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

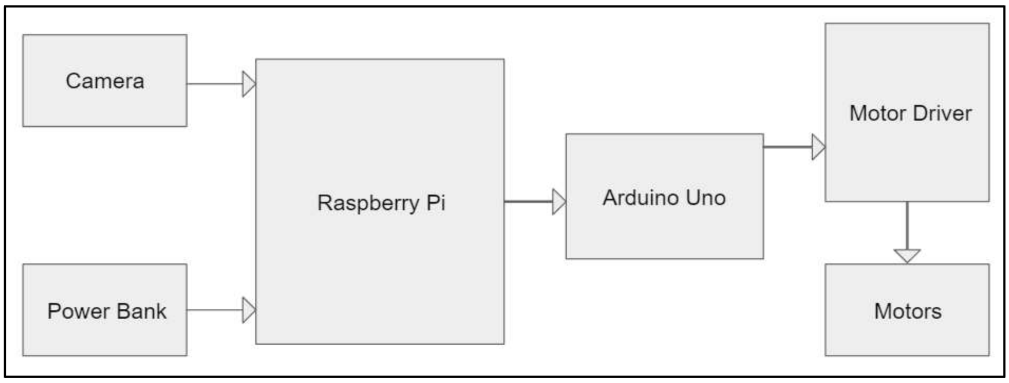


Fig. 3.1 Block Diagram of Self Driving Robot Car

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]

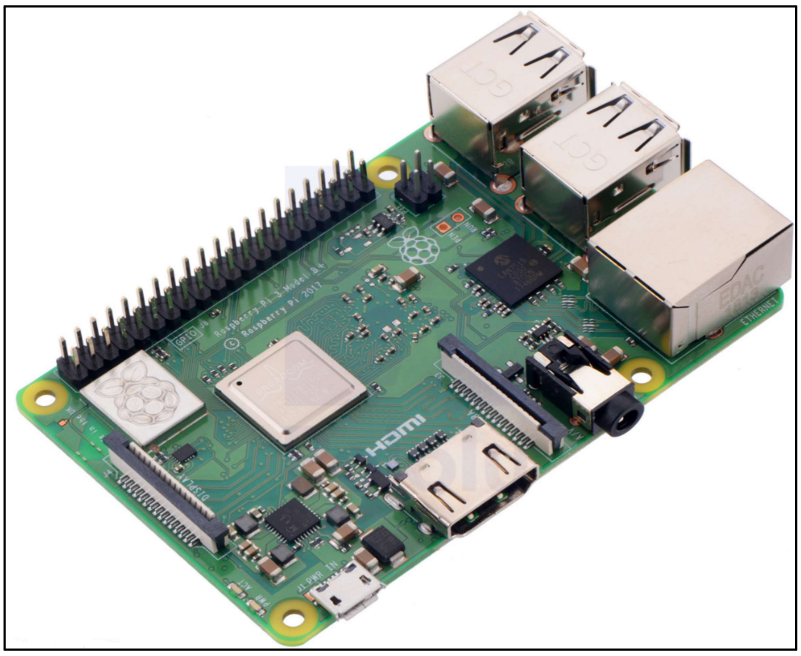


Fig. 3.2 Raspberry Pi Model 3B+

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]

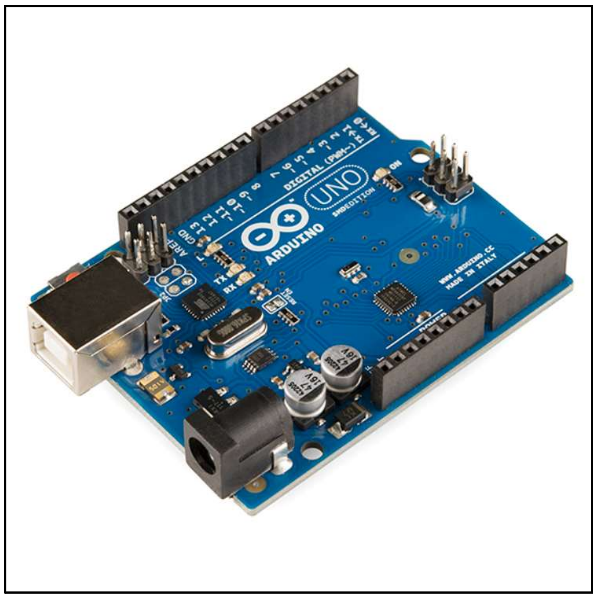


Fig. 3.3 Arduino Uno SMD R3

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

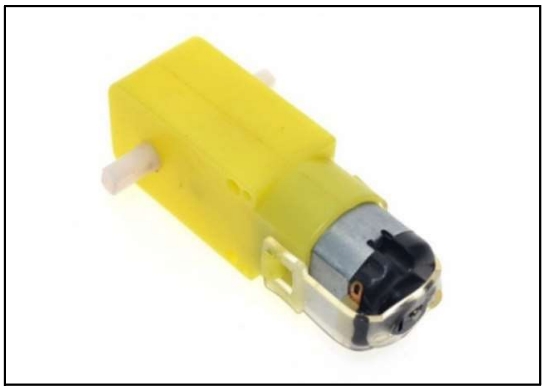


Fig. 3.4 200 RPM Dual shaft BO motor

DC Motor – 200 RPM – 12 volts 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 one can choose the most precise motor driver module from the wide range available in Motor drivers category as per your specific requirements.

Specifications:

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

4) L293D Motor Driver Module

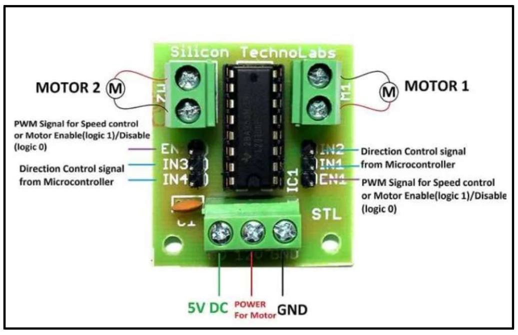

Silicon TechnoLabs
MOTOR 2
M
MOTOR 1
M
M2
M1
PWM Signal for Speed control or Motor Enable(logic 1)/Disable (logic 0)
EN
IN3
IN4
Direction Control signal from Microcontroller
IN2
IN1
EN1
Direction Control signal from Microcontroller
PWM Signal for Speed control or Motor Enable(logic 1)/Disable (logic 0)
IC1
STL
C1
5V DC
POWER For Motor
GND

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

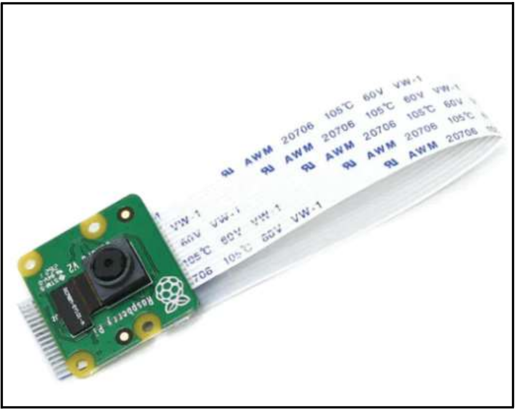


Fig. 3.6 Raspberry Pi Camera Module

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]

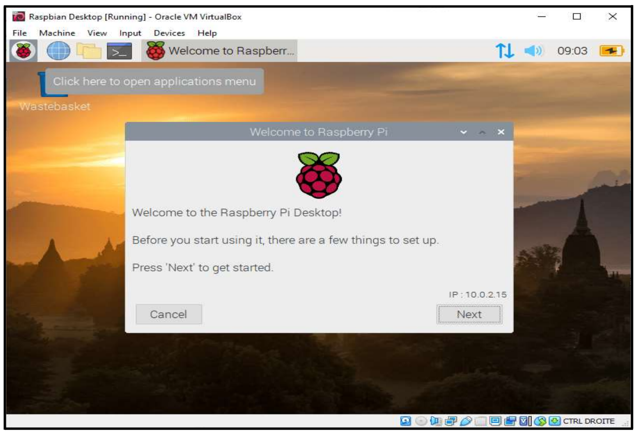
3.4 Working

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.


Raspbian Desktop [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Welcome to Raspberr...
09:03
Click here to open applications menu
Wastebasket
Welcome to Raspberry Pi
IP : 10.0.2.15
Cancel
Next
CTRL DROITE

Welcome to the Raspberry Pi Desktop!

Before you start using it, there are a few things to set up.

Press 'Next' to get started.

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.

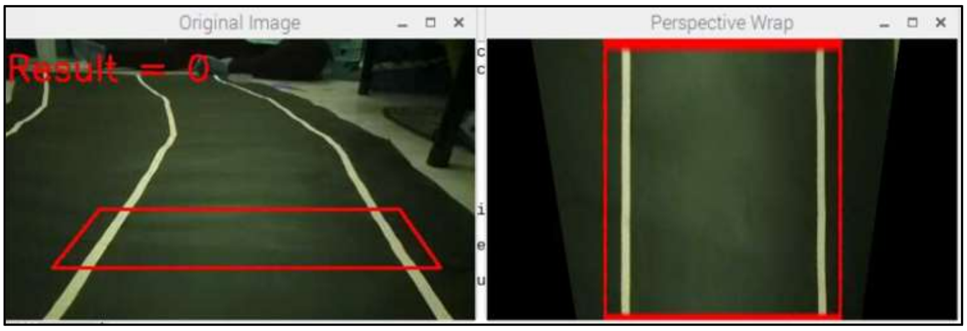


Fig. 3.8 Defining the ROI and Perspective Wrapping

   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.

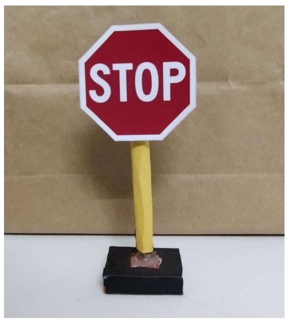


Fig. 3.9 Stop Sign

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

- Around 300 - 350 positive samples of stop signs must be taken from different angles under varying light conditions and 700 - 750 negative images. Negative samples are the samples which do not contain the stop sign. They are the images consisting of surroundings excluding the stop sign.

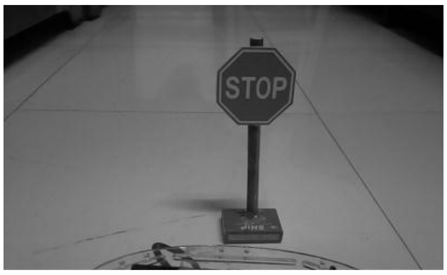


Fig. 3.10 Positive Sample

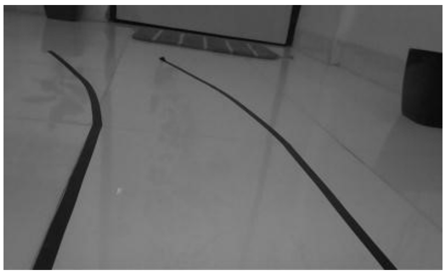


Fig. 3.11 Negative Sample

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.

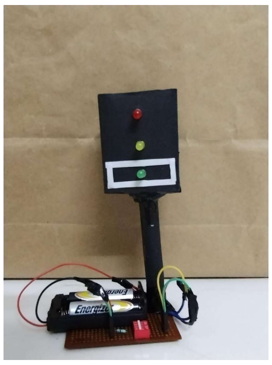


Fig. 3.12 Traffic Light

2) Taking the Positive and the Negative samples for training purposes.
    - Around 300 - 350 positive samples of traffic lights must be taken from different

angles under varying light conditions and 700 - 750 negative images. 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.

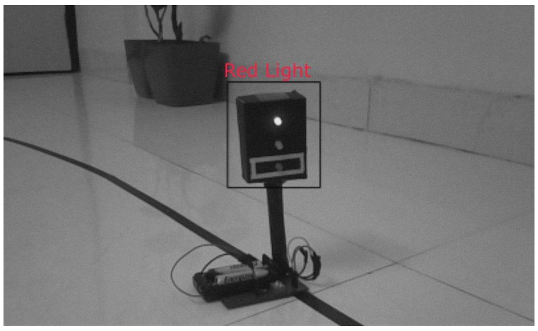


Fig. 3.13 Positive sample

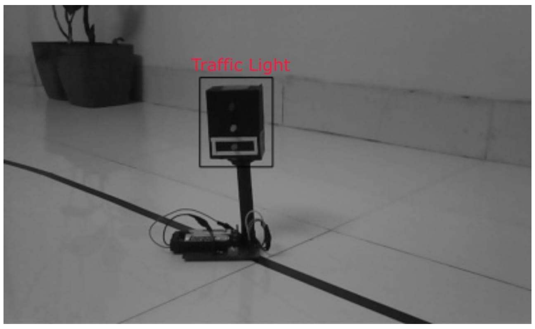


Fig. 3.14 Negative Sample

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.

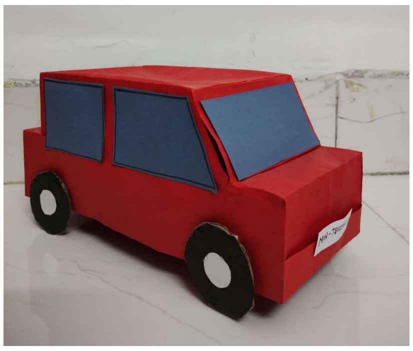


Fig. 3.15 Object (dummy car)

7) Taking the Positive and the Negative samples for training purposes.
    - Around 300 - 350 positive samples of an object must be taken from different angles under varying light conditions and 700 - 750 negative images. Negative samples are the samples which do not contain the object. They are the images consisting of surroundings excluding the object.

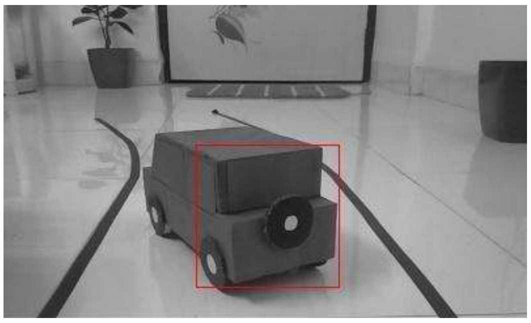


Fig. 3.16 Positive Sample

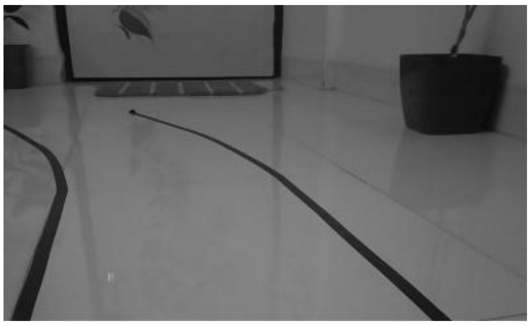


Fig. 3.17 Negative Sample

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.