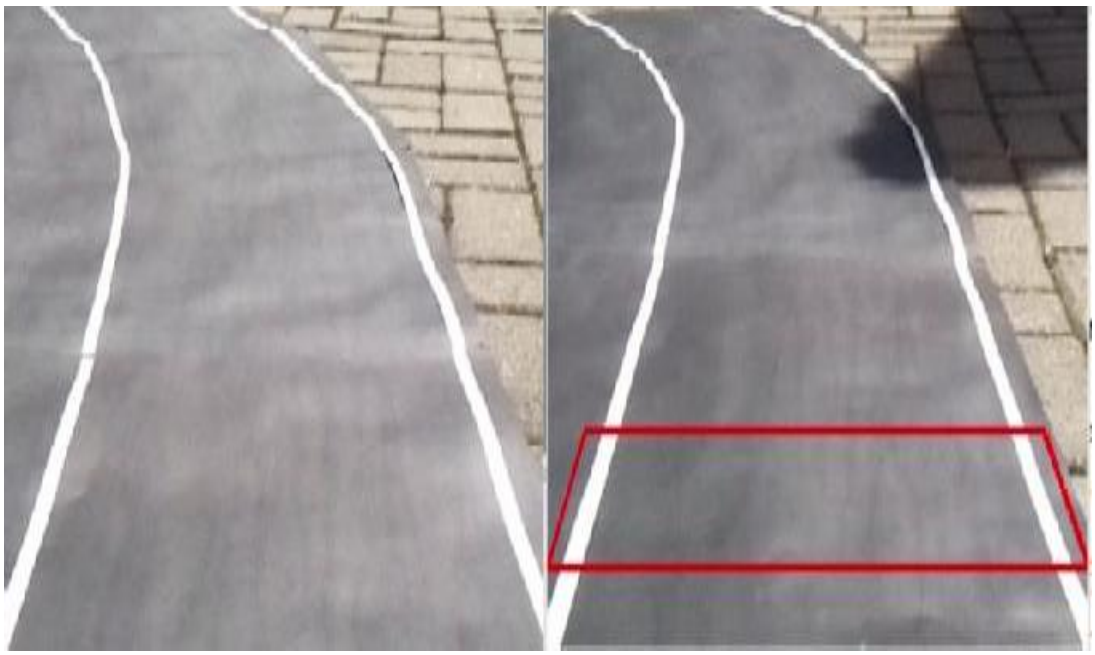


Fig: 4.2.3.1 – Required Region of Interest

4. Perspective Transformation:

Can also be called bird eye view. When something is close to the human eye, it appears larger than when it is far away. This is generally referred to as perspective. As opposed to transformation, which involves moving an object, etc., between states.Overall, the perspective transformation involves turning a three-dimensional reality into a two-dimensional image. The same underlying theory underpins both the operation of a camera and human eyesight.

To do this operation we use two functions getPerspectiveTransform and warpPerspective. Where, getPerspectiveTransform takes 4 points of corresponding points in the frame and outputs a transformation matrix M. Transformation matrix M is used by warpPerspective, which applies the perspective transformation to an image.Warp Perspective applies the perspective transformation to an image by using the transformation matrix M.

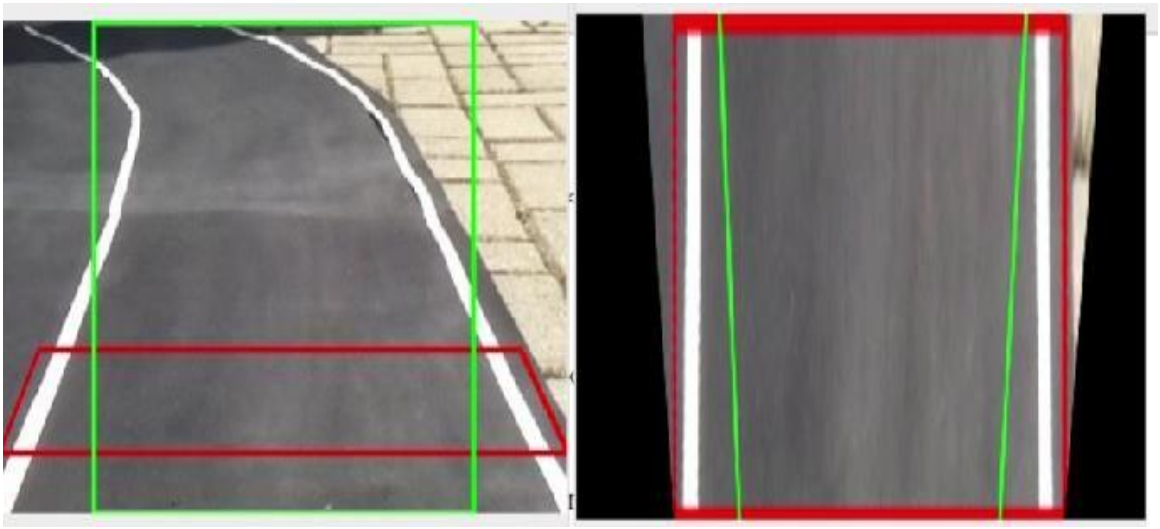


Fig: 4.2.4.1 – Perspective Transformation

5. Threshold Operation:

Image thresholding is a straightforward but efficient technique for separating an image's foreground from its background. By transforming grayscale photos into binary images, this image analysis technique is a sort of image segmentation thatisolates objects.

Here, we'll use the OpenCV library's cvtcolor function to convert an RGB image intoa GRAY image to get black-and-white intensities.

We'll employ the inRange function from the OpenCV package to apply the thresholdaction. We'll classify white as having an intensity of 200 to 255.

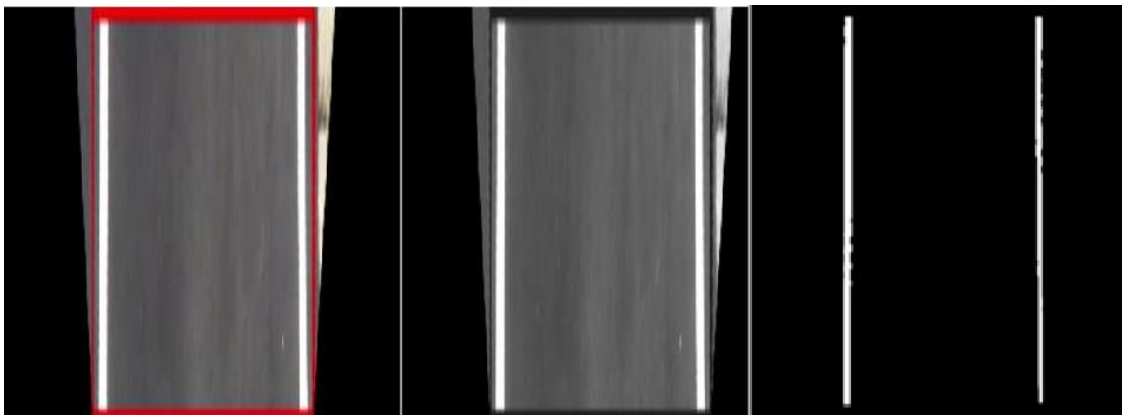


Fig: 4.2.5.1 – Threshold Image

6. Canny Edge Detection:

With the use of the Canny edge detection technology, the amount of data that needs tobe processed can be drastically reduced while still extracting meaningful structural information from various vision objects. It is frequently used in many computer vision systems. According to Canny, the prerequisites for applying edge detection to various vision systems are largely the same. Thus, a solution for edge detection that meets these needs can be used in a variety of contexts. The basic standards for edge detection are as follows:

- ✓ Edge detection with a low error rate, which means that the detectionshould precisely capture all the edges visible in the image.
- ✓ The operator's edge point detection should precisely locate on theedge's centre.
- ✓ If at all feasible, picture noise should not produce false edges, and an edge in the image should only be marked once.

Canny used the calculus of variations, a method for locating the function that best optimizes a given function, to meet these conditions. The first derivative of a Gaussian can be used to approximate the optimal function in Canny' s detector, whichis represented by the sum of four exponential terms.

The Canny edge detection algorithm is one of the most precisely specified edge detection techniques that has been created to date. It offers accurate and dependabledetection. It quickly rose to the top of the list of algorithms because of how well it met the three requirements for edge detection and how straightforward the implementation process was.

In this case, we performed canny edge detection using the canny function from theOpenCV library, where the parameters image, lower and higher threshold value inhysteresis thresholding, and aperture size of the Sobel filter are defined. Using the add function from the OpenCV library, as shown in the picture below, we combined the results of the threshold operation and the canny edge detection image. Itmakes a clear separation between the white lines, which indicate lanes, and black lines, which represent everything else.

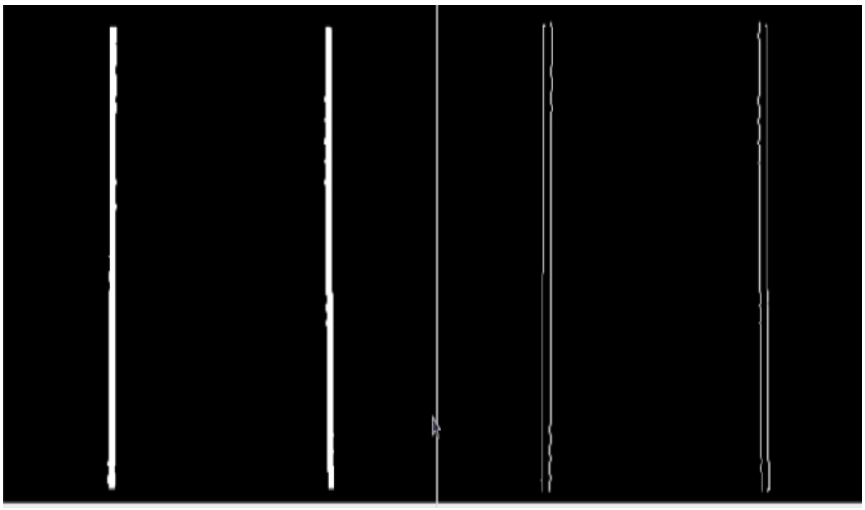


Fig: 4.2.6.1 – Canny Edge Detected Lanes

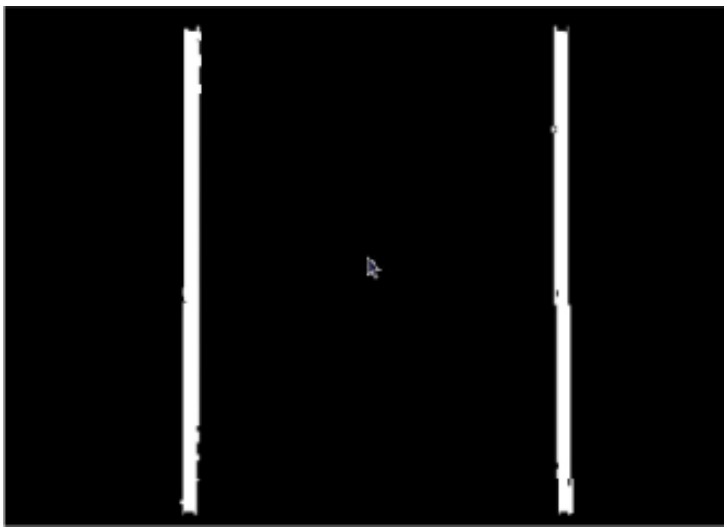


Fig: 4.2.6.2 - Canny Edge Detected Lanes

7. Finding Lane from Track:

We will find the lane's exact location in the frame using mathematical calculations.As shown in the figure,

In a frame we will choose the Rect(x, y, x+width, y+height)

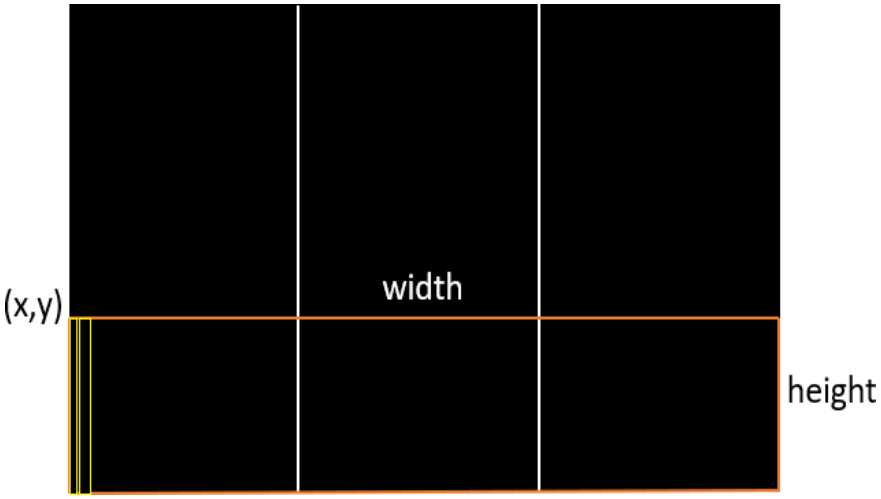


Fig: 4.2.7.1 – Finding Lanes from the Track

The rectangle will be divided into thin strips, or what is known as a histogram, with awidth of one unit and a height of y. We will have 400 strips if the width is 400. If weassign the strips to a dynamic array, each strip will stand for the intensity of a particular hue. When color intensity falls between the 150 and 255 range, lanes are present.

8. Histogram and Vectors:

We will create a dynamic array known as a vector in C++ code. The size of the dynamic array will be the width of the rectangle, considering the width of each stripor histogram is 1.

We will create a region of interest in the frame of the final output we got from addingthe outputs of threshold operation and canny edge detection. We will apply the required rectangular coordinates for the region of interest.

For example, a vector

V = [0, 0, 0, 0, 0, …., 245, …., 246, …, 0, 0, 0]

9. Iterators and Pointers:

By iterating the entire vector V, we find the position of lanes.

We must first divide the array into two parts. One part runs from position zero to position width/2. From the width/2 position to the end position, another part begins. We will determine the left lane location by repeating the first part. Similarly, by repeating the second part, we can locate the position of the right lane. By drawing aline at the lane's location and checking to see if it correlates through any white linesin the final output, we obtained by adding threshold and canny edge output, we can verify the position.

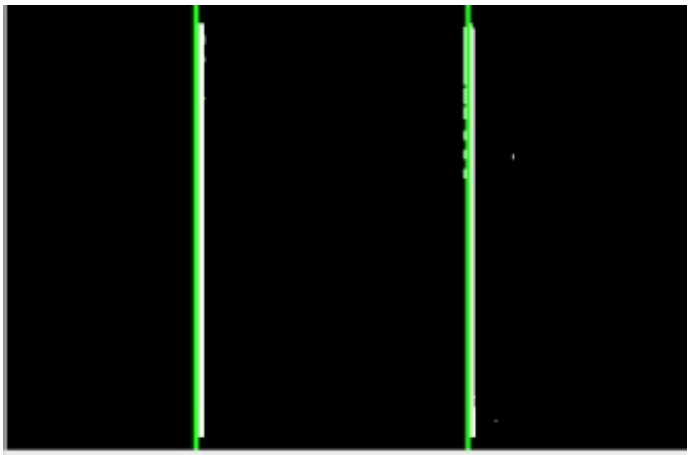


Fig: 4.2.9.1 – Iterators and Pointers

10. Calibration:

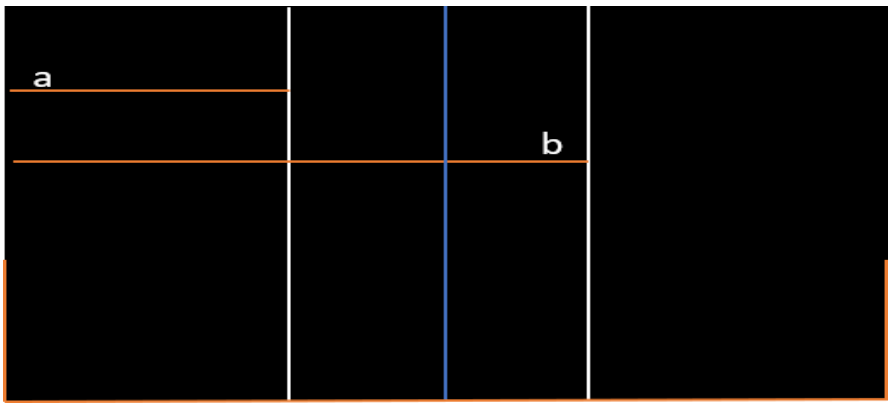


Fig: 4.2.10.1 - Calibration

The formula for finding the lane center:

$$Lane\ center = Right\ lane\ position - Left\ lane\ Position\ 2 + Left\ lane\ position$$

The frame center will be half of frame width.

We must ensure that the frame center and lane center coincide for the prototype carmodel to operate without a hitch.

11. Concluding Act:

We will calculate the difference between the frame center the and lane center.

Difference = Lane Center – Frame Center

If it's positive, the model is tilted to the left, and ifit's negative, the model is tilted to the right.

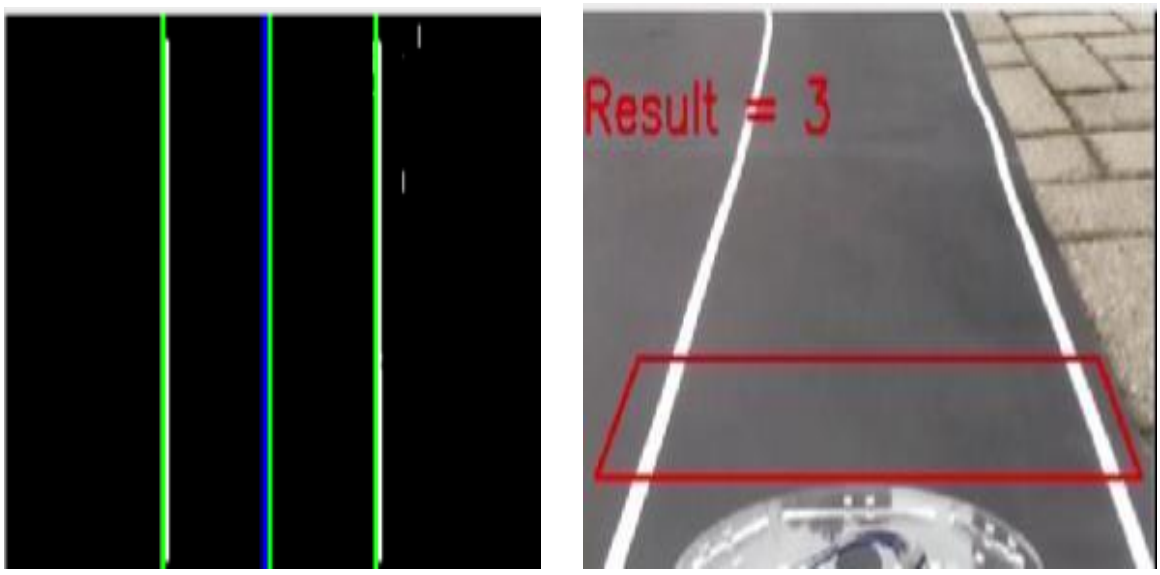


Fig: 4.2.11.1 – Lane Detection

4.3 – Master Slave Communication

To program the Raspberry Pi digital pin, we'll utilize the wiringPi library. The Arduino Uno is linked to pins 21, 22, 23, and 24.

We will utilize the conditional statement to keep the prototype car traveling in the middle of the lane. We will choose which level of the right or left turn it must make bytaking the difference between lane center and frame center into account. The output of this process will be transferred to an Arduino Uno board, which will use it to control the speed of each motor to achieve the desired outcome.

The three names of each pin are shown in Figure, which also shows which pins the Raspberry Pi can use in PWM OUTPUT mode or GPIO CLOCK mode.

Digital pins 1, 2, 3, and 4 will be used to connect to the Raspberry Pi. To read the data coming from the Raspberry Pi, we will use the digitalRead function.

For instance, if a data point's decimal value is 0, it signifies that the frame and lane centers are identical. We will be able to regulate the direction the prototype car modelmust move in this method.

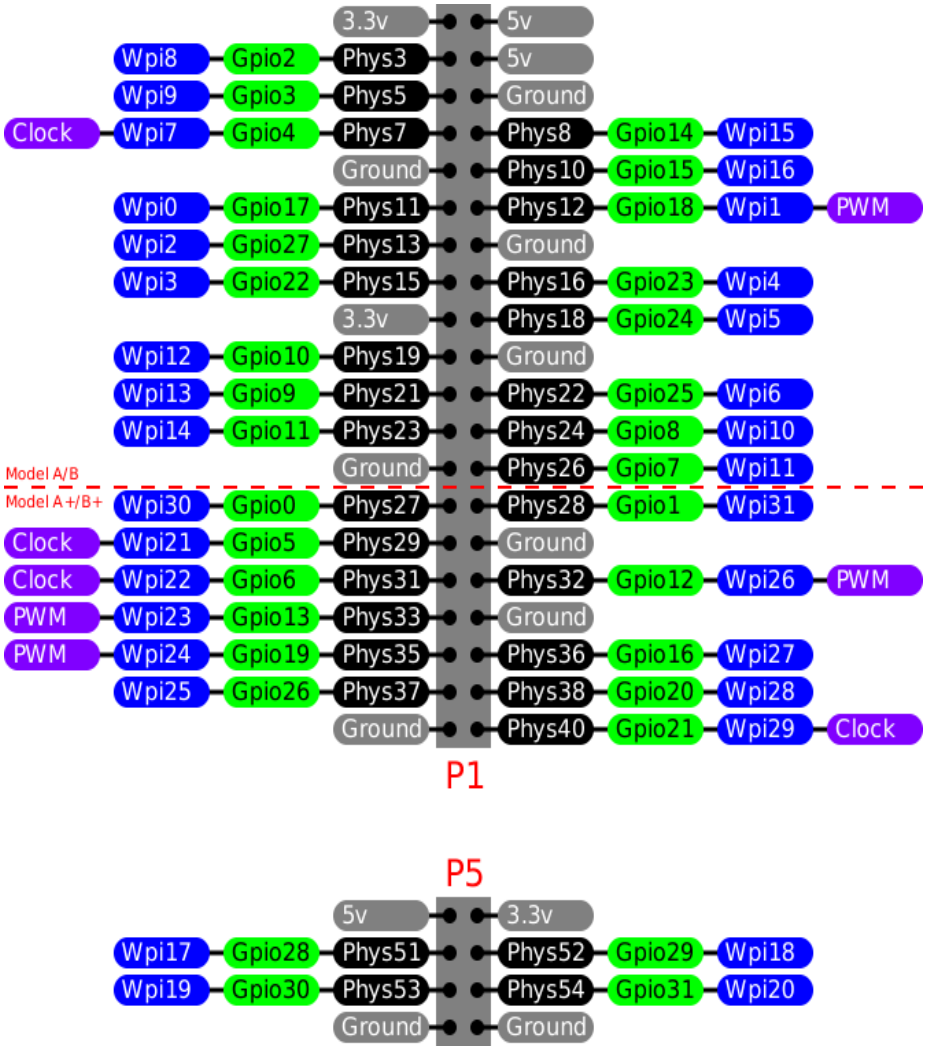


Fig: 4.3.1 – Raspberry Pi Pin Configuration

4.4 – HAAR Cascade Classifier

We know that we can extract features from the image and based on the features we can classify the object. This is like convolutional kernel. Object detection and identification on camera is done using the HAAR Cascade Classifier. An object detection that inputs HAAR features into a series of classifiers i.e., Cascading classifiers to identify objects in an image. We train this model to identify one type of object in animage (vehicle in our case).

Developing a framework for object detection using HAAR feature-based Cascade Classifiers is the main goal of this part of the project. We require OpenCV modules to carryout object detection using the C++ programming language.

- **Positive Sample:** These images include the images that we want our classifierto recognize.
- **Negative Sample:** These images include the images that we want our classifiernot to recognize.

Here is where the model training needs to be done. Therefore, utilizing positive samples and negative samples of roughly 100 to 300 images each, we must create a small dataset. We thenextract features using sliding windows of rectangular blocks. We can see that there are three types of features in HAAR Cascade Classifier.

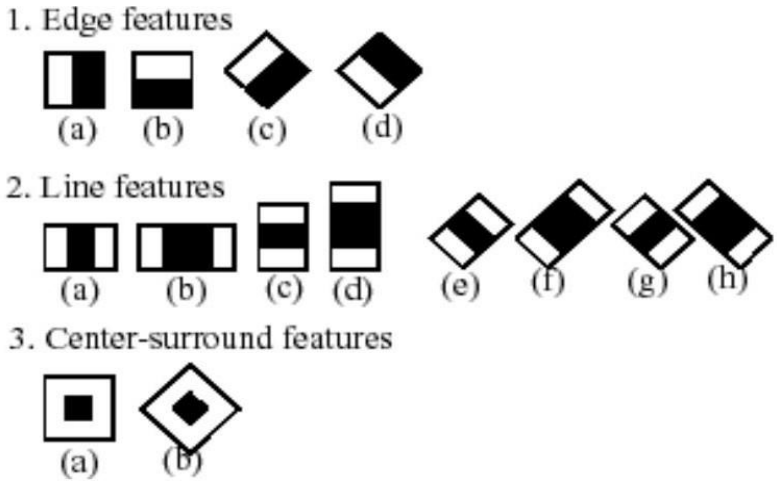


Fig: 4.4.1 – HAAR Features

These values are single-valued and are calculated by subtracting the sum of pixel intensities under white rectangles from the black rectangles. This results in ridiculous number calculations if the base window is 24 x 24 pixels (180,000 features generated). To remove these redundant features, we use Boosting. The figure clearly explains how boosting works in different iterations to remove redundant features.

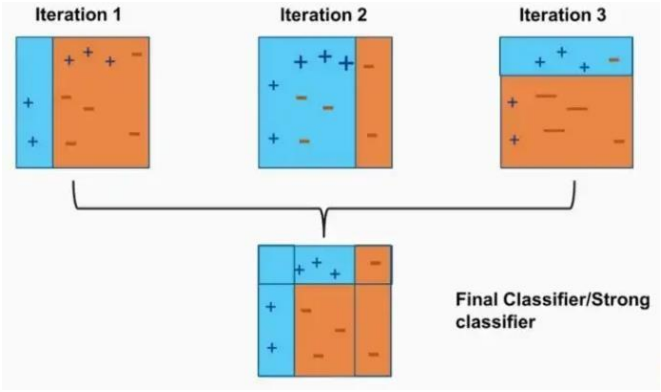


Fig: 4.4.2 - Boosting

Boosting is used to determine the most informative features, with Freund and Schapire's Adaboost being the algorithm of choice due to its ease of implementation.

Fig: 4.4.3 – Relevant & Irrelevant Images

After doing this the calculation of the feature reduces to 6000. This concept is called the Cascade of Classifiers. This is the same method used for detecting face in our mobile phones.This method is called the Viola Jones method.

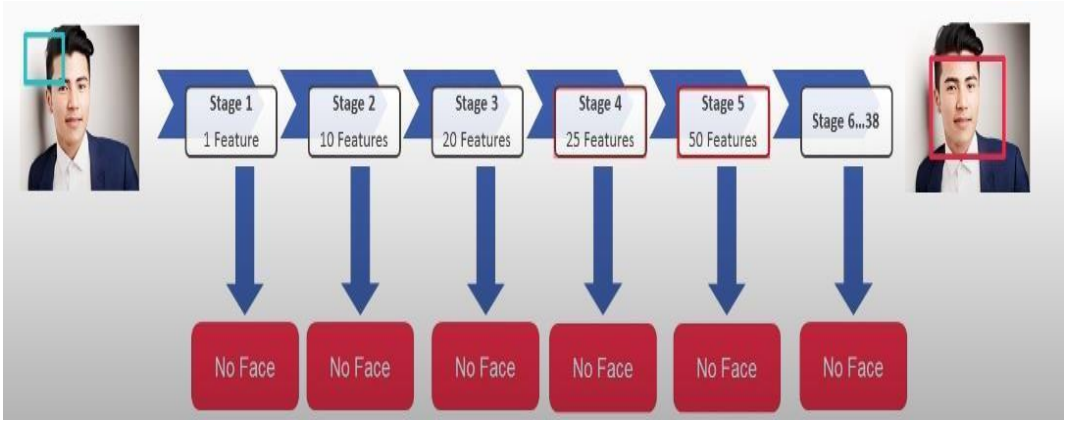


Fig: 4.4.4 – Cascade of Classifiers

4.5 - Object Detection

Detecting objects is one of the main computer vision problems that is now a crucial component of many of the consumer applications which includes security and surveillance systems, mobile text recognition, and the diagnosis of diseases from MRI/CT scans. Object Detection is another essential element for autonomous driving.

To provide safe and reliable driving performance, autonomous cars rely on their assessment of their surroundings. This vision system employs object identification algorithms to precisely identify nearby items, such as pedestrians, automobiles, traffic signs, traffic lightsand barriers.

Firstly, object detection is the ability to detect the individual objects in an image. Object detection is hard because there are different viewing angles of objects in an image, lighting,clarity, and many more.

There are a lot of applications for object detection like labeling scenes, Robot navigation, Self-driving car, Face recognition, and many more.

- HOG – Histogram of Oriented Gradients
- LBP – Linear Binary Pattern
- HAAR Cascade Classifier

We chose HAAR Cascade Classifier over the other two algorithms because when it comes to accuracy HAAR Classifier is better than LBP and HAAR Classifier is faster than HOG aswe are using Raspberry Pi 3 as our primary processor.

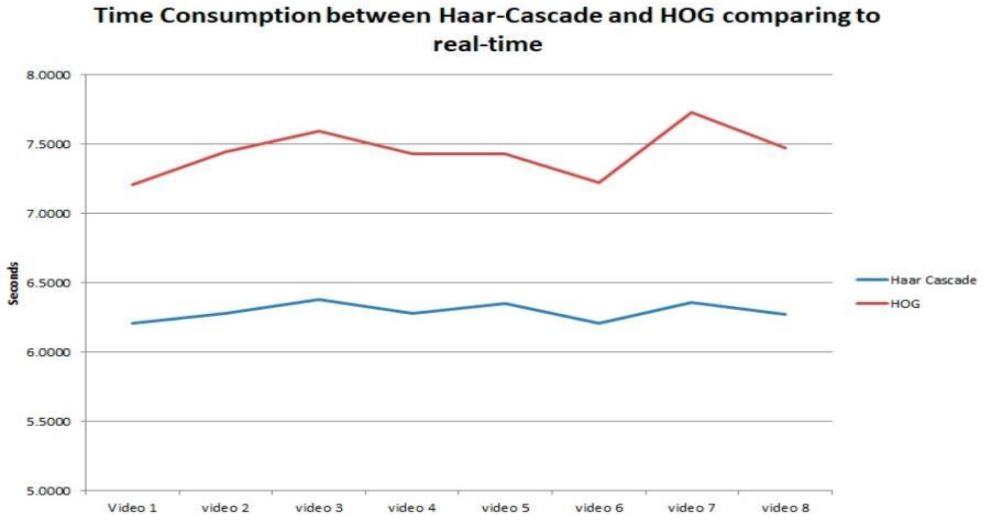


Fig: 4.5.1 – HAAR Cascade vs HOG

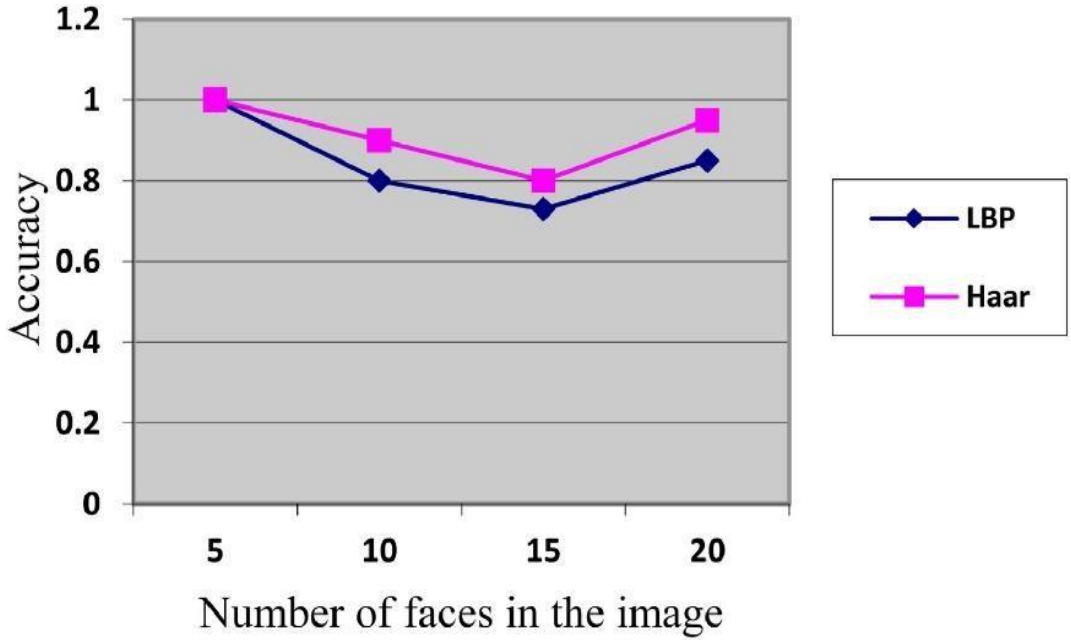


Fig: 4.5.2 – HAAR Cascade vs LBP

Predicting the distance between the Robot and the Obstacle:

After detecting the object, we must predict the distance between the robot and the obstacle to prevent a collision.

We must reduce our region of interest to the stop sign only to increase the accuracy and reduce the noise.

The steps involved in distance estimation are:

- We use this equation,

  Y = m (P1 - P2) + c,
  Where Y is the actual distance
  m is the slope
  c is the y-intercept
  (P1-P2)- width of the region of interest in pixels

- Here we keep the obstacle and robot at some separation to get the slope and the y-intercept.
- After getting the slope and y-intercept we can easily get Y (actual distance).

4.6 - Stop Sign and Traffic Light Detection

Traffic signs play a vital role in directing traffic, disciplining drivers, and thus preventing injuries, property damage, and deaths. Traffic signs can be anything like Stop Sign, Speed Limit sign, No Overtaking sign, and many more. Traffic sign management with automatic detection and recognition is a very important part of all intelligent and smart traffic systems.

Steps for Neural Network Training:

1. Collection of Positive Samples

Positive samples are the images that contain the stop sign (Region of interest) in it. We aregoing to capture the images of the stop sign in all possible directions, which can be the casefor our model.

Fig: 4.6.1 – Positive Samples

2. Collection of Negative Samples:

Negative samples are the images that do not contain the Region of interest in it. We are goingto capture the images of the track surrounding it in all possible directions. The images count should be greater compared to the positive samples. It can be at least twice the count of thepositive samples to train the classifier in a better way.

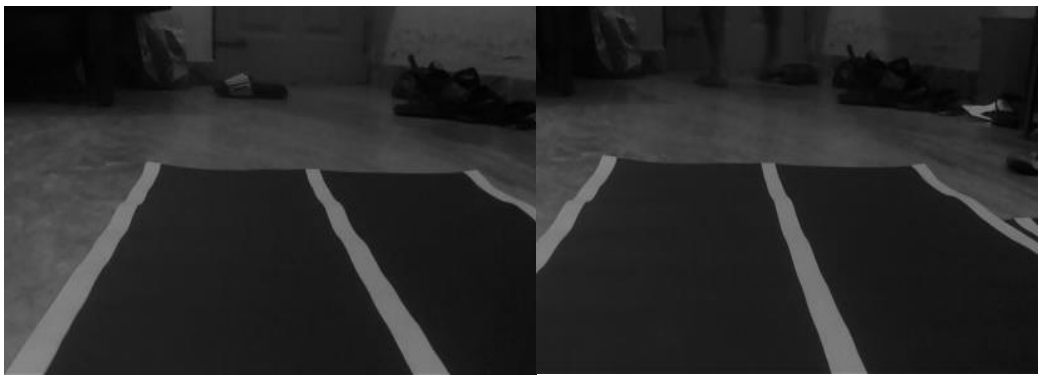


Fig: 4.6.2 – Negative Samples

3. Training of HAAR Cascade Classifier:

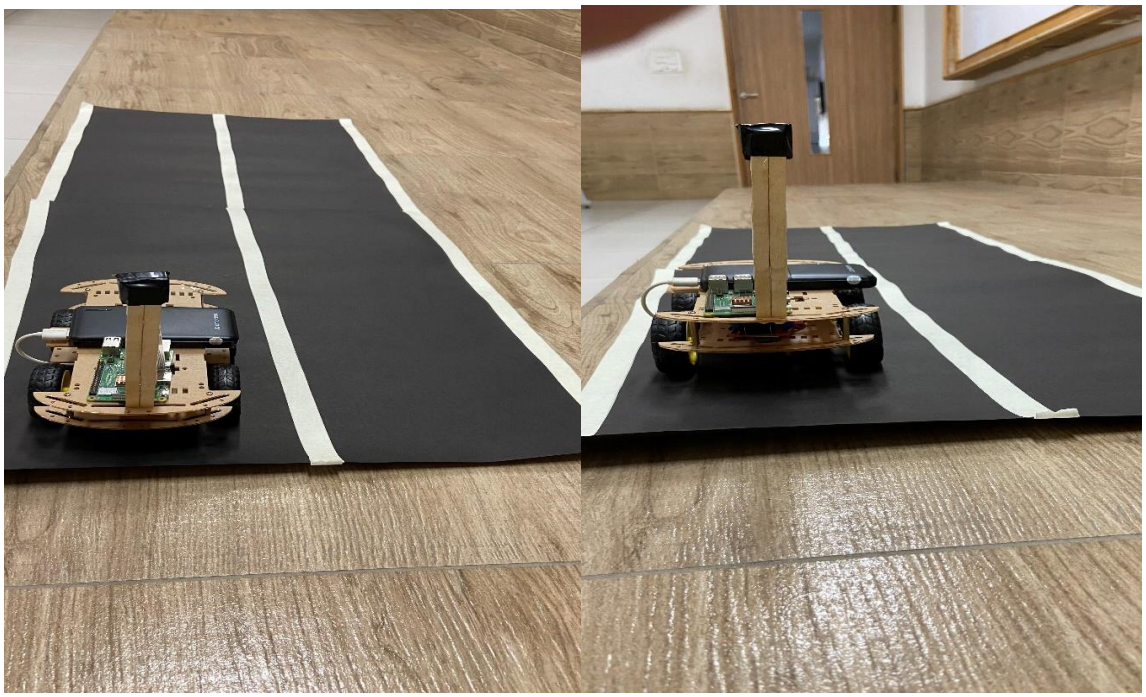
After collection of positive and negative samples, we are now training the classifier to detectthe stop sign. It is done by first highlighting the region of interest in the positive samples andfollowed by training.

4. Detection:

After successfully training the classifier, whenever our prototype detects the stop sign in its path, it will wait for a few seconds (up to 5 to 10 seconds).

5. Results

The protype model with camera mounted, and the track are as shown:



5.1 – Prototype Car Model on Lane

In our project, the model will successfully perform lane detection and highlight the region of interest with a red box. It also shows the bird's eye view of the same. The border of the road is also highlighted with a pair of green lines, using perspective transformation.

The following figures represent the way our robot detects the lane:

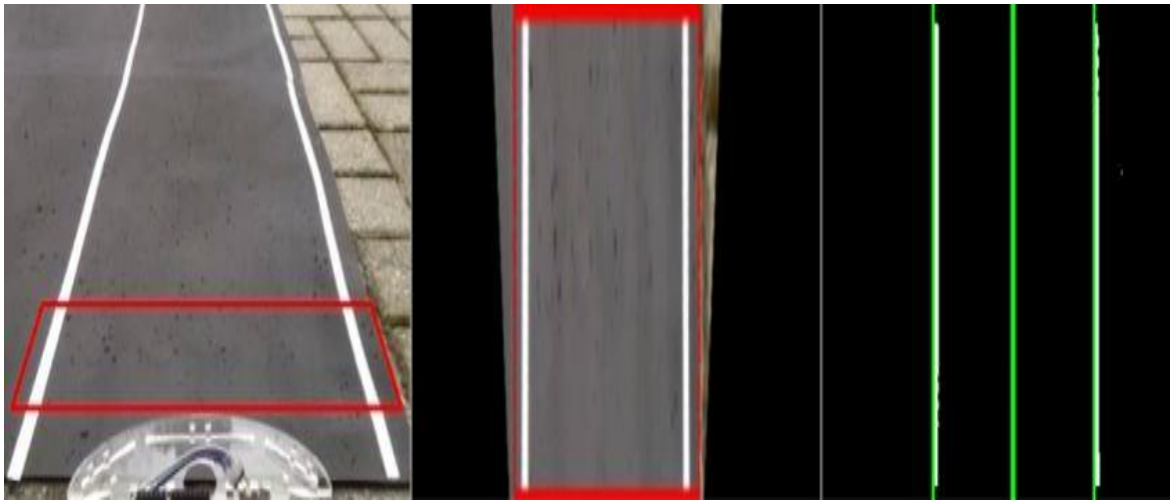


Fig: 5.2 – (a) Region of Interest (b) Perspective Transformation

(c) Green: Left, Right, Centre of the Lane

Blue: Frame Centre

Our model also detects and captures traffic signs such as the “stop” sign and highlights the sign with red borders.

Fig: 5.3 – Detection of Stop Sign

6. Conclusion and Future Scope

Conclusion:

In this project, we used a model of a prototype car to test our algorithm. Our approach was implemented using C++. To manage the processing in the car model, we used a Raspberry Pi. The speed and direction of the motor is being controlled using an Arduino and an H bridge. The prototype car model has forward and reverse motion. It is also capable of turning left andright at various levels. The model can detect the lane successfully and produce the output as expected. The model can also detect the "stop" sign as expected.

Future Scope:

Our approach can be extended to several study areas. We can improve our algorithm to function under more challenging conditions. By optimizing the algorithm, the car model will operate without any hindrance. By updating the hardware, the model can be used as a research platform by upcoming teams for additional development. Furthermore, study may be focused on making algorithms and gaining a better knowledge of the outside world. Our model can be enhanced to change lanes from the original one when it encounters obstacles such as railings, boulders, or other cars.

7. References

1) Nakib Hayat Chowdhury, Deloara Khushi, Md. Mamunur Rashid, "Algorithm for Line Follower Robots to Follow Critical Paths with Minimum Number of Sensors" International Journal of Computers (2020)

2) Hironobu Fujiyoshi, Tsubasa Hirakawa, Takayoshi Yamashita, "Deep learning-based image recognition for autonomous driving" International Association of Traffic and Safety Sciences, 2019.

3) Ahmed El Mahdawy, Amr El Mougy, "Path Planning for autonomous vehicles with dynamic lane mapping and obstacle avoidance" 13th International Conference on Agents and Artificial Intelligence, 2021.

4) Jagannath Aghav, Poorwa Hirve, Mayura Nene, "Deep learning for Real Time Collision Detection and Avoidance" ICCCN (2019)

5) Roshan Jahan, Preetam Suman, Deepak Kumar Singh, "Lane detection using Canny edge detection and Hough transform on Raspberry Pi" ISSN (2019)

6) Maicol Laurenza, Gianlunca Pepe, Dario Antonelli, Antonio Carcaterra, "Car collision avoidance with velocity obstacle approach" University of Rome, 2019

7) Boris Crnokić, Slaven Pehar, "Comparison of Edge Detection Methods for Obstacles Detection in a Mobile Robot Environment" (2018)

8) Shilp Dixit, Umberto Montanaro, Mehrdad Dianati and Saber Fallah, "Trajectory Planning for Autonomous High-Speed Overtaking in Structured Environments Using Robust MPC " IEEE JUNE 2020

Final_year_capstone (2).pdf

ORIGINALITY REPORT

8%	8%	0%	0%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.coursehero.com Internet Source	8%

Exclude quotes	On	Exclude matches	< 5 words
Exclude bibliography	On		

ADVANCED DRIVER ASSISTENCE SYSTEM AND TRACKING DEVICE USING MACHINE LEARNING

Aditya V K, Sadir Irfan, Sanjan I S, Sathvik Prakash

Project Guide: Prof. Karpagavalli S

MOTIVATION

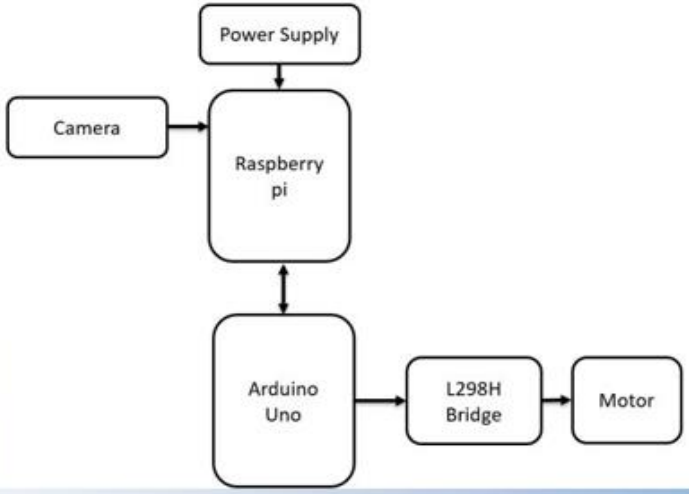
- According to data, 93% of accidents occur due to human error. Hence autonomous driving systems are a step towards significantly reducing that number. Therefore, we want to make a model which will significantly contribute for the development of autonomous vehicle systems.
- Possibility of radically altering transportation and society by enhancing mobility, road safety, freight productivity, and easing traffic congestion.
- Our model will perform path following and stop sign detection

IMPLEMENTATION

- Building hardware of a self-driving car
- Slave device setup
- Master device setup
- Camera Setup for Raspberry Pi
- C++ code to capture Image and Video
- Image processing using OpenCV4 and C++ for lane change detection
- Neural network training and testing for stop sign
- Stop sign detection using camera
- Neural Network training for obstacle
- Obstacle detection using camera
- Stop sign detection and object detection is done using HAAR Cascade Classifier

WORKING

- The model captures the images of the environment and uses image processing to identify the lanes and obstacles.
- Raspberry Pi acts as the main processing unit in the car model.
- Arduino Uno instructs the L298 H Bridge to manage the current flow to the motors.
- Power is supplied to the motors through a power bank.



RESULTS AND CONCLUSIONS

- Lane detection is performed. The region of interest is defined.
- A bird's eye view of the same is produced for precise viewing.
- A pair of green is used to show the border of the road.
- Car model functions best in moderate environments; under adverse circumstances, they may not perform on optimal level.
- The car model can successfully use the algorithm we created and perform in line with our expectations.

Advanced Driver Assistance System and Tracking Device using Machine Learning

Aditya V Kakarambi Sadir Irfan Sanjan I S Sathvik Prakash
Dept. of Electronics and Communication
PES University
Bengaluru, India
adityakakarambi5@gmail.com sadirirfan625@gmail.com sanjanis24@gmail.com sathvikprakash08@gmail.com

Abstract—By improving mobility, road safety, freight productivity, and reducing traffic congestion, autonomous vehicle development has the potential to fundamentally revolutionize both transportation and society. We focused on creating an algorithm that will use machine learning to do path following, obstacle recognition, and collision avoidance. After which, implementing that algorithm in the robot car model. The Raspberry Pi served as the master device in our arrangement, while the Arduino served as the slave. We will send instructions to the slave device, which will use the H Bridge to control the motion of the models' motors, by taking an image with the master device and processing it using an algorithm in the master device. We can demonstrate the robot automobile that will use our algorithm.

Index Terms—Raspberry Pi 3B+, Arduino Uno, Canny Edge Detection, HAAR Cascade Classifiers

I. Introduction

The development of a completely autonomous vehicle is currently a trend and area of focus in the automotive industry. According to GIDAS, human error is to blame for 93.5 of automobile-related accidents. The primary goal of our project is to offer a vehicle that can drive independently, recognize traffic signs like stop, and speed limit, and also follow traffic lights. The primary goal of this initiative is to offer absolution to the growing number of traffic accidents and reduce fuel usage by offering drivers a vehicle that can drive independently, recognize the signal, start, and stop when there is a green, or reutilize the appropriate lighting and the ultrasonic sensor, which also benefits the user of not just observing movement but also possess motion-control capabilities for the vehicle using the remote mode, and instructions.

In order to achieve the above endeavor, we wish to concentrate on designing an algorithm that does so. We will be able to see the algorithm's result by implementing it in a mechanical

platform, which will be useful for upcoming teams working on the continuation of our project. Our main goal is to create an algorithm that will help us drive and then further develop it into a completely autonomous driving system.

II. Block Diagram

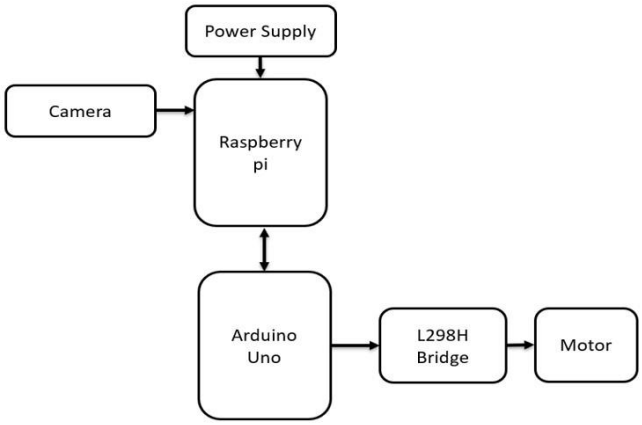


Fig. 1. System Architecture

This is how our prototype works. It first captures the images of the environment and sends them to Raspberry Pi3b+. Raspberry Pi here is responsible for image processing. If the image captured has objects which are needed to be detected like stop signs, obstacles, etc Raspberry pi3B+ detects that object with a red color rectangle around it. This Raspberry based on the detected object it controls the speed of the Motors through giving information to Arduino Uno.

III. Methodology

The main methodology acquired by is that the image captured by the camera is sent to Raspberry Pi for image processing. This image processing is done by HAAR Cascade classifier which is used for traffic sign detection, object detection and traffic light detection. The image catures is also

used for Lane detection which is done using Canny edge Detection algorithm.

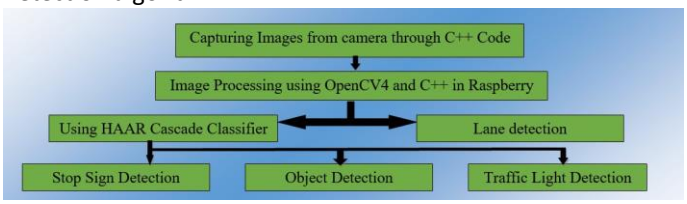


Fig. 2. Methodology

IV. Implementation

A. Hardware Architecture

To implement the algorithm of advanced self-driving assistance, we planned to build a prototype car model. To process real-time information, we need a microcomputer aa s central processor unit. It will process the information taken from the camera using an algorithm. Here we will use a master-slave device setup between the microcomputer and microcontroller to make the circuit edge sensitive which helps to control and enhance the performance of the prototype of the car model.DC motors will be used to rotate the wheels present in the robot car chassis. In order to have freedom of rotation and control the speed of both wheels at the same time, DC motors will be connected to H Bridge. The role of Uno will be to act as a mediator between sensors and the commanding machine. This Uno is instructed by the master device .i.e. Raspberry Pi 3B+ to control the movement of the prototype car model.

The materials used to build a prototype model also make a

difference. Its weight has to be minimal to avoid the stress on motors. The power source used on the model should be able to handle the power requirement required by the prototype car model

B. Master Device Setup

Firstly, we need to have an operating system. Thus, flashed the Raspbian OS on Raspberry Pi. To connect Raspberry Pi to a Personal computer Ethernet and micro-USB cable can be used. Ethernet cable is used in communication with the Personal computer and resource sharing whereas micro-USB is used for power supply.

C. Slave Device Setup

The speed of motors will be controlled by using pulse width modulation. Hence, we used PWM pins. For forward and backward movements of motors, here Arduino Uno is used for controlling the voltage supplied to the motor using logic. To change the direction of motors we will control the speed of motors

V. Lane Detection

Lane detection involves various steps to detect a lane.Firstly, its captures the real-time images of the environment from the camera which is mounted to Raspberry Pi on the robot car. We will set the value for frame width, frame height, brightness, contrast, saturation, gain, and frame per second, thus we can focus on what we needed from the surroundings to increase efficiency. Next we have to create region of interest so the our algorithm works on that specific region. Next we try to change the perspective to birds eye-views. We use 4 points of corresponding points in the frame and outputs a transformation matrix M. Transformation matrix M is used by warpPerspective, which applies the perspective transformation to an image. Warp Perspective applies the perspective transformation to an image by using the transformation matrix M.

Subsequently, using a threshold operation, we convert this RGB image to a grey-scale image and then to a binary image to obtain black-and-white intensities. The edges are then found using the Canny edge detection technique. The final image will then be created by overlapping the images created by the threshold operation using canny edge detection. We will select a place of interest in that image that is a rectangle, and we will then draw thin, rectangular strips there that are one unit wide. the intensities of each strip are then written into a dynamic array, V = [0,0,0,0, . . . , 234, . . ., 245, 0,0,0,0]. The next step is to iterate through the dynamic array to find the positions of the left lane (from the left half of the array to the middle of the array) and the right lane (from the middle of the array to the end). The centre of the lane will then be determined using the equation

Lane center = (Right lane position - Left lane position) / 2 + Left lane position, and we'll aim to calibrate so that it overlaps with the centre of the frame. The difference between frame centre and lane centre will be determined. The car will be kept in the middle of the lane with its assistance.

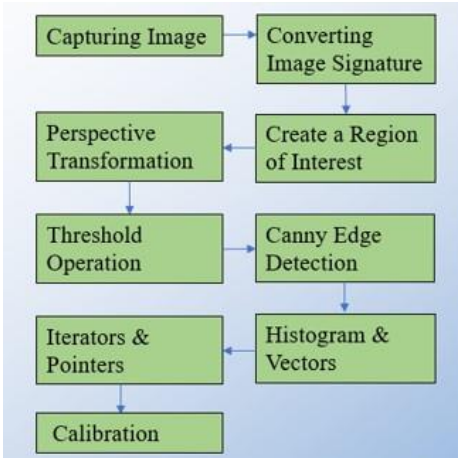


Fig. 3. Steps for Lane Detection

VI. HAAR Cascade Classifier

We know that we can extract features from the image and based on the features we can classify the object. This is similar to convolutional kernel. Object detection and identification on camera is done using the HAAR Cascade Classifier. An object detection that inputs HAAR features into a series of classifiers .i.e. Cascading classifiers to identify objects in an image. We train this model to identify one type of object in an image.Developing a framework for Object detection using HAAR feature-based Cascade Classifiers is the main goal of this part of the project.

- Positive Sample: : These images include the images that we want our classifier to recognize
- Negative Samples: These images include the images that we want our classifier not to recognize

A. Object Detection

Detecting objects is one of the main computer vision problems that is now a crucial component of many of the consumer applications which includes security and surveillance systems, mobile text recognition, and the diagnosis of diseases from MRI/CT scans. Object detection is another essential element for autonomous driving.

Firstly object detection is the ability to detect the individual objects in an image. Object detection is hard because there are different viewing angles of objects in an image, lighting, clarity, and many more.

There are a lot of objects detection algorithms:

- HOG (Histogram of Oriented Gradients)
- LBP (Linear Binary Pattern)
- HAAR Cascade Classifier

We choose HAAR Cascade Classifier over the other two algorithms because when it comes to accuracy HAAR Classifier is better than LBP and HAAR Classifier is faster than HOG as we are using Raspberry pi 3 as our primary processor.

B. Stop Sign and Traffic Light Signal Detection

Traffic signs play a vital role in directing traffic, disciplining drivers, and thus preventing injuries, property damage, and deaths. Traffic signs can be anything like Stop Sign, Speed Limit sign, Traffic lights and many more. Traffic sign management with automatic detection and recognition is a very important part of all intelligent and smart traffic systems. Steps for neural network training:

- Collection of positive samples.
- Collection of negative samples.
- Training using Haar Cascade Classifier.
- Detection of the stop sign and Traffic light .i.e. to stop if traffic light turns red.

VII. Result and Conclusion

A. Result

The figure above represents the way the robot detects the stop signal.

Fig. 4. Detected Stop Sign

In our project the model will successfully perform lane detection and highlight the borders of the road with a pair of green lines using perspective transformation.

Fig. 5. Lane Detection

B. Conclusion

In this project, we used a model of a prototype car to test our algorithm. Our approach was implemented using C++. To manage the processing in the car model, we used a Raspberry Pi. The speed and direction of the motor is being controlled using an Arduino and an H bridge. The prototype car model has forward and reverse motion. It is also capable of turning left and right at various levels. In addition, it could recognize the track's lanes and move between them. The vehicle can identify the objects. The prototype detects lanes as expected.

C. Future Scope

Our approach can be extended to several study areas. We can improve our algorithm to function under more challenging conditions. By optimizing the algorithm, the car model will operate without any hindrance. By updating the hardware, the model can be used as a research platform by upcoming teams for additional development. Furthermore, study may be focused on making algorithms and gaining a better knowledge of the outside world. Our model can be enhanced to change lanes from the original one when it encounters obstacles such as railings, boulders or other cars

Acknowledgment

We would like to take this opportunity to thank Prof. KARPAGAVALLI, Professor, Department of Electronics and Communication Engineering, PES University, for her persistent guidance, suggestions, assistance, and encouragement throughout the development of this project. It was an honor

for me to work on the project under her supervision and understand the importance of the project at various stages.

We are grateful to the project coordinator Prof. RAJASEKAR M, Department of Electronics and Communication Engineering for, organizing, managing, and helping with the entire process.

We would like to thank Dr. ANURADHA M, Professor, and Chairperson, Department of Electronics and Communication Engineering, PES University, for her invaluable support and thankful for the continuous support given by the department. I am also very grateful to all the professors and non-teaching staff of the Department who have directly or indirectly contributed to our research towards enriching work. Chancellor Dr. M R DORESWAMY, Pro-Chancellor Prof. JAWAHAR DORESWAMY, Vice Chancellor Dr. SURYAPRASAD J, the Registrar Dr. K S SRIDHAR and Dean of Engineering Dr. KESHAVAN B K of PES University for giving me this wonderful opportunity to complete this project by providing all the necessaries required to do so. It has motivated us and supported our work thoroughly.

We are grateful and obliged to our family and friend for providing the energy and encouragement when we need them the most.

REFERENCES

[1] Nakib Hayat Chowdhury, Deloara Khushi, Md. Mamunur Rashid, ”Algorithm for Line Follower Robots to Follow Critical Paths with Minimum Number of Sensors” International Journal of Computers (2020)

[2] Hironobu Fujiyoshi, Tsubasa Hirakawa, Takayoshi Yamashita, “Deep learning-based image recognition for autonomous driving” International Association of Traffic and Safety Sciences, 2019.

[3] Ahmed El Mahdawy, Amr El Mougy, “Path Planning for autonomous vehicles with dynamic lane mapping and obstacle avoidance” 13th International Conference on Agents and Artificial Intelligence, 2021.

[4] Jagannath Aghav, Poorwa Hirve, Mayura Nene, “Deep learning for Real Time Collision Detection and Avoidance” ICCCN (2019).

[5] Roshan Jahan, Preetam Suman, Deepak Kumar Singh, ”Lane detection using Canny edge detection and Hough transform on Raspberry Pi” ISSN (2019).

[6] Boris Crnokic, Slaven Pehar, ”Comparison of Edge Detection Methods´ for Obstacles Detection in a Mobile Robot Environment” (2018).

[7] Shilp Dixit, Umberto Montanaro, Mehrdad Dianati and Saber Fallah, ”Trajectory Planning for Autonomous High-Speed Overtaking in Structured Environments Using Robust MPC ” IEEE JUNE 2020.

[8] Maicol Laurenza, Gianlunca Pepe, Dario Antonelli, Antonio Carcaterra, “Car collision avoidance with velocity obstacle approach” University of Rome, 2019.