TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

HIMALAYA COLLEGE OF ENGINEERING

CODE: CT 755

A

FOURTH YEAR MAJOR PROJECT PROPOSAL

ON

# AUTONOMOUS CAR

BY

| | |
|---|---|
| ANIL SHRESTHA | (HCE076BEI002) |
| NISCHAL MAHARJAN | (HCE076BEI007) |
| SANDESH BHUSAL | (HCE076BEI014) |

A MAJOR PROJECT MID TERM PROPOSAL SUBMITTED TO DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR BACHELOR'S DEGREE IN ELECTRONICS COMMUNICATION AND INFORMATION ENGINEERING

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

CHYASAL, LALITPUR, NEPAL

AUGUST, 2023

# AUTONOMOUS CAR

BY

| | |
|---|---|
| ANIL SHRESTHA | (HCE076BEI002) |
| NISCHAL MAHARJAN | (HCE076BEI007) |
| SANDESH BHUSAL | (HCE076BEI014) |

SUPERVISED BY

SIVRAJ LUITEL

A MAJOR PROJECT MID TERM PROPOSAL SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE DEGREE OF BACHELOR'S IN ELECTRONICS COMMUNICATION AND INFORMATION ENGINEERING

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

HIMALAYA COLLEGE OF ENGINEERING

TRIBHUVAN UNIVERSITY

CHYASAL, LALITPUR, NEPAL

AUGUST, 2023

## ABSTRACT

This project aims to design and implement a comprehensive self-driving car system, incorporating advanced algorithms for lane detection, stop sign recognition, traffic light detection, zebra crossing identification, obstacle detection, and overtaking maneuvers. It addresses the pressing need for safer and more efficient transportation through the convergence of artificial intelligence, computer vision, and robotics. By processing sensor data and employing machine learning models, the system makes informed driving decisions while adhering to traffic rules and ensuring passenger safety. The project's scope encompasses real-world scenarios, with extensive testing and validation conducted in both simulated and actual environments. Ultimately, this endeavor contributes to the evolution of autonomous transportation, offering improved road safety, enhanced accessibility, and technological advancements in the fields of robotics and artificial intelligence

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# 1. INTRODUCTION

## 1.1 Background

In recent years, the automotive industry has witnessed a trans-formative shift toward autonomous driving technology, driven by the promise of safer roads, reduced traffic congestion, and enhanced mobility for all. The convergence of artificial intelligence, computer vision, sensor technology, and robotics has paved the way for the development of self-driving cars that can navigate roads with minimal human intervention. This project aims to contribute to this burgeoning field by designing and implementing a self-driving car system with a comprehensive set of advanced features.

## 1.2 Problem Statement

The modern landscape of transportation demands innovative solutions that not only improve efficiency but also prioritize safety and environmental sustainability. Despite advancements in vehicle automation, the transition to fully autonomous vehicles remains a complex challenge. This project addresses the need for a comprehensive self-driving car system with advanced features to overcome current limitations and propel the evolution of autonomous transportation.

## 1.3 Objective

The objectives of the project are:

1. To design and integrate robust algorithms for lane detection, stop sign recognition, traffic light detection, zebra crossing identification, obstacle detection, and overtaking maneuvers.

2. To develop software components and Machine Learning Model that process sensor data, make informed driving decisions, and ensure safe navigation.

3. To implement control algorithms that govern the vehicle's movements while adhering to traffic rules and ensuring passenger safety.

4. To conduct thorough testing and validation in both simulated and real-world environments to assess the system's performance and safety.

## 1.4 Scope

The scopes of the project are:

1. The project encompasses the design, development, and implementation of a comprehensive self-driving car system with a range of advanced functionalities.

2. The project focus on addressing key challenges in autonomous transportation while showcasing the capabilities of the self-driving car in real-world scenarios.

## 1.5 Application

The applications of the project are:

1. Reduced Accidents: By accurately interpreting road signs, signals, and lane markings, the self-driving car will significantly reduce the occurrence of accidents caused by human error.

2. Enhanced Accessibility: The self-driving car system will provide accessible transportation options for individuals who are unable to drive due to age, disability, or other factors.

3. Technological Advancement: The project will contribute to the advancement of self-driving technology, fostering research and innovation in the fields of robotics, computer vision, and artificial intelligence.

# 2. LITERATURE REVIEW

An autonomous car is a self-driving vehicle that has the capability to perceive the surrounding environment and navigate itself without human intervention. For autonomous driving, complex autonomous driving algorithms, including perception, localization, planning, and control, are required with many heterogeneous sensors, actuators, and computers. To manage the complexity of the driving algorithms and the heterogeneity of the system components, this paper applies distributed system architecture to the autonomous driving system, and proposes a development process and a system platform for the distributed system of an autonomous car. The development process provides the guidelines to design and develop the distributed system of an autonomous vehicle. For the heterogeneous computing system of the distributed system, a system platform is presented, which provides a common development environment by minimizing the dependence between the software and the computing hardware. A time-triggered network protocol, FlexRay, is applied as the main network of the software platform to improve the network bandwidth, fault tolerance, and system performance. Part II of this paper will provide the evaluation of the development process and system platform by using an autonomous car, which has the ability to drive in an urban area.[2] Throughout the last century, the automobile industry achieved remarkable milestones in manufacturing reliable, safe, and affordable vehicles. Because of significant recent advances in computation and communication technologies, autonomous cars are becoming a reality. Already autonomous car prototype models have covered millions of miles in test driving. Leading technical companies and car manufacturers have invested a staggering amount of resources in autonomous car technology, as they prepare for autonomous cars' full commercialization in the coming years. However, to achieve this goal, several technical and nontechnical issues remain: software complexity, real-time data analytics, and testing and verification are among the greater technical challenges; and consumer stimulation, insurance management, and ethical/moral concerns rank high among the nontechnical issues. Tackling these challenges requires thoughtful solutions that satisfy consumers, industry, and governmental requirements, regulations, and policies.[1] Part I of this paper proposed a development process and a system platform for the development of autonomous cars based on a distributed system architecture. The proposed development methodology enabled the design and development of an autonomous car with benefits such as a reduction in computational complexity, fault-tolerant characteristics, and system modularity. In this paper (Part II), a case study of the proposed development methodology is addressed by showing the implementation process of an autonomous driving system. In order to describe the implementation process intuitively, core autonomous driving algorithms (localization, perception, planning, vehicle control, and system management) are briefly introduced and applied to the implementation of an autonomous driving system. We are able to examine the advantages of a distributed system architecture and the proposed development process by conducting a case study on the autonomous system implementation. The validity of the proposed methodology is proved through the autonomous car A1 that won the 2012 Autonomous Vehicle Competition in Korea with all missions completed.[3] Autonomous cars are the future smart cars anticipated to be driver less, efficient and crash avoiding ideal urban car of the future. To reach this goal automakers have started working in this area to realized the potential and solve the challenges currently in this area to reach the expected outcome. In this regard the first challenge would be to customize and imbibe existing technology in conventional vehicle to translate them to a near expected autonomous car. This transition of conventional vehicles into an autonomous vehicle by adopting and implementing different upcoming technologies is discussed in this paper. This includes the objectives of autonomous vehicles and their implementation difficulties. The paper also touches upon the existing standards for the same and compares the introduction of autonomous vehicles in Indian market in comparison to other markets. [3] Throughout the last century, the automobile industry achieved remarkable milestones in manufacturing reliable, safe, and affordable vehicles. Because of significant recent advances in computation and communication technologies, autonomous cars are becoming a reality. Already autonomous car prototype models have covered millions of miles in test driving. Leading technical companies and car manufacturers have invested a staggering amount of resources in autonomous car technology, as they prepare for autonomous cars' full commercialization in the coming years. However, to achieve this goal, several technical and nontechnical issues remain: software complexity, real-time data analytics, and testing and verification are among the greater technical challenges; and consumer stimulation, insurance management, and ethical/moral concerns rank high among the nontechnical issues. Tackling these challenges requires thoughtful solutions that satisfy consumers, industry, and governmental requirements, regulations, and policies. Thus, here we present a comprehensive review of state-of-the-art results for autonomous car technology. We discuss current issues that hinder autonomous cars' development and deployment on a large scale. We also highlight autonomous car applications that will benefit consumers and many other sectors. Finally, to enable cost-effective, safe, and efficient autonomous cars, we discuss several challenges that must be addressed (and provide helpful suggestions for adoption) by designers, implementers, policymakers, regulatory organizations, and car manufacturers.[4]

# 3. SYSTEM REQUIREMENTS

## 3.1 FUNCTIONAL REQUIREMENTS

The functional requirements of the project are:

1. Lane Detection and Balancing: The car should able to detect the lane and balance the car to run inside the lane, take turns and U turns.

2. Perception and Object Detection: The car should be able to accurately detect and identify the relevant objects and take evasive actions to avoid collision.

3. Decision Making and Control: The autonomous system should make real-time decision based on camera data obeying traffic rules including traffic lights and other stationary signaling objects.

## 3.2 NON-FUNCTIONAL REQUIREMENTS

The non-functional requirements of the project are:

1. Safety: The system must meet or exceed safety standards for autonomous vehicles, minimizing the risk of accidents and hazards.

2. Reliability: The autonomous system should consistently perform its functions accurately and reliably, minimizing errors and failures.

3. Scalability: The architecture should be designed to accommodate future hardware and software upgrades, allowing for improved performance and capabilities.

4. Performance: The system should be able to process sensor data, make decisions, and control the vehicle in real-time with minimal latency.

## 3.3 HARDWARE REQUIREMENTS

The hardware requirements of the project are:

1. Raspberry Pi 3 B+ (with heat sink)(1)

2. Camera Sensor Module (1)

3. Arduino UNO (1)
4. Bo Motors (4)
5. Battery Pack 12V (1)
6. Jumper Wires
7. ON/OFF Switch (Min: 12V, 1.5 Amp)

# 4. METHODOLOGY

## 4.1 Block Diagram of the System

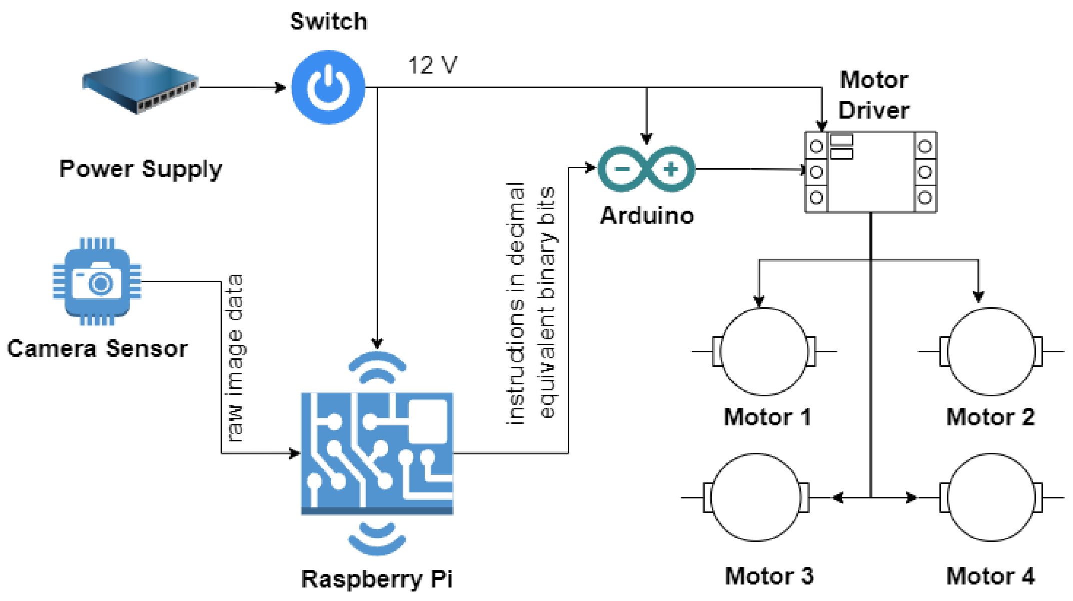

Figure 4.1: Block diagram of the system

The brief explanation of components of block diagram:

### 4.1.1 Power Supply

The power supply distributes stable 12 Volt supply to all the components like Raspberry Pi Arduino and Motor Driver.

### 4.1.2 Camera Sensor

The Camera Sensor captures the raw images of the field view and transfer to the Raspberry Pi to for processing of the images.

### 4.1.3 Raspberry Pi

The Raspberry Pi is the central component of the system that handles the task like interfacing Camera Sensor and Arduino, storing images, running Image Processing Algorithm and Machine Learning Model, sending signals to the Arduino for controlling the car.

### 4.1.4 Arduino

The Arduino is interfaced with Raspberry Pi and Motor Driver. It receives signal form the Raspberry Pi and transmits to the Motor Driver for Controlling the car and also handles the tasks of powering side lights.

## 4.2 Data Collection

The camera sensor plays a pivotal role in gathering data for both image processing and machine learning purposes. This data encompasses a diverse array of images capturing lanes from various angles and under differing lighting conditions, as well as images of traffic lights, stop signals, and the presence of different objects on the road. Upon establishing the calibrated resolution and an optimal frame rate, the acquired data is carefully processed by cropping out regions pertinent to obstacle, traffic light, and stop signals. Subsequently, this curated data is collected and stored for further analysis and utilization.

## 4.3 Flow Chart and Algorithm of Lane Detection

Flow Chart of Lane Detection:

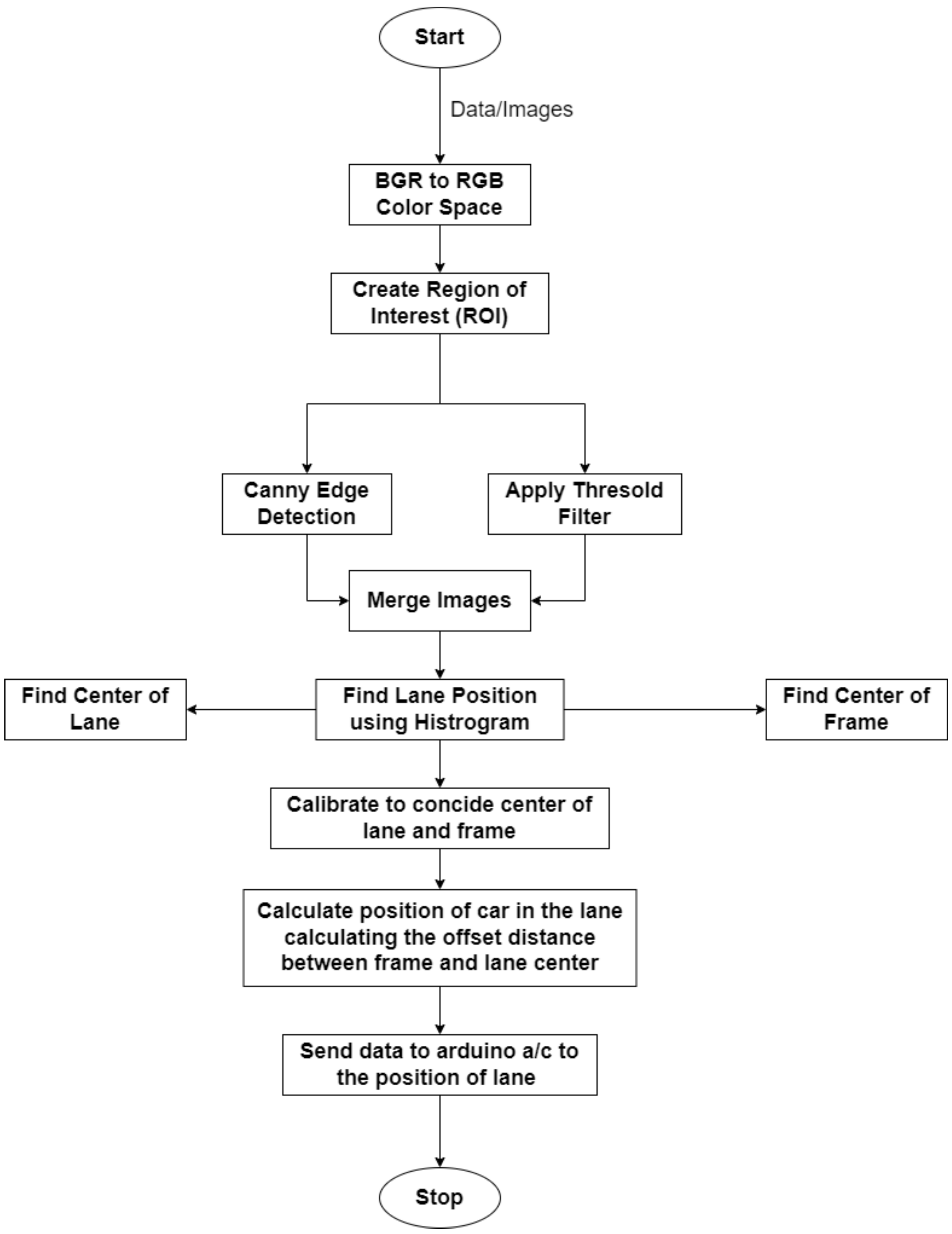
Start
Data/Images
BGR to RGB Color Space
Create Region of Interest (ROI)
Canny Edge Detection
Apply Thresold Filter
Merge Images
Find Center of Lane
Find Lane Position using Histrogram
Find Center of Frame
Calibrate to concide center of lane and frame
Calculate position of car in the lane calculating the offset distance between frame and lane center
Send data to arduino a/c to the position of lane
Stop

Figure 4.2: Flow chart of Lane Detection

The algorithm of Lane Detection:

1. BGR Color Space to RGB Color Space:
   In computer vision, images are usually represented in different color spaces. The BGR color space is commonly used in OpenCV, while RGB is more

familiar to humans. Converting from BGR to RGB ensures that the color channels are in the order that humans are accustomed to (Red, Green, Blue).

2. Create ROI (Region of Interest):
   A region of interest (ROI) defines a specific area within the image where lane markings are expected to appear. By defining four points in the original image, we create a quadrilateral region. A perspective transformation is applied to warp this quadrilateral into a rectangle, creating a bird's eye view. This transformation is crucial for accurately detecting lane curvature and making it easier to process lanes as straight lines.

3. RGB Color to Gray Scale Color Transformation:
   Converting the RGB frame to grayscale simplifies further processing and reduces computational load. Grayscale images contain only intensity information, making edge detection and thresholding easier.

4. Apply Threshold Filter:
   A thresholding filter converts the grayscale image into a binary image, where pixels are either black (0) or white (255) based on intensity. Applying this filter helps us isolate the lane markings from the rest of the image and reduces the complexity of subsequent steps.

5. Canny Edge Detection to Gray Image:
   The Canny edge detection algorithm identifies rapid changes in intensity (edges) in the grayscale image. This step highlights potential lane edges, making it easier to detect lane markings.

6. Merge The Frames/Images:
   Merging the binary threshold image and the edge-detected image combines their features to create a single image that retains the useful information from both.

7. Find Lane Position Using Histogram:
   A histogram is a graphical representation of the distribution of pixel values in an image. Summing up pixel values along the horizontal axis of the binary threshold image generates a histogram. Peaks in this histogram correspond to the lane markings, helping us find the position of the lanes

8. Find Center of Lane:
   By identifying the peaks in the histogram, we locate the positions of the lane

markings. The midpoint between these peaks is a rough estimate of the center of the lane.

9. Find Center of Frame:
   The center of the frame can be calculated by dividing the width of the image by 2. This center point serves as a reference for us to compare with the lane center.

10. Calibrate Lane Center and Frame Center:
    The offset between the lane center and frame center indicates how much the car is deviating from the lane center. This calibration step helps us quantify the deviation and guides corrective action

11. Calculate Distance and Send Data to Arduino:
    The calculated offset is used to determine the lateral distance of the car from the lane center. This distance information can be sent to an Arduino or micro-controller to adjust the car's steering and keep it within the lane

## 4.4 Flow Chart and Algorithm of Stop Sign Detection

Flow Chart of Stop Sign Detection:

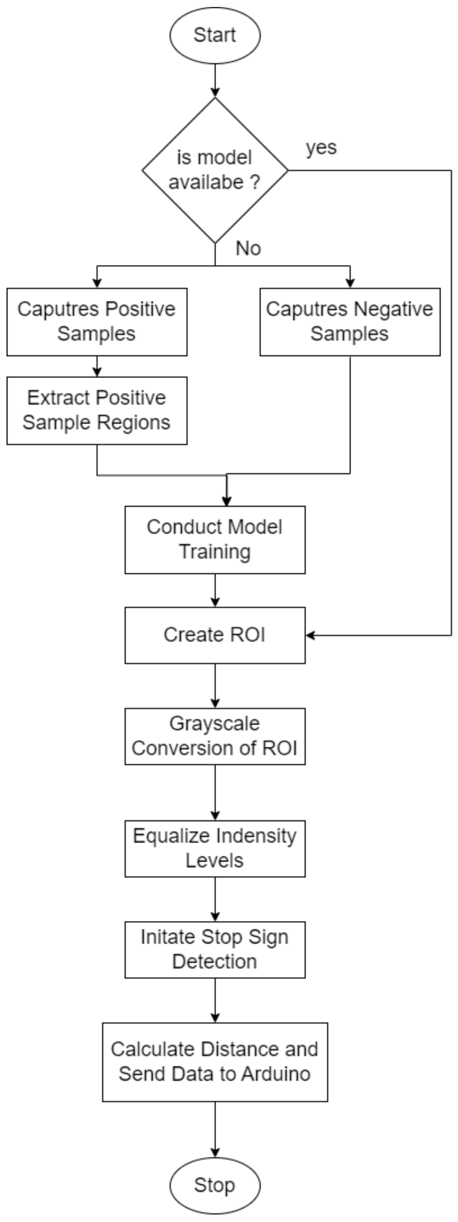
Start
is model availabe ?
yes
No
Caputres Positive Samples
Caputres Negative Samples
Extract Positive Sample Regions
Conduct Model Training
Create ROI
Grayscale Conversion of ROI
Equalize Indensity Levels
Initate Stop Sign Detection
Calculate Distance and Send Data to Arduino
Stop

Figure 4.3: Flow chart of Stop Sign Detection

The algorithm of Stop Sign Detection:

1. Capture Positive Samples:
   A diverse array of positive samples capturing the stop sign from myriad perspectives, orientations, and lighting conditions is acquired. This diversity enriches the training dataset, enabling the model to generalize effectively across varying scenarios.

2. Capture Negative Samples:
   Images devoid of any stop sign presence, encompassing diverse backgrounds and environmental elements, are procured as negative samples. These samples contribute to training the model to differentiate between the presence and absence of a stop sign.

3. Extract Positive Sample Regions: Positive samples are meticulously dissected to isolate regions explicitly featuring the stop sign emblem. By focusing exclusively on the emblematic visual content, the training process is geared towards honing the model's capacity to identify stop signs.

4. Conduct Model Training: Cascade Trainer GUI employs the cascade training algorithm, a variant of the Adaboost algorithm, optimizing the model for stop sign detection through multiple stages of classifier refinement. Using the cascade trainer GUI, the model is trained iteratively. The software ascertains optimal parameters through a sequence of stages, progressively refining the model's discriminatory capabilities by identifying features that distinguish stop signs from non-sign objects.

5. Attain Trained Model: Upon training convergence, the machine learning model, having incorporated the refined classifiers, is encapsulated within an .xml file. This formatted model is stored within the designated classifier repository.

6. Define Region of Interest (ROI): The Region of Interest (ROI), which constitutes the entirety of the captured frame, is delineated. This spatial subset is subjected to subsequent analysis for potential stop sign occurrences.

7. Grayscale Conversion of ROI: Conversion of the ROI into grayscale serves to simplify computational complexity and eliminates the chromaticity-based processing, facilitating the model's ability to focus on intensity patterns.

8. Equalize Intensity Levels: Intensity equalization, a pivotal step, enhances the homogeneity of intensity distribution across the grayscale image. This preprocessing technique heightens contrast and delineates subtle features for subsequent detection.

9. Initiate Stop Sign Detection: By invoking the cascade classifier class's core method, detectMultiScale, stop sign instances are identified within the grayscale image. This method leverages the multi-stage cascade classifier to detect potential stop sign patterns through the analysis of image features.

10. Calculate Distance and Send Data to Arduino:
    The distance is calculated by calibrating the real distance of the car form the object and the number of pixel acquired by the object. This distance information is sent to an Arduino or micro-controller to stop or run the car.

## 4.5 Flow Chart and Algorithm of Zebra Cross Detection

The flow chart of zebra cross detection:

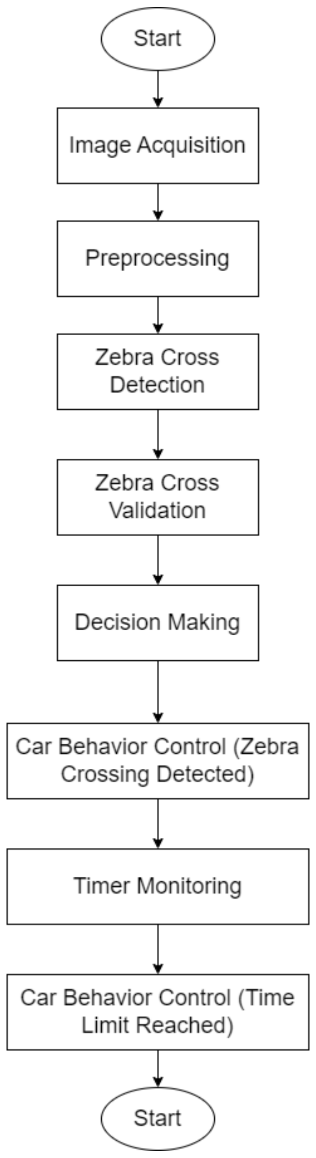

Figure 4.4: Flow chart of Zebra Cross Detection

The algorithm for zebra cross detection:

1. Image Acquisition: Capture live video frames from a camera mounted on the car.

2. Preprocessing:

   Convert the frames to an appropriate color space (such as RGB or HSV). Apply noise reduction techniques, such as Gaussian blurring or median filtering.

3. Zebra Crossing Detection:

   Use image segmentation or edge detection techniques (e.g., Canny) to identify potential edges and features in the frame. Apply a zebra-crossing-specific filter or algorithm to locate zebra-crossing-like patterns based on shape and color cues.

4. Zebra Crossing Validation:

   Evaluate the detected patterns against known zebra crossing templates or shapes. Implement additional criteria like size, position, and orientation to validate detected patterns as actual zebra crossings.

5. Decision Making:

   If a zebra crossing is detected, trigger the car's behavior control logic. If not detected, proceed with normal driving behavior.

6. Car Behavior Control (Zebra Crossing Detected):

   Stop the car at a safe distance before the zebra crossing. Start a timer to count the duration the car remains stationary (e.g., for 1 minute). Activate hazard lights to indicate the car's stationary status.

7. Timer Monitoring:

   Continuously monitor the timer while the car is stopped. If the timer exceeds the defined duration (1 minute in this case), proceed to the next step.

8. Car Behavior Control (Time Limit Reached):

   After the specified time limit is reached, deactivate hazard lights. Resume car movement with appropriate acceleration and speed.

## 4.6 Flow Chart for Lane Switching

The flow chart of zebra Lane Switching:

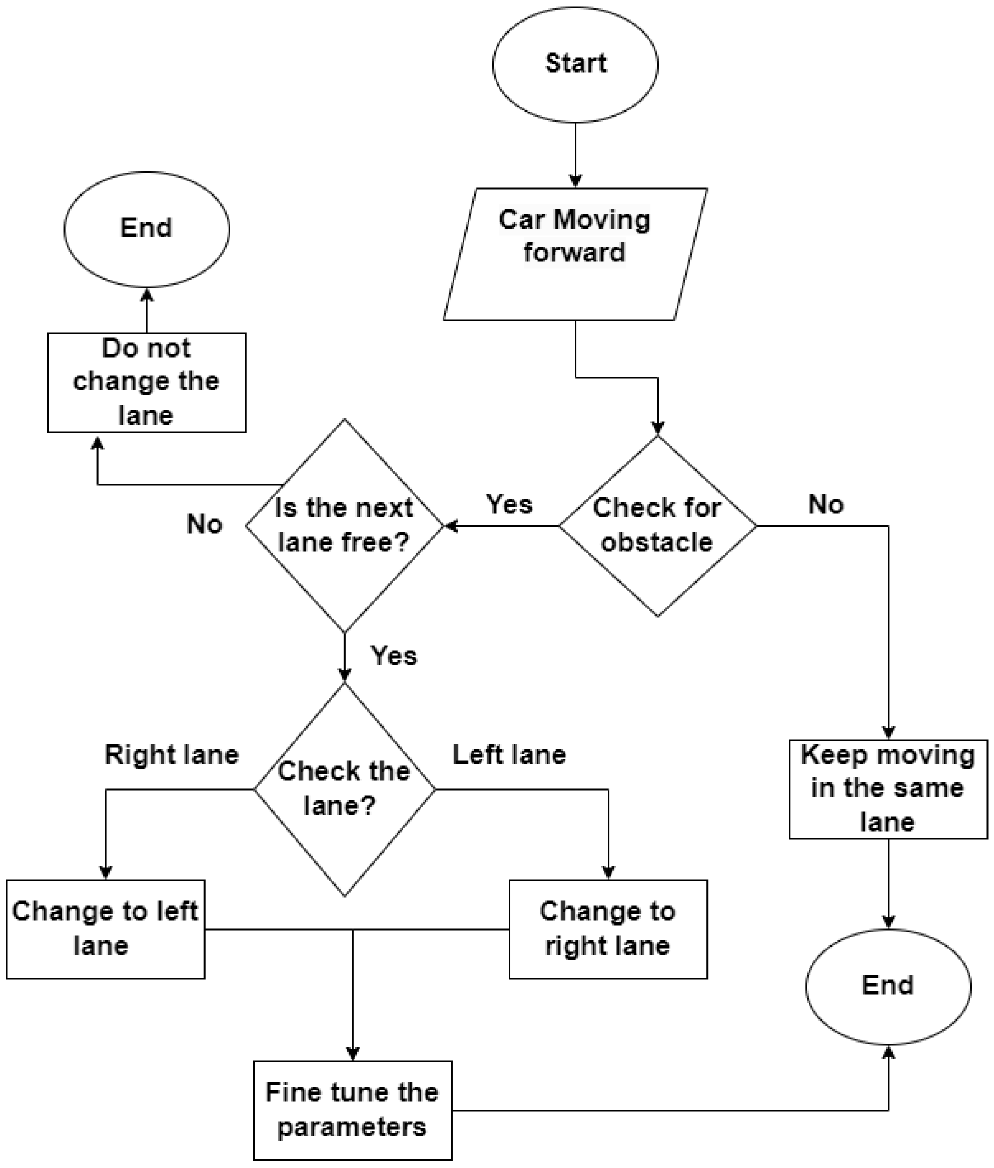

Figure 4.5: Flow chart of Lane Switching

# 5. COST ESTIMATION

The cost estimation of the project:

| S.N | Components | Amount (Rs) | Remarks |
|---|---|---|---|
| 1 | Raspberry Pi 3 B+ | 9500 | With Heat Sink |
| 2 | Camera Sensor Module | 1500 | |
| 3 | Arduino UNO | 1600 | |
| 4 | Bo Motors(4) | 600 | |
| 5 | Battery Pack 12 V | 1000 | |
| 6 | Jumper Wires | 300 | F-F / M-F / M-M |
| 7 | ON/OFF Switch | 100 | |
| 8 | Exhaust Fan | 300 | 12 V |
| | Total | 14,900 | Min: 12V, 1.5 Amp |

Table 5.1: Cost estimation of the project

## 6. EXPECTED OUTPUT

An autonomous car with the ability to stop when it detects obstacle, change lanes,make U-turn, detect traffic lights, and recognize stop signs would operate by smoothly transitioning between lanes, using sensors to monitor neighboring vehicles and signaling appropriately. It would identify the status of traffic lights, stopping at red lights and proceeding cautiously on green lights, while also accounting for pedestrians and other vehicles. Additionally, the car would recognize stop signs, coming to a complete stop at intersections and resuming movement when safe. Safety and rules would be prioritized, with the system prepared for potential errors and effectively communicating intentions to human drivers and pedestrians for a secure and efficient driving experience.

## REFERENCES

[1] Rasheed Hussain and Sherali Zeadally. Autonomous cars: Research results, issues, and future challenges. *IEEE Communications Surveys Tutorials*, 21(2):1275–1313, 2019.

[2] Kichun Jo, Junsoo Kim, Dongchul Kim, Chulhoon Jang, and Myoungho Sunwoo. Development of autonomous car—part i: Distributed system architecture and development process. *IEEE Transactions on Industrial Electronics*, 61(12):7131–7140, 2014.

[3] Kichun Jo, Junsoo Kim, Dongchul Kim, Chulhoon Jang, and Myoungho Sunwoo. Development of autonomous car—part ii: A case study on the implementation of an autonomous driving system based on distributed architecture. *IEEE Transactions on Industrial Electronics*, 62(8):5119–5132, 2015.

[4] M V Rajasekhar and Anil Kumar Jaswal. Autonomous vehicles: The future of automobiles. In *2015 IEEE International Transportation Electrification Conference (ITEC)*, pages 1–6, 2015.