A Project Report on

AI based Self Driving Car

Submitted in partial fulfillment of the requirements for the award
of the degree of

Bachelor of Engineering

in

Information Technology

by

Mahek Jain (17204010)
Vinit Shah (17204014)
Hiral Thadeshwar (17204012)

Under the Guidance of

Ms. Rujata Chaudhari
Mr. Vishal Badgujar

Department of Information Technology
A.P. Shah Institute of Technology
G.B.Road,Kasarvadavli, Thane(W), Mumbai-400615
UNIVERSITY OF MUMBAI
2019-2020

Approval Sheet

This Project Report entitled ***"AI based Self Driving Car"*** Submitted by ***"Mahek Jain"(17204010),"Vinit Shah"(17204014),"Hiral Thadeshwar"(17204012)*** is approved for the partial fulfillment of the requirenment for the award of the degree of ***Bachelor of Engineering*** in ***Information Technology*** from ***University of Mumbai***.

(Mr. Vishal Badgujar)
Co-Guide

(Ms. Rujata Chaudhari)
Guide

Mr. Kiran Deshpande
Head Department of Information Technology

Place:A.P.Shah Institute of Technology, Thane
Date:

CERTIFICATE

This is to certify that the project Synopsis entitled ***"AI based Self Driving Car"*** Submitted by ***"Mahek Jain" (17204010),"Vinit Shah"(17204014),"Hiral Thadeshwar"(17204012)*** for the partial fulfillment of the requirement for award of a degree ***Bachelor of Engineering*** in ***Information Technology.***to the University of Mumbai,is a bonafide work carried out during academic year 2019-2020.

Mr. Vishal Badgujar
Co-Guide

Ms. Rujata Chaudhari
Guide

Mr. Kiran Deshpande
Head Department of Information Technology

Dr. Uttam D.Kolekar
Principal

External Examiner(s)

1.

2.

Place:A.P.Shah Institute of Technology, Thane
Date:

Declaration

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

_____

(Signature)

(Mahek Jain and 17204010)

_____

(Signature)

(Vinit Shah and 17204014)

_____

(Signature)

(Hiral Thadeshwar and 17204012)

Date:

Abstract

The basic idea behind the project is to develop an automated car that is capable of sensing its environment and moving without any human input. The automation will be achieved by detecting lane marking, signals, obstacles, stop sign using image processing and neural network to react and take decisions such as changing the course of the car, stopping on stop signs and red signal, and moving on green signal. Self-driving car processes inputs, plots a path, and sends instructions to the vehicle's actuators, which control acceleration, braking, and steering. Hard-coded rules, obstacle avoidance algorithms, predictive modeling, and "smart" object discrimination help the software follow traffic rules and navigate obstacles.

A monocular vision-based self-driving car prototype using Deep Neural Network on Raspberry Pi is proposed. Self-driving cars are one of the most increasing interests in recent years as the definitely developing relevant hardware and software technologies toward fully autonomous driving capability with no human intervention. Level-3/4 autonomous vehicles are potentially turning into a reality in near future. Convolutional Neural Networks (CNNs) have been shown to achieve significant performance in various perception and control tasks in comparison to other techniques in the latest years. The key factors behind these impressive results are their ability to learn millions of parameters using a large amount of labeled data. In this work, we concentrate on finding a model that directly maps raw input images to a predicted steering angle as output using a deep neural network. The technical contributions of this work are two-fold.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Human driven cars use technologies to provide safety and detect obstacles and auto stop in various high end cars but none of them works completely driver less. The existing cars does not contain the feature of automation to the extent that car can drive autonomously. There is constant need of drivers without it the car becomes unavailable but with self driving car we can make the availability of car constant on roads. In traditional cars the driver constantly needs to keep check on the signals, road safety signs, obstacles and lane and needs to make decisions accordingly.

Whereas a self-driving car, also known as a robot car, autonomous car, or driverless car, is a vehicle that is capable of sensing its environment and moving with little or no human input. Self-driving cars can detect environments using a variety of techniques such as radar, GPS and computer vision. It will also interpret sensory information to identify appropriate navigational paths, as well as obstacles and relevant signage. Self-driving cars have control systems that are capable of analyzing sensory data to distinguish between different cars on the road. This is very useful in planning a path to the desired destination.

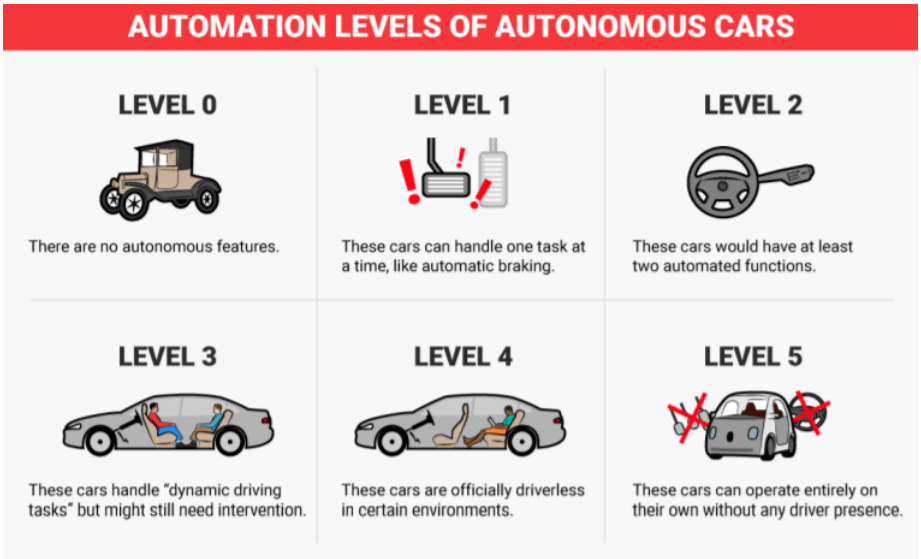


Figure 1.1: Automation Levels of Autonomous Car

The above diagram shows different automation levels which are described below:

Level 0 (No Automation): The full-time performance by the human driver of all aspects of the dynamic driving task, even when enhanced by warning or intervention systems.
Level 1 (Driver Assistance): The driving mode-specific execution by a driver assistance system of either steering or acceleration/deceleration using information about the driving environment and with the expectation that the human driver performs all remaining aspects of the dynamic driving task.
Level 2 (Partial Automation): The driving mode-specific execution by one or more driver assistance systems of both steering and acceleration/deceleration using information about the driving environment and with the expectation that the human driver performs all remaining aspects of the dynamic driving task.
Level 3 (Conditional Automation): The driving mode-specific performance by an automated driving system of all aspects of the dynamic driving task with the expectation that the human driver will respond appropriately to a request to intervene.
Level 4 (High Automation): The driving mode-specific performance by an automated driving system of all aspects of the dynamic driving task even if a human driver does not respond appropriately to a request to intervene.
Level 5 (Full Automation): The driving mode-specific performance by an automated driving system of all aspects of the dynamic driving task under all roadway and environmental conditions that can be managed by a human driver.

The advantages of a self driving car over human driven car can be:

1. Safety

Driving safety experts predict that once driverless technology has been fully developed, traffic collisions (and resulting deaths and injuries and costs), caused by human error, such as delayed reaction time, tailgating, rubbernecking, and other forms of distracted or aggressive driving should be substantially reduced.

2. Welfare

Automated cars could reduce labor costs relieve travelers from driving and navigation chores, thereby replacing behind-the-wheel commuting hours with more time for leisure or work and also would lift constraints on occupant ability to drive, distracted and texting while driving, intoxicated, prone to seizures, or otherwise impaired. For the young, the elderly, people with disabilities, and low income citizens, automated cars could provide enhanced mobility.

3. Traffic

Additional advantages could include higher speed limits smoother rides and increased roadway capacity and minimized traffic congestion, due to decreased need for safety gaps and higher speeds.

1.1 Objectives

The objectives behind developing this project are:

1. To have a driverless car.
Cars without driver will be helpfull to humans to a great extent,human mind can get diverted but the system has predefined set of rules to act in various situation which will reduce accidents.

2. To provide user convenience.
User just needs to enter the destination and he/she is done, user can sit back do his/her work or simply can rest for sometime instead of driving.

3. To make service available 24*7.
Human need rest so no one can drive for hours without rest, the system enables user to rest and travel 24*7 non stop without getting exhausted.

4. If a practical self-driving car is ever mass produced, the first major change would be to ride services.
If the implementation of project gets successful it will be a boon to companies providing ride services(ola,uber,zoom cars etc), as there will be no risk at all, the risk of appointing a good driver, checking driving skills, car safety all this risk will vanish.

5. Instead of relying on human instincts built up over millions of years that ensure selfpreservation, a computer will be safer.
Humans act differently in same situation so the chance of getting misintercepted increases which cause rise in number of accidents, but if things get computerised there will be same action taken by all cars in same scenerio which decreases accidents and increases safety.

1.2 Problem Definition

Problem in the existing system is that the existing cars needs effort to be driven and certain amount of time goes in reaching one destination which could be utilised in some other work.

To provide a solution to the above problem we are implementing the project where the existing human driven cars will be replaced by effective self driven cars keeping in mind of various safety conditions. So developing a model scaled car which can be implemented using monocular vision method and technology in an actual car. Implementing machine learning and image processing to gain better knowledge of new technologies. We are creating a prototype to explain the concepts of machine learning which will be helpful to teach the students as it can be presented as the live example of machine learning.

1.3 Technology Stack

-This project builds a self-driving RC car using Raspberry Pi, Arduino and open source software.
-Raspberry Pi collects inputs from a camera module and an ultrasonic sensor, and sends data to a computer wirelessly.
-The computer processes input images and sensor data for object detection (stop sign and traffic light) and collision avoidance respectively.
-A neural network model runs on computer and makes predictions for steering based on input images. Predictions are then sent to the Arduino for RC car control.
-The system uses sensor and camera constantly to check for obstacles and safety signs.

Hardware components used in the project are:

Raspberry Pi 3: SoC: Broadcom BCM2837, CPU: 4× ARM Cortex-A53, 1.2GHz, GPU: Broadcom VideoCore IV, RAM: 1GB LPDDR2 (900 MHz), Networking: 10/100 Ethernet, 2.4GHz 802.11n wireless, Bluetooth: Bluetooth 4.1 Classic, Bluetooth Low Energy, Storage: microSD, GPIO: 40-pin header, populated, Ports: HDMI, 3.5mm analogue audio-video jack, 4× USB 2.0, Ethernet, Camera Serial Interface (CSI), Display Serial Interface (DSI).

OpenCV : Decision tree learning, Expectation-maximization algorithm, k-nearest neighbor algorithm, Naive Bayes classifier, Artificial neural networks, Random forest, Support vector machine (SVM), Deep neural networks (DNN).

Arduino Nano: Microcontroller ATmega328, Operating Voltage (logic level): 5 V, Input Voltage (recommended): 7-12 V, Input Voltage (limits): 6-20 V, Digital I/O Pins : 14 (of which 6 provide PWM output), Analog Input Pins: 8, DC Current per I/O Pin: 40 mA, Flash Memory 32 KB (ATmega328) of which 2 KB used by bootloader, SRAM: 2 KB (ATmega328), EEPROM: 1 KB (ATmega328), Clock Speed: 16 MHz, Dimensions: 0.73” x 1.70”.

Ultrasonic sensor : Power Supply :+5V DC, Quiescent Current : ¡2mA, Working Current: 15mA, Effectual Angle: ¡15°, Ranging Distance : 2cm – 400 cm/1 – 13ft, Resolution : 0.3 cm, Measuring Angle: 30 degree, Trigger Input Pulse width: 10uS, Dimension: 45mm x 20mm x 15mm.

Pi camera : 8 megapixel native resolution sensor-capable of 3280 x 2464 pixel static images, Supports 1080p30, 720p60 and 640x480p90 video, Camera is supported in the latest version of Raspbian, Raspberry Pi’s preferred operating system.

Chapter 2

Literature Review

During the development of the project following papers has been referred :

1. Real Time Self-Driving Car Navigation Using Deep Neural Network, Truong-Dong Do, MinhThien Duong, Quoc-Vu Dang and My-Ha Le, in International Conference on Green Technology and Sustainable Development (GTSD), 2018

-In this paper, they have presented an autonomous car platform based on the softmax function squashes the outputs of each unit to be between 0 and 1, just like a sigmoid function.[1] The softmax function acts as sigmoid function by ranging the output while an actual softmax function does not do that. Using deep neural network helps in giving real time output. They have implemented the model on MATLAB simulator before implementing actually. The system uses only one single camera for all inputs and it drives at about 5-6 km/hr whether the lane markings are present or no. This model only detects lane markings and turn signs. It just hovers the car left or right and does not sense signals or stop sign.

2. Neural controller of autonomous driving mobile robot by an embedded camera, Hajer Omrane, Mohamed Slim Masmoudi and Mohamed Masmoudi, in International Conference on Advanced Technologies For Signal and Image Processing - ATSIP, 2018

-They have built an autonomous RC Car that uses Artificial Neural Network (ANN) for control. It describes the theory behind the neural network and autonomous vehicles.[2] Using L298N IC and motor driver a car is made which can be controlled by a microcontroller and then in return sent to the model car. Using cnn helps in only detection of gray scale parts and it ignores the unnecessary data in the detection. The system used is very limited but accurate. Using an embedded pi camera for input and gray scale of images for training in neural network.[3] The system detects lane markings for each direction and does not offer any other functionality other than that.

3. Driverless Car: Autonomous Driving Using Deep Reinforcement Learning In Urban Environment, Abdur R. Fayjie, Sabir Hossain, Doukhi Oualid, and Deok-Jin Lee, in 15th International Conference on Ubiquitous Robots (UR) Hawaii Convention Center, Hawai'i, USA, June 27-30, 2018

-In this paper, they have presented a reinforcement-learning based approach with Deep Q Network implemented in autonomous driving.[4] Using lidar sensors it detect objects at a very far distance. The whole system is developed on a simulator depicting actual roads and city streets with traffic. Using fusion of camera and lidar helps in better knowing of the surroundings and all kinds of obstacles. They have implemented a model using lidar(laser sensor) which is a very costly sensor and it is applicable for large scale cars.

4. Object Detection Using Deep Neural Networks, Malay Shah, Prof, Rupal Kapdi, in International Conference on Intelligent Computing and Control Systems ICICCS 2017

-The problem discussed in this article is object detection using deep neural network especially convolution neural networks. Object detection was previously done using only conventional deep convolution neural network whereas using regional based convolution network[5] increases the accuracy and also decreases the time required to complete the program.[6] Training a neural network from scratch takes more time and processing power as it is very difficult to find the dataset of sufficient size and ground truth. Using Regional Convolutional Neural Network(RCNN) helps in finding appropriate regions in image and it enables the system to give real time outputs. This deep neural network is used for image processing, mainly for medical uses like tumor and such where the data set is too complex to detect regions in comparison to a model road environment.

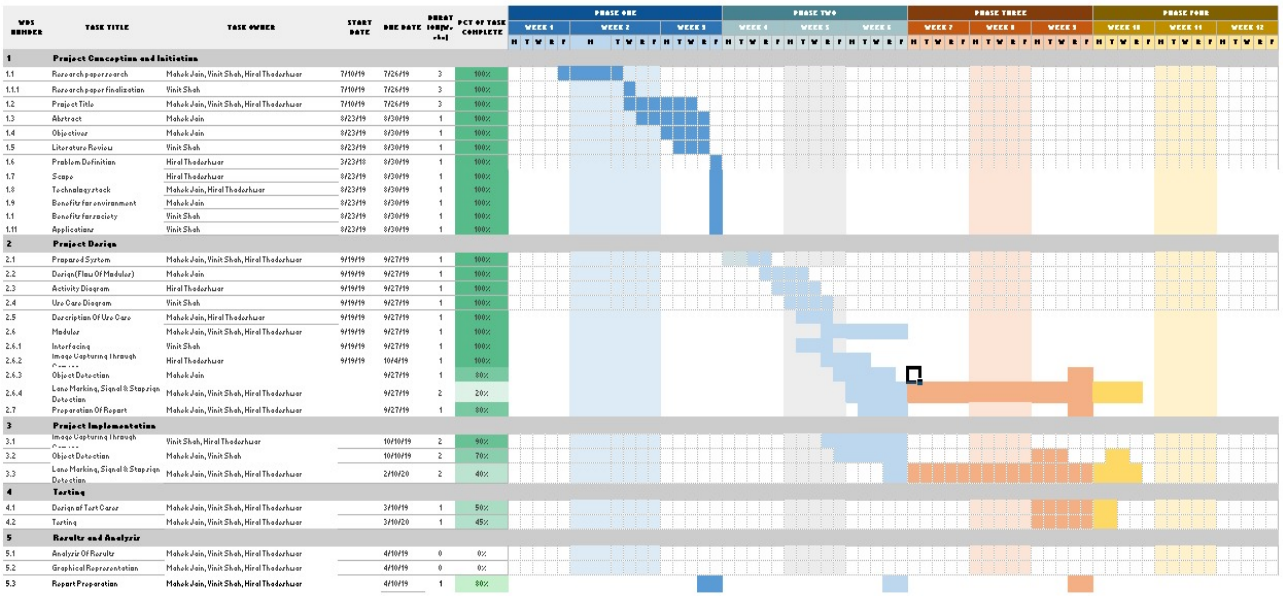
5. Self-driving Cars Using CNN and Q-learning, Syed Owais Ali Chishti, Sana Riaz, Muhammad Bilal Zaib, Mohammad Nauman, in 2018 IEEE 21st International Multi-Topic Conference (INMIC)

-They have developed a car which is trained for three different road signs; Stop, No left, and Traffic light using DQN with existing CNN model. These road signs are detected in the environment using OpenCV cascade classifiers. Supervised learning was done using Convolutional Neural Network and we achieved 73perc test and 89perc train accuracy.[7] In addition to this they trained the car with reinforcement learning for three different sign boards; Stop, No left and Traffic light using deep Q-learning technique with CNN model. Currently, the training is done in a constrained environment, many factors of real environment can sway the prediction of model. Lightening effects and weather conditions influences the images they get from cameras which can have a great impact on model prediction.

6. Working model of Self-driving car using Convolutional Neural Network, Raspberry Pi and Arduino, Aditya Kumar Jain, in Proceedings of the 2nd International conference on Electronics, Communication and Aerospace Technology (ICECA 2018)

-Their proposed model takes an image with the help of Pi cam attached with Raspberry Pi on the car. The Raspberry-Pi and the laptop is connected to the same network, the Raspberry Pi sends the image captured which serves as the input image to the Convolutional Neural Network. The image is gray-scaled before passing it to the Neural Network. Upon prediction the model gives one of the four output i.e. left, right, forward or stop. When the result is predicted corresponding Arduino signal is triggered which in turn helps the car to move in a particular direction with the help of its controller.[8] Their car was trained under different combinations of the track i.e. straight, curved, combination of straight and curved and etc. Total of 24 videos were recorded out of which images were extracted.10868 images were extracted and was categorically placed in different folders like left, right, straight and stop. In this paper, a method to make a model of self-driving car is presented. The different hardware components along with software and neural network configuration are clearly described. With the help of Image Processing and Machine Learning a successful model was developed which worked as per expectation. Thus the model was successfully designed, implemented and tested. The car slightly moves out of the track which can be a serious issue if it hits nearby objects if we consider a real car.

Project Timeline Chart



WBS NUMBER	TASK TITLE	TASK OWNER	START DATE	DUE DATE	DURATION (DAYS)	PCT OF TASK COMPLETE
1	**Project Conception and Initiation**					
1.1	Research paper search	Mahek Jain, Vinit Shah, Hiral Thadeshwar	7/10/19	7/24/19	3	100%
1.1.1	Research paper finalization	Vinit Shah	7/10/19	7/24/19	3	100%
1.2	Project Title	Mahek Jain, Vinit Shah, Hiral Thadeshwar	7/10/19	7/24/19	3	100%
1.3	Abstract	Mahek Jain	8/23/19	8/30/19	1	100%
1.4	Objective	Mahek Jain	8/23/19	8/30/19	1	100%
1.5	Literature Review	Vinit Shah	8/23/19	8/30/19	1	100%
1.6	Problem Definition	Hiral Thadeshwar	8/23/19	8/30/19	1	100%
1.7	Scope	Hiral Thadeshwar	8/23/19	8/30/19	1	100%
1.8	Technology stack	Mahek Jain, Hiral Thadeshwar	8/23/19	8/30/19	1	100%
1.9	Benefits for environment	Mahek Jain	8/23/19	8/30/19	1	100%
1.1	Benefits for society	Vinit Shah	8/23/19	8/30/19	1	100%
1.11	Applications	Vinit Shah	8/23/19	8/30/19	1	100%
2	**Project Design**					
2.1	Proposed System	Mahek Jain, Vinit Shah, Hiral Thadeshwar	9/19/19	9/27/19	1	100%
2.2	Design(Flow Of Modular)	Mahek Jain	9/19/19	9/27/19	1	100%
2.3	Activity Diagram	Hiral Thadeshwar	9/19/19	9/27/19	1	100%
2.4	Use Case Diagram	Vinit Shah	9/19/19	9/27/19	1	100%
2.5	Description Of Use Case	Mahek Jain, Hiral Thadeshwar	9/19/19	9/27/19	1	100%
2.6	Modular	Mahek Jain, Vinit Shah, Hiral Thadeshwar	9/19/19	9/27/19	1	100%
2.6.1	Interfacing	Vinit Shah	9/19/19	9/27/19	1	100%
2.6.2	Image Capturing through Camera	Hiral Thadeshwar	9/19/19	10/4/19	1	100%
2.6.3	Object Detection	Mahek Jain		9/27/19	1	80%
2.6.4	Lane Marking, Signal & Stop sign Detection	Mahek Jain, Vinit Shah, Hiral Thadeshwar		9/27/19	2	20%
2.7	Preparation Of Report	Mahek Jain, Vinit Shah, Hiral Thadeshwar		9/27/19	1	80%
3	**Project Implementation**					
3.1	Image Capturing through Camera	Vinit Shah, Hiral Thadeshwar		10/10/19	2	90%
3.2	Object Detection	Mahek Jain, Vinit Shah		10/10/19	2	70%
3.3	Lane Marking, Signal & Stop sign Detection	Mahek Jain, Vinit Shah, Hiral Thadeshwar		2/10/20	2	40%
4	**Testing**					
4.1	Design of Test Cases	Mahek Jain, Vinit Shah, Hiral Thadeshwar		3/10/19	1	50%
4.2	Testing	Mahek Jain, Vinit Shah, Hiral Thadeshwar		3/10/20	1	45%
5	**Results and Analysis**					
5.1	Analysis Of Results	Mahek Jain, Vinit Shah, Hiral Thadeshwar		4/10/19	0	0%
5.2	Graphical Representation	Mahek Jain, Vinit Shah, Hiral Thadeshwar		4/10/19	0	0%
5.3	Report Preparation	Mahek Jain, Vinit Shah, Hiral Thadeshwar		4/10/19	1	80%

Figure 2.1: Project Timeline Chart

Chapter 3

Proposed System

The system consists of three subsystems: input unit (camera, ultrasonic sensor), processing unit (computer) and RC car control unit.

Input Unit
A Raspberry Pi board (model B+), attached with a pi camera module and an HC-SR04 ultrasonic sensor is used to collect input data. Two client programs run on Raspberry Pi for streaming color video and ultrasonic sensor data to the computer via local Wi-Fi connection. In order to achieve low latency video streaming, video is scaled down to QVGA (320×240) resolution.

Processing Unit
The processing unit (computer) handles multiple tasks: receiving data from Raspberry Pi, neural network training and prediction(steering), object detection(stop sign and traffic light), distance measurement(monocular vision), and sending instructions to Arduino through USB connection.

TCP Server
A multithread TCP server program runs on the computer to receive streamed image frames and ultrasonic data from the Raspberry Pi. Image frames are converted to gray scale and are decoded into numpy arrays.

Neural Network
One advantage of using neural network is that once the network is trained, it only needs to load trained parameters afterwards, thus prediction can be very fast. Only lower half of the input image is used for training and prediction purposes. There are 38,400 (320×120) nodes in the input layer and 32 nodes in the hidden layer. The number of nodes in the hidden layer is chosen fairly arbitrary. There are four nodes in the output layer where each node corresponds to the steering control instructions: left, right, forward and reverse respectively (though reverse is not used anywhere in this project, it's still included in the output layer). First each frame is cropped and converted to a numpy array. Then the train image is paired with train label (human input). Finally, all paired image data and labels are saved into a npz file. The neural network is trained in OpenCV using back propagation method. Once training is done, weights are saved into a xml file. To generate predictions, the same neural network is constructed and loaded with the trained xml file.

Object Detection
This project adapted the shape-based approach and used Haar feature-based cascade classifiers for object detection. Since each object requires its own classifier and follows the same process in training and detection, this project only focused on stop sign and traffic light detection.
OpenCV provides a trainer as well as detector. Positive samples (contain target object) were acquired using a cell phone, and were cropped that only desired object is visible. Negative samples (without target object), on the other hand, were collected randomly. In particular, traffic light positive samples contains equal number of red traffic lights and green traffic light. The same negative sample dataset was used for both stop sign and traffic light training. Below shows some positive and negative samples used in this project.

To recognize different states of the traffic light(red, green), some image processing is needed beyond detection. Flowchart below summarizes the traffic light recognition process.
Firstly, trained cascade classifier is used to detect traffic light. The bounding box is considered as a region of interest (ROI). Secondly, Gaussian blur is applied inside the ROI to reduce noises. Thirdly, find the brightest point in the ROI. Finally, red or green states are determined simply based on the position of the brightest spot in the ROI.

Figure 3.1: Haar cascade classifiers for object detection

Firstly, trained cascade classifier is used to detect traffic light. The bounding box is considered as a region of interest (ROI). Secondly, Gaussian blur is applied inside the ROI to reduce noises. Thirdly, find the brightest point in the ROI. Finally, red or green states are determined simply based on the position of the brightest spot in the ROI.

Distance Measurement
Raspberry Pi can only support one pi camera module. Using two USB web cameras will bring extra weight to the RC car and also seems unpractical. Therefore, monocular vision method is chosen.
This project adapted a geometry model of detecting distance to an object using monocular vision method proposed by Chu, Ji, Guo, Li and Wang (2004).

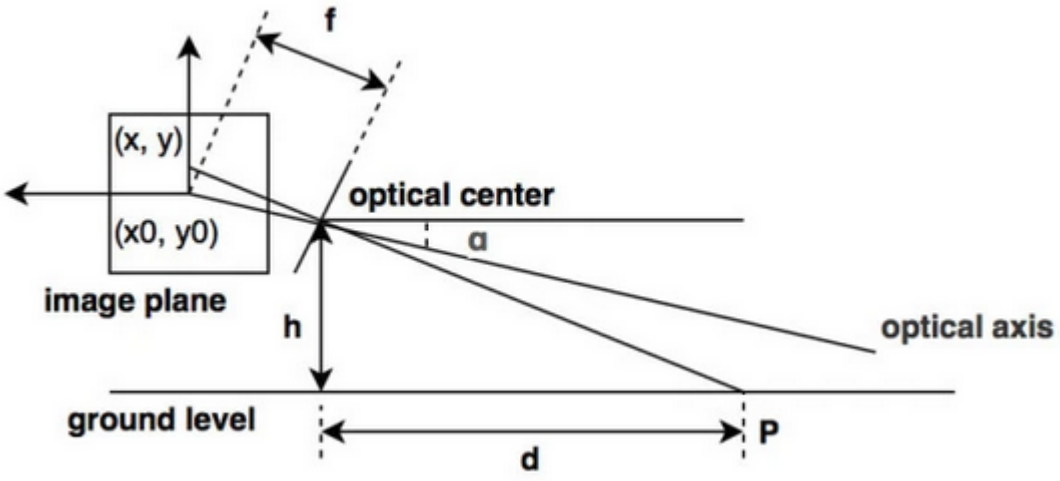


Figure 3.2: Distance measurement using monocular vision

P is a point on the target object; d is the distance from optical center to the point P. Based on the geometry relationship above, formula (1) shows how to calculate the distance d. In the formula (1), f is the focal length of the camera; is camera tilt angle; h is optical center height; (x0, y0) refers to the intersection point of image plane and optical axis; (x, y) refers to projection of point P on the image plane. Suppose O1 (u0,v0) is the camera coordinate of intersection point of optical axis and image plane, also suppose the physical dimension of a pixel corresponding to x-axis and y-axis on the image plane are dx and dy. Then:

$$d = h/\tan(\partial + \arctan((y - y_0)/f)) \quad (1)$$

$$u = \frac{x}{dx} + u_0 \quad v = \frac{y}{dy} + v_0 \quad (2)$$

Let $x_0 = y_0 = 0$, from (1) and (2):

$$d = h/\tan(\alpha + \arctan((v - v_0)/a_y)), \qquad (a_y = f/dy) \quad (3)$$

v is the camera coordinates on y-axis and can be returned from the object detection process. All other parameters are camera's intrinsic parameters that can be retrieved from camera matrix. ;ine OpenCV provides functions for camera calibration. Camera matrix for the 5MP pi camera is returned after calibration. Ideally, a(x) and a(y) have the same value. Variance of these two values will result in non-square pixels in the image. The matrix below indicates that the fixed focal length lens on pi camera provides a reasonably good result in handling distortion aspect. Here is an interesting article discussing the focal length of pi camera with stock lens and its equivalent to 35mm camera.

The matrix returns values in pixels and h is measured in centimeters. By applying formula

$$\begin{bmatrix} a_x = 331.7 & 0 & u_0 = 161.9 \\ 0 & a_y = 332.3 & v_0 = 119.8 \\ 0 & 0 & 1 \end{bmatrix}$$

(3), the physical distance d is calculated in centimeters.

RC Car Control Unit

The RC car used in this project has an on/off switch type controller. When a button is pressed, the resistance between the relevant chip pin and ground is zero. Thus, an Arduino board is used to simulate button-press actions. Four Arduino pins are chosen to connect four chip pins on the controller, corresponding to forward, reverse, left and right actions respectively. Arduino pins sending LOW signal indicates grounding the chip pins of the controller; on the other hand sending HIGH signal indicates the resistance between chip pins and ground remain unchanged. The Arduino is connected to the computer via USB. The computer outputs commands to Arduino using serial interface, and then the Arduino reads the commands and writes out LOW or HIGH signals, simulating button-press actions to drive the RC car.

Chapter 4

Project Design

4.1 System Architecture :

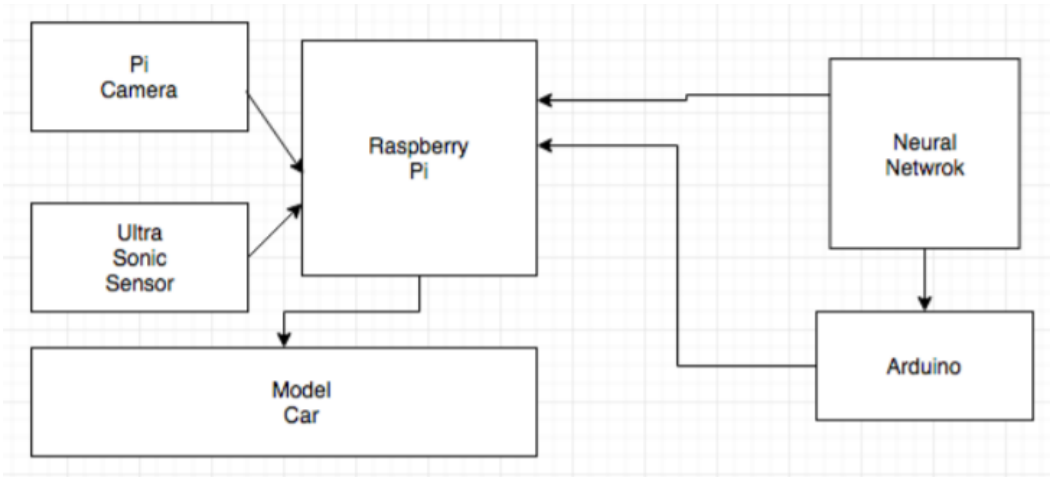


Figure 4.1: System Architecture

The system architecture consists of a pi camera mounted on the raspberry pi with an ultrasonic sensor, these two send data to the raspberry pi. The raspberry is connected to the neural network on the computer via local wireless network. The Arduino is connected to the computer (neural network) which sends remote controls to the car. The car is mounted with the raspberry pi which has the ultrasonic sensor and pi camera attached to it.

4.2 Usecase Diagram :

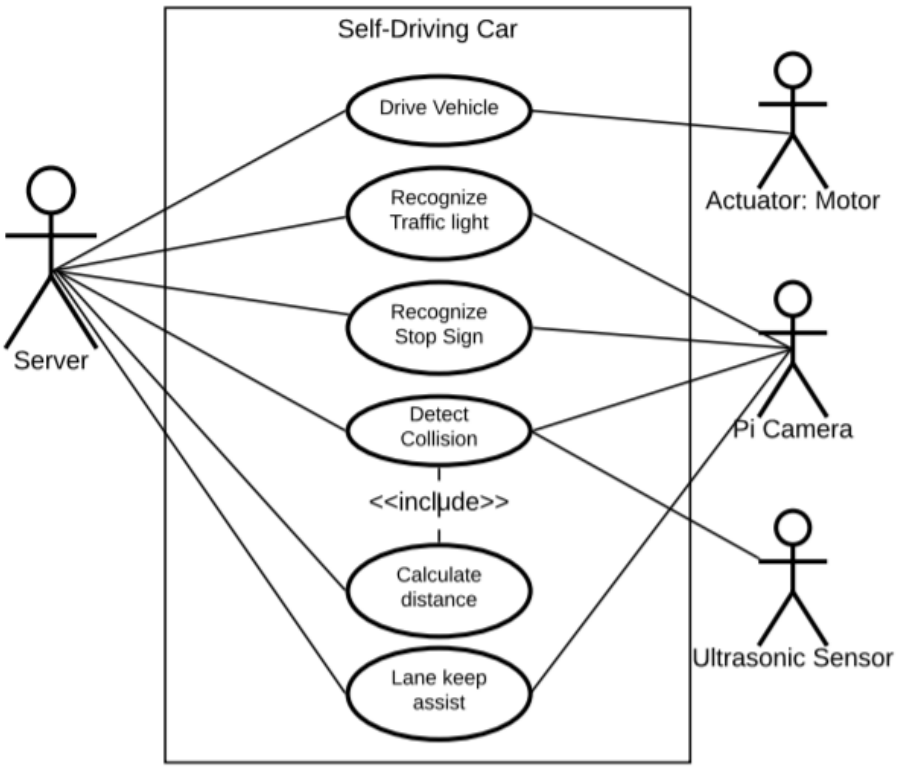


Figure 4.2: Usecase Diagram

There are four actors in the use cases consisting of server, pi camera, ultrasonic sensor and the motor. The motor of the car drives the vehicle and the server gives instructions on how to drive. The pi camera recognises signal and stop sign with the help of the server and detects collision as well which the ultrasonic sonic sensor helps in. The distance for the same is calculated by the server. The server and pi camera together keeps the car inside the lanes and hover around the markings.

4.3 Flow Diagram :

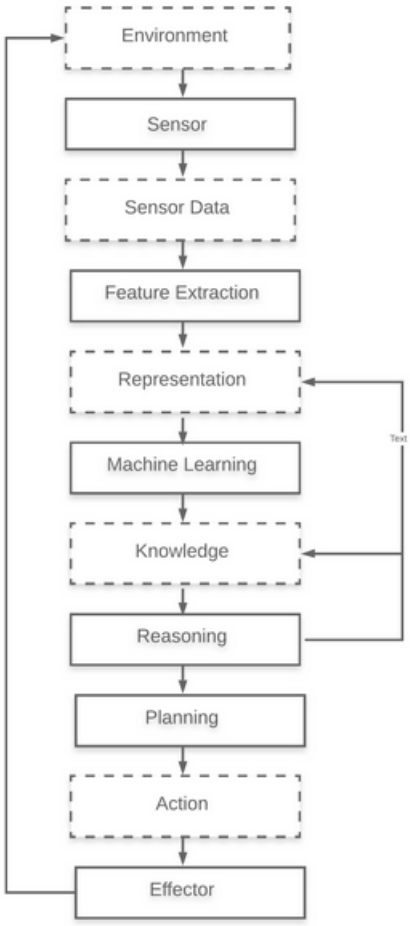


Figure 4.3: Flow Diagram

The environment for our car consist of lane lines, signal and stop sign and other obstacles which are detected using the sensors like camera, ultrasonic sensor the sensor data is streamed to server for feature extraction that is the data from the image and sensor is used to analyze the position and situation of car and then data representation is done, machine learning is used for autonomous decision making to decide the direction in which the car should turn or take hold, with more application of machine learning knowledge and reasoning is developed with that knowledge the planning is done for the action to be taken after that as per decided effectors are triggered to make the car take actions.

4.4 Flowchart for Image Processing :

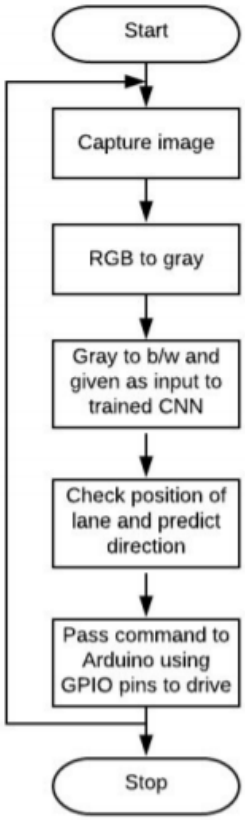


Figure 4.4: Flowchart for Image Processing

The pi camera is used to capture the image of the lane it is streamed to the server where it is converted from rgb to grey scale image and then from grey to black and white to reduce the size and remove the noise so that it can furthur be given to CNN for training, after that CNN checks the position of the lane and takes direction like left, right, front or stop. After that the commands respective to the direction are sent to arduino via gpio pin.

4.5 Flowchart for Lane Detection :

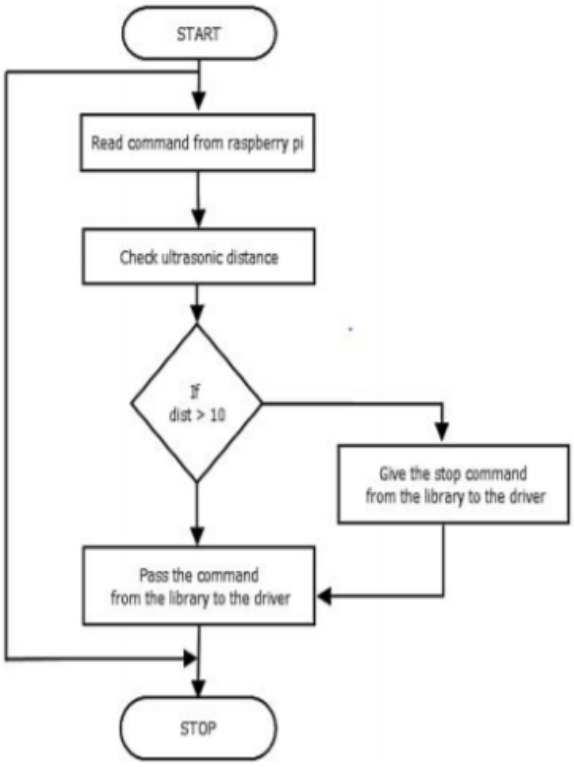


Figure 4.5: Flowchart for Lane Detection

The raspberry pi board attached to ultrasonic continuously sends sensor data to the server where distance is checked, if the distance is less than the specified distance then the car is stoped otherwise move normally.

4.6 Sequence diagram for Image Processing :

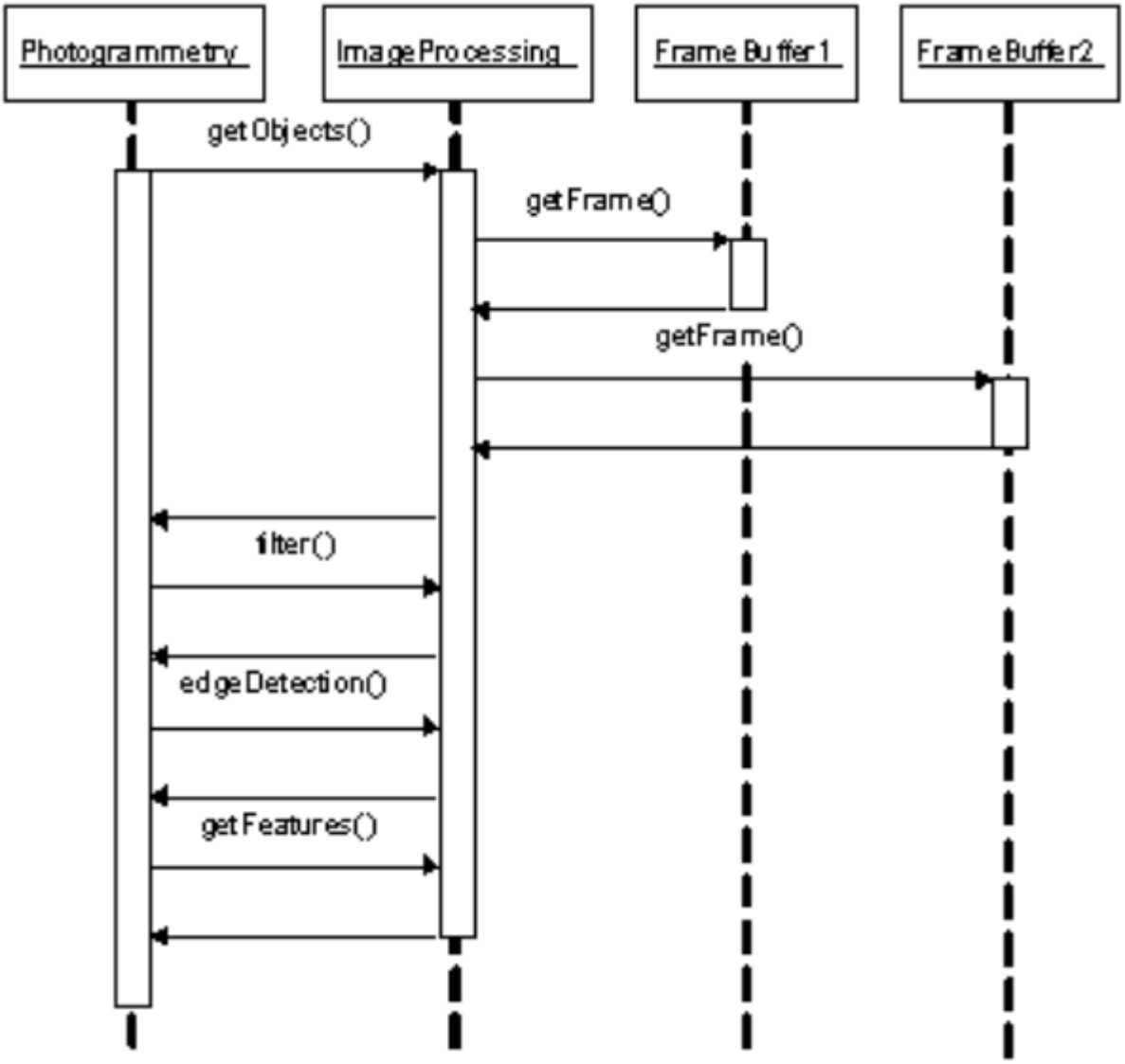


Figure 4.6: Sequence Diagram for Image Processing

In this sequence diagram shows the sequence of image processing first the image is captured and by getobject() sent for image processing where it is broken into frames by getframe() and send frames to image processing which then sends to photogrammetry where filters are applied to frame and then edge detection is done after that the features are extracted.

4.7 Sequence diagram for Obstacle Detection :

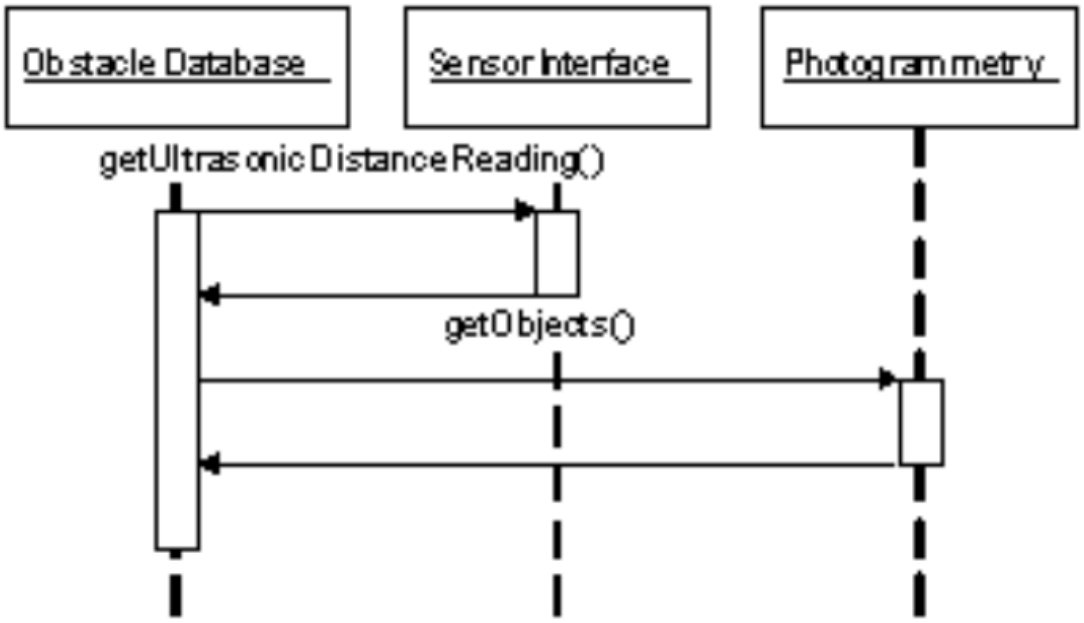


Figure 4.7: Sequence Diagram for Obstacle Detection

In this sequence diagram first the ultrasonic sensor's distance readings are taken from sensor interface and sent to obstacle database via getultrasonicdistancereading() to check if the distance is less than specified so that stop action can be taken also image is taken from photogrammetry via getobjects() to check the distance.

4.8 Activity Diagram :

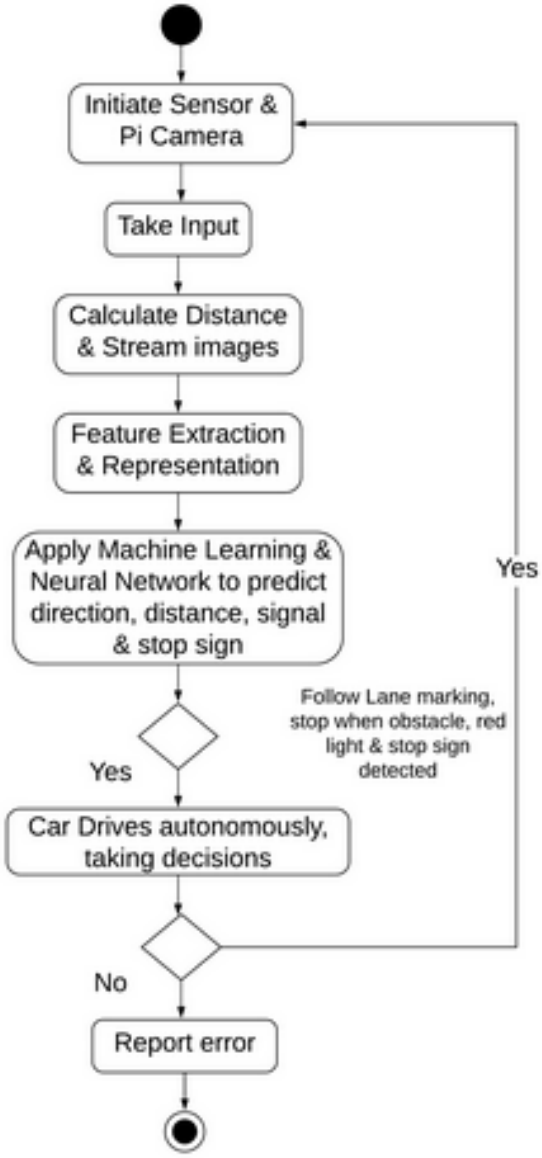


Figure 4.8: Activity Diagram

The model aims to implement automation by handling tasks such as self-driving through lane line detection, stop sign and traffic signal detection and fore collision avoidance. The system design for implementing the same will consist of three units first is input unit containing a pi camera and ultrasonic sensor, the second is processing unit which is our laptop that will act as a server, the neural network will be running over here and third, is RC control unit which is Arduino. Firstly, in the input unit, the raspberry pi board of the B+ model will be connected with the pi camera and an ultrasonic sensor to stream input data. Two client programs will be running on raspberry pi one to stream images collected by the pi camera and another to send sensor data through a local wi-fi connection.

Then the processing handles more than one tasks such as collecting data from raspberry pi, neural network training and steering prediction, stop sign and signal detection, distance measurement using monocular vision and sending commands to Arduino through a USB connection. A multithreaded TCP server program will execute on the raspberry pi to receive image frames and sensor data. The image frames will be converted to grayscale and decoded into NumPy arrays. The neural network which is the convolutional neural network will be trained to make steering predictions based on detected lane markings. The lower half from the input image will be used for training and prediction purposes. There will be input, hidden and output layers, there will be four nodes in the output layer each corresponding to steering control instructions that are left, right, forward and reverse. The training data can be collected or the datasets already available can be used. For training, each frame will be cropped and transformed to a NumPy array. Then the train image will be paired with the train label. Then all the paired image data and labels will be stored in the npz file. OpenCV will be used to train the neural network. After training the weights will be stored in an XML file and for generating predictions the same neural network will be created and loaded with the trained XML file. For signal and stop sign detection that is part of object detection will be done using a shape-based approach; Haar feature-based cascade classifier will be used for object detection. Since each object requires its classifier Haar cascade is used. Monocular vision will be used for distance measurement for object detection, it will be helpful in determining at what distance from the object the car should take hold. Also, it will be more appropriate for the design as the raspberry pi has only one pi camera slot and using a USB web-camera will also add weight to the car. The RC car used in our prototype has an on/off or high/low switch type controller. Therefore, the Arduino board will be used to imitate button press actions. Four Arduino pins will be used to connect four chip pins on the rc remote controller, corresponding to forward, reverse, left and right actions respectively. The Arduino will be connected to the computer through USB and the computer will send the output commands and write out low or high signals, simulating button press actions to drive the car autonomously.

Chapter 5

Project Implementation

5.1 Rccontroller.ino on the arduino

```
void forward_right(int time){
  digitalWrite(forward_pin, LOW);
  digitalWrite(right_pin, LOW);
  delay(time);
}

void reverse_right(int time){
  digitalWrite(reverse_pin, LOW);
  digitalWrite(right_pin, LOW);
  delay(time);
}

void forward_left(int time){
  digitalWrite(forward_pin, LOW);
  digitalWrite(left_pin, LOW);
  delay(time);
}

void reverse_left(int time){
  digitalWrite(reverse_pin, LOW);
  digitalWrite(left_pin, LOW);
  delay(time);
}
```

This is the part of a program loaded into the Arduino where the pins according to direct are sent high or low signals to move the car in either of the directions. A toggle switch on the remote controller is connected to the arduino's pins and set for four different directions.

5.2 Ultrasonicstream programs on the raspberry pi

```
# referring to the pins by GPIO numbers
GPIO.setmode(GPIO.BCM)

# define pi GPIO
GPIO_TRIGGER = 23
GPIO_ECHO    = 24

# output pin: Trigger
GPIO.setup(GPIO_TRIGGER,GPIO.OUT)
# input pin: Echo
GPIO.setup(GPIO_ECHO,GPIO.IN)
# initialize trigger pin to low
GPIO.output(GPIO_TRIGGER, False)

try:
    while True:
        distance = measure()
        print "Distance : %.1f cm" % distance
        # send data to the host every 0.5 sec
        client_socket.send(str(distance).encode('utf-8'))
        time.sleep(0.5)
finally:
    client_socket.close()
    GPIO.cleanup()
```

It specifies the pin to which the ultrasonic sensor is connected and the distance is measured through the ultrasonic sensor which is sent through a socket connection between the server and the raspberry. The distance is measured by echoing and the time is calculated by which it has elapsed multiplied by 34300 and it is halved to get the distance in centromeres.

5.3 PI camera streaming program on the raspberry pi

```
# create socket and bind host
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client_socket.connect(('192.168.1.100', 8000))
connection = client_socket.makefile('wb')

try:
    with picamera.PiCamera() as camera:
        camera.resolution = (320, 240)      # pi camera resolution
        camera.framerate = 15               # 15 frames/sec
        time.sleep(2)                       # give 2 secs for camera to initilize
        start = time.time()
        stream = io.BytesIO()

        # send jpeg format video stream
        for foo in camera.capture_continuous(stream, 'jpeg', use_video_port = True):
            connection.write(struct.pack('<L', stream.tell()))
            connection.flush()
            stream.seek(0)
            connection.write(stream.read())
            if time.time() - start > 600:
                break
            stream.seek(0)
            stream.truncate()
    connection.write(struct.pack('<L', 0))
```

This is a snippet of pi camera streaming program on the raspberry Pi. A socket connection is made to the server by adding the host ip address and a port number. A resolution of 320*420 is taken from the pi camera at the frame rate of 15 frames/sec. The bytes are streamed in JPEG format by the video port of raspberry pi.

5.4 Collect training data on server

```
# collect images for training
print("Start collecting images...")
print("Press 'q' or 'x' to finish...")
start = cv2.getTickCount()

X = np.empty((0, self.input_size))
y = np.empty((0, 4))

# stream video frames one by one
try:
    stream_bytes = b' '
    frame = 1
    while self.send_inst:
        stream_bytes += self.connection.read(1024)
        first = stream_bytes.find(b'\xff\xd8')
        last = stream_bytes.find(b'\xff\xd9')

        if first != -1 and last != -1:
            jpg = stream_bytes[first:last + 2]
            stream_bytes = stream_bytes[last + 2:]
            image = cv2.imdecode(np.frombuffer(jpg, dtype=np.uint8), cv2.IMREAD_GRAYSCALE)

            # select lower half of the image
            height, width = image.shape
            roi = image[int(height/2):height, :]

            cv2.imshow('image', image)

            # reshape the roi image into a vector
            temp_array = roi.reshape(1, int(height/2) * width).astype(np.float32)

            frame += 1
            total_frame += 1
```

This program is to collect images to train the model. The video is streamed from the pi camera on the car, where the car is run in the model to capture images on a key press. The images are converted into grayscale through a cv2 function. The frames are collected and stored into a npz file created by this program. This program is ran on the server with the streamserver program on the raspberry Pi which streams the camera video to the server.

5.5 Program to train the model(model.py) on the server

```
class NeuralNetwork(object):
    def __init__(self):
        self.model = None

    def create(self, layer_sizes):
        # create neural network
        self.model = cv2.ml.ANN_MLP_create()
        self.model.setLayerSizes(np.int32(layer_sizes))
        self.model.setTrainMethod(cv2.ml.ANN_MLP_BACKPROP)
        self.model.setActivationFunction(cv2.ml.ANN_MLP_SIGMOID_SYM, 2, 1)
        self.model.setTermCriteria((cv2.TERM_CRITERIA_COUNT, 100, 0.01))

    def train(self, X, y):
        # set start time
        start = time.time()

        print("Training ...")
        self.model.train(np.float32(X), cv2.ml.ROW_SAMPLE, np.float32(y))

        # set end time
        end = time.time()
        print("Training duration: %.2fs" % (end - start))

    def evaluate(self, X, y):
        ret, resp = self.model.predict(X)
        prediction = resp.argmax(-1)
        true_labels = y.argmax(-1)
        accuracy = np.mean(prediction == true_labels)
        return accuracy
```

This is a part of the training of the model from the frames collected in collect-training-data. A neural network is created by using cv2 where an artificial neural networks class multi-layer perception (MLP) is used. The training duration is calculated with the highest prediction taken into consideration, the accuracy of it is returned.

5.6 rcdriver.py program on the server for definition on functions and threshold

```
class VideoStreamHandler(socketserver.StreamRequestHandler):

    # h1: stop sign, measured manually
    # h2: traffic light, measured manually
    h1 = 5.5  # cm
    h2 = 5.5

    # load trained neural network
    nn = NeuralNetwork()
    nn.load_model("saved_model/nn_model.xml")

    obj_detection = ObjectDetection()
    rc_car = RCControl("/dev/tty.usbmodem1421")

    # cascade classifiers
    stop_cascade = cv2.CascadeClassifier("cascade_xml/stop_sign.xml")
    light_cascade = cv2.CascadeClassifier("cascade_xml/traffic_light.xml")

    d_to_camera = DistanceToCamera()
    # hard coded thresholds for stopping, sensor 30cm, other two 25cm
    d_sensor_thresh = 30
    d_stop_light_thresh = 25
    d_stop_sign = d_stop_light_thresh
    d_light = d_stop_light_thresh

    stop_start = 0  # start time when stop at the stop sign
    stop_finish = 0
    stop_time = 0
    drive_time_after_stop = 0
```

This shows the part of the program where the actual car is driven. This snippet specifies the threshold values for the ultrasonic sensor to detect an obstacle and the camera to detect a stop sign and a traffic signal. The distance for stop sign and traffic signal is measured manually.

5.7 rcdriver.py program on the server for distance measurement and stop conditions

```
# lower half of the image
height, width = gray.shape
roi = gray[int(height/2):height, :]

# object detection
v_param1 = self.obj_detection.detect(self.stop_cascade, gray, image)
v_param2 = self.obj_detection.detect(self.light_cascade, gray, image)

# distance measurement
if v_param1 > 0 or v_param2 > 0:
    d1 = self.d_to_camera.calculate(v_param1, self.h1, 300, image)
    d2 = self.d_to_camera.calculate(v_param2, self.h2, 100, image)
    self.d_stop_sign = d1
    self.d_light = d2

cv2.imshow('image', image)
# cv2.imshow('mlp_image', roi)

# reshape image
image_array = roi.reshape(1, int(height/2) * width).astype(np.float32)

# neural network makes prediction
prediction = self.nn.predict(image_array)

# stop conditions
if sensor_data and int(sensor_data) < self.d_sensor_thresh:
    print("Stop, obstacle in front")
    self.rc_car.stop()
    sensor_data = None

elif 0 < self.d_stop_sign < self.d_stop_light_thresh and stop_sign_active:
    print("Stop sign ahead")
    self.rc_car.stop()
```

This is the part of the rcdriver program where the image from the pi camera is calculated for various parameters. The object detection for traffic light and stop sign is detected by stop-cascade and light-cascade. The distance is measured between the stop sign and traffic light. The neural network makes the prediction of the image-array. The ultrasonic sensor when catches an object within the threshold, it stops the car and the same conditions are checked for the stop sign and traffic light and accordingly the decision is taken.

Chapter 6

Testing

For the testing purpose we opted to go for the functional testing methods. Functional testing involves testing the application against the business requirements. It incorporates all test types designed to guarantee each part of a piece of software behaves as expected by using uses cases provided by the design team or business analyst. Function testing includes :
1. Unit Testing
2. Integration Testing

Unit Testing :

Unit testing is the first level of testing and is often performed by the developers themselves. It is the process of ensuring individual components of a piece of software at the code level are functional and work as they were designed to. Developers in a test-driven environment will typically write and run the tests prior to the software or feature being passed over to the test team. Unit testing also makes debugging easier because finding issues earlier means they take less time to fix than if they were discovered later in the testing process.

Therefore, opting for the unit testing method in our project played a crucial role in assessing each module of the application separately. We test different units as the connection between the hardware components is tested, then the streaming of ultrasonic sensor and camera frames is tested separately.

Integration Testing :

After each unit is thoroughly tested, it is integrated with other units to create modules or components that are designed to perform specific tasks or activities. These are then tested as group through integration testing to ensure whole segments of an application behave as expected (i.e, the interactions between units are seamless). Integrated tests can be conducted by either developers or independent testers and are usually comprised of a combination of automated functional and manual tests.

We have performed integration testing by integrating the streaming of sensor and camera data to the server and check if the tcp connection works for both. Also the units are integrated and tested in programs where the model is trained and images are collected.

Various Testcases :

Test Case Name	Test Condition	Test Steps/ Procedure	Test Data	Expected Result	Actual Result	Pass/ Fail
rc_control_test.py	Control from keyboard	A pygame window which takes input from the keyboard	Directions	The car should move by the keyboard keys	The car could be controlled by the computers keyboard	Pass
stream_server_test.py	Stream video	A stream program on raspberry captures the video	Video frames	The program should open a window to display the cameras video	A 320*120 window displayed the pi cameras output.	Pass
Ultrasonic_server_test.py	Ultrasonic sensor data	A stream program on raspberry pi that captures ultrasonic data and measures into distance	Distance in cms	To get the distance of any object from sensor on the computer	The server got the distance in centimetres on the screen for a time limit.	Pass
collect_training_data.py	Frames captured	Captures frames to train the model which are calculated	Total frames and dropped frames	The frames should be all saved as captured.	10 frames dropped out of total frames	Pass
model.py	Accuracy	Training of model followed by prediction of it	Labels for prediction with arguments	Achieve maximum accuracy	93% accuracy	Pass

Table 6.1: Testcases

Chapter 7

Conclusions and Future Scope

Automation in cars in real world is a very big field, where many sensors come into action, our idea is to solve many of those into two sensors by detecting distance and objects like stop sign, signals and other obstacles with a single method of monocular vision which can be enhanced in future from a scaled car to an actual car. The prototype focuses on these functionalities which we are developing in a model rc car while others focus on only one aspect of it.

The issue we saw in preparing/testing the system is camera latency. It is characterized as the period from the time the camera sensor watches the scene to the time the computer reads all things considered peruses the digitized picture information. Lamentably, this time can be fundamentally long depending upon the camera and the performance of the Pi, which is around 300-350 milliseconds. This is strikingly higher than the dormancy of human observation, which is referred to be as quick as 13 milliseconds [16]. Higher camera idleness could adversely influence control execution, particularly for security basic applications, in light of the fact that the profound neural system would break down stale scenes.

As can be seen, there are numerous territories we could investigate to drive this venture further and get much all the more persuading results. Later on, we keep on examining approaches to accomplish better expectation exactness in preparing the system, recognize and utilize low-latency cameras, just as improving the presentation of the RC vehicle stage, particularly identified with precise steering angle control. Moreover, we are going to bring throttle into the model with the desire of accomplishing more significant levels of autonomous vehicles.

Bibliography

[1] Truong-Dong Do, Minh-Thien Duong, Quoc-Vu Dang and My-Ha Le, "Real Time Self-Driving Car Navigation Using Deep Neural Network" in International Conference on Green Technology and Sustainable Development (GTSD), 2018

[2] Hajer Omrane, Mohamed Slim Masmoudi and Mohamed Masmoudi, "Neural controller of autonomous driving mobile robot by an embedded camera" in International Conference on Advanced Technologies For Signal and Image Processing - ATSIP, 2018

[3] F. Shumaila Mateenuddin, Mrs. V.S. Jahagirdar, "An Android Controlled Mini Rover for real time surveillance using Raspberry Pi 3", in International Journal Of Advanced Research in Engineering Management (IJAREM), 2017.

[4] Abdur R. Fayjie, Sabir Hossain, Doukhi Oualid, and Deok-Jin Lee, "Driverless Car: Autonomous Driving Using Deep Reinforcement Learning In Urban Environment" in 15th International Conference on Ubiquitous Robots (UR) Hawaii Convention Center, Hawai'i, USA, June 27-30, 2018

[5] Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014, pp. 580-587

[6] Malay Shah, Prof, Rupal Kapdi, "Object Detection Using Deep Neural Networks", in International Conference on Intelligent Computing and Control Systems ICICCS 2017

[7] Syed Owais Ali Chishti, Sana Riaz, Muhammad Bilal Zaib, Mohammad Nauman, "Self-driving Cars Using CNN and Q-learning", in 2018 IEEE 21st International Multi-Topic Conference (INMIC)

[8] Aditya Kumar Jain, "Working model of Self-driving car using Convolutional Neural Network, Raspberry Pi and Arduino", in Proceedings of the 2nd International conference on Electronics, Communication and Aerospace Technology (ICECA 2018) IEEE Conference Record 42487; IEEE Xplore ISBN:978-1-5386-0965-1

Appendices

Appendix-A: Setting up environment with Anaconda

Step 1 : Install miniconda(Python3) on your computer
Download the installer:
Miniconda installer for Windows.
Anaconda installer for Windows.
Verify your installer hashes.
Double-click the .exe file.
Follow the instructions on the screen.
If you are unsure about any setting, accept the defaults. You can change them later.
When installation is finished, from the Start menu, open the Anaconda Prompt.
Test your installation. In your terminal window or Anaconda Prompt, run the command conda list. A list of installed packages appears if it has been installed correctly.
Step 2 : Create auto-rccar environment with all necessary libraries for this project conda env create -f environment.yml
name: auto-rccar
dependencies: python,numpy,matplotlib,jupyter,scikit-learn,cython,
pip: pygame,pyserial,opencv-python
This yml file will add all the dependencies into conda environment so there is no need of installation of all those packages separately.
Step 3 : Activate auto-rccar environment source activate auto-rccar
Step 4 : Activate the environment and browse to the directory where the programs are stored and run the programs according the requirement

Appendix-B: Programs

test/

rc_control_test.py: RC car control with keyboard

stream_server_test.py: video streaming from Pi to computer

ultrasonic_server_test.py: sensor data streaming from Pi to computer

model_train_test/

data_test.npz: sample data

train_predict_test.ipynb: a jupyter notebook that goes through neural network model in OpenCV3

raspberryPi/

stream_client.py: stream video frames in jpeg format to the host computer

ultrasonic_client.py: send distance data measured by sensor to the host computer

arduino/

rc_keyboard_control.ino: control RC car controller

computer/

cascade_xml/

trained cascade classifiers

chess_board/

images for calibration, captured by pi camera

picam_calibration.py: pi camera calibration

collect_training_data.py: collect images in grayscale, data saved as *.npz

model.py: neural network model

model_training.py: model training and validation

rc_driver_helper.py: helper classes/functions for rc_driver.py

rc_driver.py: receive data from raspberry pi and drive the RC car based on model prediction

rc_driver_nn_only.py: simplified rc_driver.py without object detection

Appendix-C: Procedure and Calibration

Testing: Flash rc_keyboard_control.ino to Arduino and run rc_control_test.py to drive the RC car with keyboard. Run stream_server_test.py on computer and then run stream_client.py on raspberry pi to test video streaming. Similarly, ultrasonic_server_test.py and ultrasonic_client.py can be used for sensor data streaming testing.

Pi Camera calibration (optional): Take multiple chess board images using pi camera module at various angles and put them into chess_board folder, run picam_calibration.py and returned parameters from the camera matrix will be used in rc_driver.py.

Collect training/validation data: First run collect_training_data.py and then run stream_client.py on raspberry pi. Press arrow keys to drive the RC car, press q to exit. Frames are saved only when there is a key press action. Once exit, data will be saved into newly created training_data folder.

Neural network training: Run model_training.py to train a neural network model. Please feel free to tune the model architecture/parameters to achieve a better result. After training, model will be saved into newly created saved_model folder.

Cascade classifiers training (optional): Trained stop sign and traffic light classifiers are included in the cascade_xml folder, if you are interested in training your own classifiers, please refer to OpenCV doc and this great tutorial.

Self-driving in action: First run rc_driver.py to start the server on the computer, and then run stream_client.py and ultrasonic_client.py on raspberry pi.

Acknowledgement

We have great pleasure in presenting the report on **AI based Self Driving Car.** We take this opportunity to express our sincere thanks towards our guide **Ms. Rujata Chaudhari** & Co-Guide **Mr. Vishal Badgujar** Department of IT, APSIT thane for providing the technical guidelines and suggestions regarding line of work. We would like to express our gratitude towards his constant encouragement, support and guidance through the development of project.

We thank **Mr. Kiran B. Deshpande** Head of Department,IT, APSIT for his encouragement during progress meeting and providing guidelines to write this report.

We thank **Ms. Apeksha Mohite** and **Ms. Anagha Aher** BE project co-ordinator, Department of IT, APSIT for being encouraging throughout the course and for guidance.

We also thank the entire staff of APSIT for their invaluable help rendered during the course of this work. We wish to express our deep gratitude towards all our colleagues of APSIT for their encouragement.

Hiral Thadeshwar:
17204012:

Mahek Jain:
17204010:

Vinit Shah:
17204014:

Publication

Paper entitled **"AI based Self Driving Car"** is accepted at **"Communication and Signal Processing,ICCCSP 2020"** by **"Hiral Thadeshwar","Mahek Jain","Vinit Shah","Rujata Chaudhari","Vishal Badgujar".**