# Autonomous Delivery Robot

by

Sarin Rahman Zavin
20101307
Kazi Habibur Rahaman
20101559
Safat Ahmed Nayeem
20101109
SK Rubayet Bin Mujahid
19241009
Swapnil Sarkar
20141013

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

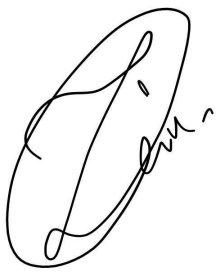
Department of Computer Science and Engineering
Brac University
May 2023

# Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

**Student's Full Name & Signature:**

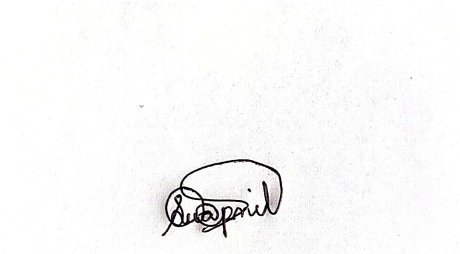


**Sarin Rahman Zavin**
20101307

**Kazi Habibur Rahaman**
20101559

**Safat Ahmed Nayeem**
20101109

**SK Rubayet Bin Mujahid**
19241009



**Swapnil Sarkar**
20141013

# Approval

The thesis/project titled "Autonomous Delivery Robot" submitted by

1. Sarin Rahman Zavin (20101307)
2. Kazi Habibur Rahaman (20101559)
3. Safat Ahmed Nayeem (20101109)
4. SK Rubayet Bin Mujahid (19241009)
5. Swapnil Sarkar (20141013)

Of Spring, 2023 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on May 22, 2023.

**Examining Committee:**

Supervisor:
(Member)

Dr. Md. Khalilur Rahman
Professor
Department of Computer Science and Engineering
Brac University

Program Coordinator:
(Member)

Dr. Golam Rabiul Alam
Associate Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD
Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

# Abstract

Autonomous driving is now a topic of great interest in the modern world. With the advancement of AI and hardware technology, it is now possible to tackle many obstacles a human usually faces. The concept of the autonomous delivery robot explains and interacts with the process of transportation of products without any human interference. This is a broad platform that does not only work with advanced AI mechanisms but also implements the problem-solving and data analyzing power of the machine. In today's world, it is a necessity to complete a task efficiently and punctually which is near impossible to do without the involvement of technology. However, if the technology can be utilized to do the whole process, it would greatly increase the work's effectiveness using minimum time. The right usage of the autonomous delivery robot can ensure a task's completion even under unfavorable circumstances. Moreover, the problem of traffic congestion and limited space can be overcome by this. Additionally, it will enable the opportunity to invest the human workforce in more important situations rather than being stuck at the delivery works. Therefore, it is high time to adapt to the process of performing delivery tasks through the autonomous system for our own benefits and advancements.

**Keywords:** Autonomous, Self-Driving Robot, Last-Mile Delivery, Computer Vision

# Acknowledgement

Firstly, all praise to the Great Allah for whom our thesis has been completed without any major interruption.

Secondly, we thank our supervisor Dr. Md. Khalilur Rahman for giving us the opportunity to research this topic under his guidance. We would like to express our special gratitude and appreciation to him for guiding and assisting us throughout the whole phase and giving us such attention and time.

Thirdly, to research assistant Mr. Sayantan Roy Arko sir for his kind support and advice in our work. He helped us whenever we needed help.

And finally to our parents. Without their support, it may not be possible. With their kind support and prayer, we are now on the verge of our graduation.

# Table of Contents

# List of Figures

# Chapter 1

# Introduction

## 1.1 Background

Autonomy is the concept of mechanisms governing themself or having control over their own actions. For modern technology, autonomy is a matter of great importance. The main reason for its popularity is that it opens up a great improvement section for the current technology. The usage of machines already made day-to-day tasks easy. But what if the machines are given the capability of solving and finishing the work from start to end all by themselves? It would not only save humans from the struggle of maintaining the workflow but also create a self-operating society. Moreover, it would decrease time wastage and labor risks, increase efficiency and reduce the error margin of jobs.

The term "Last Mile" refers to the last part of the supply chain, where the product is handed over to the end consumer. In logistics, this step is the final and the most challenging part of a shipment. The problems related to last-mile delivery are the following.

1. **Navigation:** Traditionally searching the drop-off point to plan the route to the destination, everything is done by the delivery person. Normally the routes are not optimized and the delivery process gets slower.
2. **Human Errors:** Humans can take wrong turns when navigating and also cannot avoid traffic signals. These issues also add up to delays in the delivery process.
3. **Larger Carbon Footprint:** Since a car is being driven by a human who is susceptible to mistakes and makes the delivery process longer. The vehicle has to be driven for a longer time which would cause more carbon emissions.
4. **Customer Unavailability:** As delivery personnel has to deliver a huge amount of parcels, it would is often disappointing for the worker to arrive at a locked door. It is also not viable for a human to call and reach out to every customer before delivering their product.

5. **Reverse Logistics:** Returning purchased goods can be a problem with human-driven delivery systems. A person cannot deliver and receive products from the customers at the same time.

6. **High Cost:** Fuel, Driver, Assistant, and packaging all add up to the cost of the Last-Mile delivery system.

## 1.2 Reasons to use autonomy for delivery

The delivery process is a scattered and diverse section of society. This line of work is of multiple functionalities requiring both time efficiency and service quality. Additionally, there are chances of errors if done without or with limited technical assistance. Moreover, the delivery processes can cause traffic congestion, stress among workers, and accidents. These can be solved by making machines do all the work. The idea is to give the delivery responsibilities solely to the machines and minimize the drawbacks. For this purpose, level 4 driving automation also known as high automation will be implemented as it gives full control to the automation devices. The current world is greatly involved with the order and delivery process. Also, the field itself is full of varieties. Hence, assigning a human workforce to maintain this network will require a lot of resources even then it would not be error-free. That is why the autonomous system should be implemented to raise the effectiveness and perfection of the delivery system.

## 1.3 Research Objectives

This paper aims to solve the aforementioned problems and make Last-Mile delivery a seamless process.

1. Understand the working principles of Computer vision and autonomous driving.

2. Integrating optimized navigating system.

3. Lane, obstacle, and street sign detection.

4. Preserving the parcels and delivering them to the end user securely.

## 1.4 Autonomous Delivery Robot

This thesis focuses on introducing and creating a fast, efficient and affordable delivery system. Different functionality and methods are to be used to construct a suitable carrier device. The delivery machine is a wheeled vehicle so it will have comparatively low cost and higher reach. The main function of the carrier device is delivering products to the destinations but overcoming any obstacles in its path without any human interactions. The devices will use computer vision and data analysis abilities to avoid hindrances on track. Moreover, the usage of sonar and infrared-based obstacle detection devices are to be used to monitor, judge, and choose delivery paths. Finally, the Raspberry Pi model will be the base of the computing. GPS and Graphhopper API will be used to help in navigation. After shipment, the receiver would use their phone's inbuilt NFC to open the secure compartment of their parcel. Additionally, displays and other indicators would be used for status signals.

# Chapter 2

# Literature Review

## 2.1 Related Works

To make advances in driving without drivers on board, several tests and studies have been conducted. A completely autonomous system is still some way off. Researchers from all across the world have been working hard for many years to make this concept a reality. In 1939, General Motors pioneered the first self-driving automotive model, the first of its kind. This model was powered by an electric motor that ran on magnetic metal spikes embedded in the road and was regulated by radio-driven electromagnetic fields. This model became operational in 1958. The vehicle was outfitted with sensors that detected the electric current flowing through a wire buried in the pavement. By altering the current, the steering wheel could be moved left or right. Later, in 1961, at the height of the space race, scientists began to consider how to land cars on the moon. As a result, James Adams created the Stanford Cart, which is fitted with cameras and is meant to automatically detect and follow lines on the ground. By the early 2000s, the autonomous car industry was thriving. DARPA, the research arm of the United States Department of Defense, sponsored a number of competitions to enhance autonomous vehicles. In the modern era, Tesla is one of the leading manufacturers of autonomous vehicles and they are already on the third level of automation. Convolutional neural networks (CNNs) were employed by Nvidia engineers in a recent automotive application to translate the unprocessed images from a front-facing camera into steering instructions for a self-driving vehicle. This effective end-to-end methodology implies that the system learns to steer, with or without lane markers, on both local roads and highways, with the least amount of training data from people. The technology is also capable of working in environments with hazy visual cues, such as parking lots or gravel roadways. Our proposed model will utilize Convolutional Neural Networks to Process Visual Data.

## 2.2 The Pursuit of Level 5 automation and its difficulties

Known as the ultimate destination for self-driving cars, Level 5 autonomy introduces true driverless cars. At level 5, a vehicle can manoeuvre itself anywhere in any situation without any human intervention. A vehicle with level 5 autonomy is not constrained by geofencing or weather, and it carries components efficiently without the need for a driver, eliminating the requirement for a steering wheel and pedals. The only human involvement will be in deciding where to go.

High-level AV development concepts are mostly built on scenario-driven and task-oriented methodologies to undertake particular service development under predetermined situations. As a result of the unlimited number of conceivable scenarios, the situation-driven notion may pose challenges for AVs, suggesting that autonomous cars need to adjust to any circumstance. As a result, Level 5 AVs are predicted to outperform human drivers, as well as significantly improve vehicle performance.

There are numerous traffic participants in a real-world traffic scenario. The time fluctuation and behavioral unpredictability of each participant, for raises the complexity of a traffic system. Some common problems in developing AVs, as discussed in Ref. [26], include limitations in present technology and insufficient infrastructures. According to the existing state of perception, decision-making, and control technologies, as described in prior research [6], [10], these numerous innovations are necessary for the actual emergence of Level 5 AVs. Chen et al. [27], for example, proposed a unique idea of event-based autonomous driving neuron morphological vision, which can assist advanced AVs in acquiring more accurate visual perception data. This is critical for the advancement of more complex AVs, particularly Level 5 AVs. However, there is a mismatch between the technical standards of high-level AVs and the present state of developments.

## 2.3 Problems in delivery Sector

In the modern era, the supply and logistics sector is still facing a great deal of hurdles. This is mainly due to how modern cities are being built and major changes in the transportation systems. The rise of home deliveries has exacerbated the problem which requires high degrees of service, and fast delivery with little margin for error. The most prominent issue the delivery sector faces is not delivering products on time. Urbanization and stringent consumer demands for availability and timeliness have increased shipping frequency but have come at a cost of poor delivery efficiency with a lack of sustainable vision. The old logistic system with man-powered machines is simply not enough to keep up with the demand and challenges of these times. A rapid change and a complete overhaul are needed where the reliance on humans has to be reduced.

## 2.4 Emerging Solutions and other related works

Multiple companies have begun to realize the gravity of the problem. Rising expenses, the decline in manpower, and the rise in consumerism and demand for faster delivery have propelled firms to come up with innovative solutions.
DHL and Bring, two global logistics companies with operations in Sweden, have made investments in self-service technology that allow flexible parcel collecting for online shoppers.

Joerss et al. (2016) highlight four dominant home delivery methods in McKinsey-Company's fall report on travel, transportation, and logistics. They mention drones, autonomous ground vehicles, bike couriers, and droids as prominent delivery models in terms of expense and consumer choices, in addition to today's model, which depends primarily on human delivery personnel.

AGVs, or autonomous ground vehicles, enclose packages while they are being transported, acting almost like mobile warehouses. The size and loading capacity of these vehicles varies, but they all adhere to non-human, movable pick-up sites. AGVs operate on streets in unison with automobile traffic because of their size, in contrast to autonomous droids, smaller autonomous ground vehicles intended for pavement transit (Joerss et al., 2014). A company called starship is already operating in US and UK since 2018 where the bots have a plethora of sensors, including radar, ultrasonic, and cameras with computer vision that can recognize solid things like curbs and walls. However, human operators are still needed to standby. Safety is always a concern and these robots are suitable for only small distances within a specific city. Another company called CARNET has implemented the same concept but with a standout feature-they're able to go up and down stairs. But the autonomous ecosystem must be further perfected to obtain acceptance and trust among citizens.

In summary, the prerequisites for new delivery choices favor ground-based devices because the criteria for aerial solutions may be judged too high for near-term profitability. Furthermore, despite tiny city centers and busy streets, anyone unfamiliar with driving is frequently directed to pavement areas. Furthermore, principles introduced in the last mile context should be capable to respond to and reschedule potentially unsuccessful delivery efforts, as well as take appropriate countermeasures. They should also have flexibility with regard to time and place to completely meet the needs of their customers. Furthermore, obtaining technical advancements is required for proper manoeuvring and building a valuable presence in urban locations in order to garner market acceptability.

# Chapter 3

# Work Plan

The purpose of this robot would be to take an address and the parcel from the courier and drop it off at the destination address. The robot itself would be approximately a 2ft-by-2ft vehicle each wheel driven by an individual motor and a ball caster. The steering of the vehicle will be achieved with individual rotation of the drive wheels and the free movement of the ball caster. The base of the vehicle would house the electronic components and above that, the parcel compartment would be situated. The tasks and the working of the vehicle are listed below.

1. The vehicle would get its navigation data from Graphhopper API once it is input by the user at the courier facility.
2. Using Computer Vision and inbuilt sensor the robot would navigate the world to find its way to the destination.
3. Once the destination is reached, the recipient would receive a notification on their phone that their parcel is nearby.
4. The receiver would approach the vehicle and enter a pin to retrieve the parcel.
5. The vehicle would then move on to deliver the next parcel.

## 3.1 Description of the model

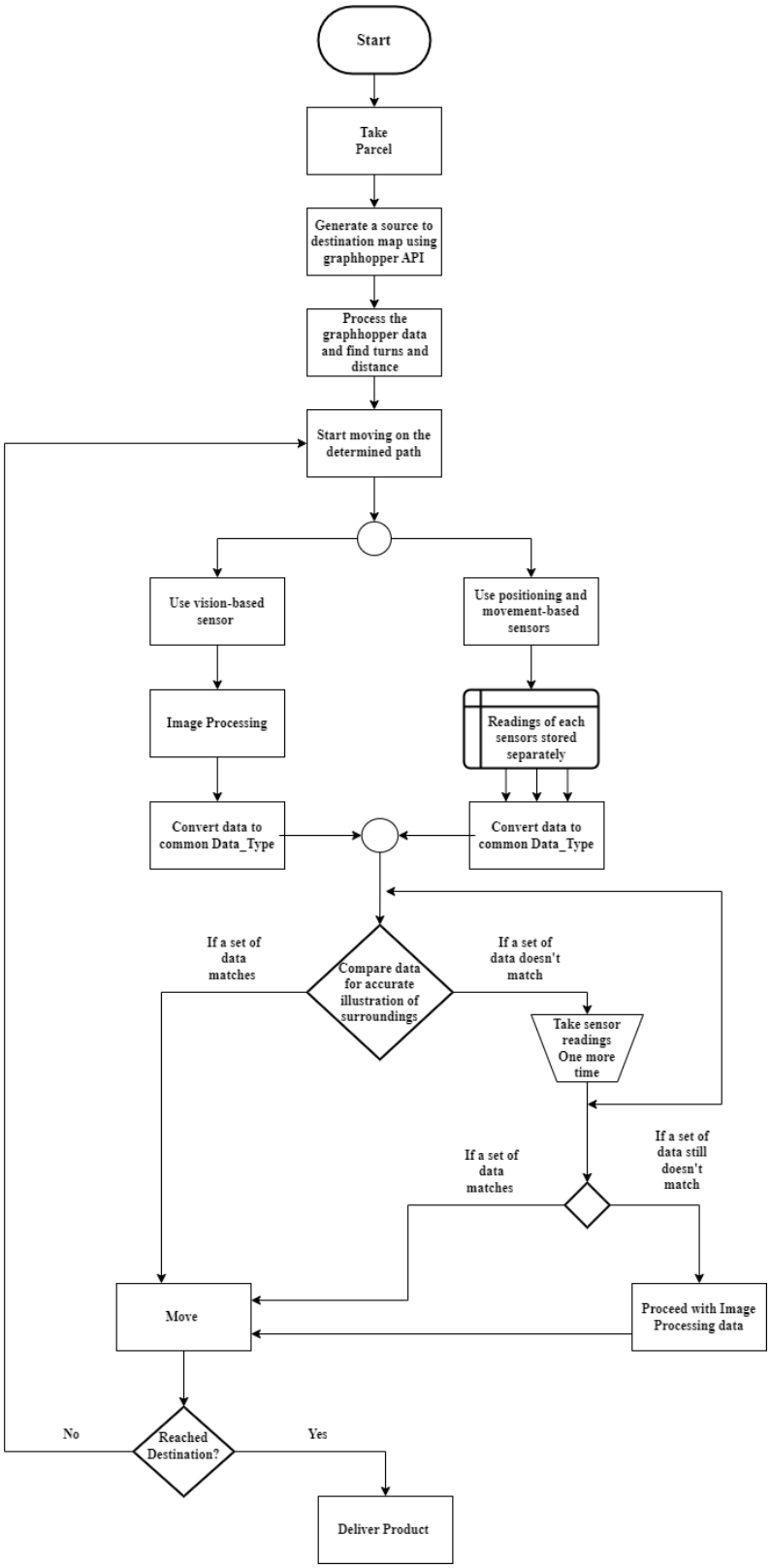



Figure 3.1: Workflow

At first, the delivery bot stores the parcels within its chamber. The bot has a carry-weight limitation. When the limit is reached it will not receive anymore products and will dispatch.

Each product will have a label attached to it which will have the receiver's information as a QR code or Bar code. The bot can get the destination by manual configuration or by scanning the label of the product. The bot will take its own position via GPS and will calculate a feasible path to the destination using the graphhopper API. For data import, the graphhopper uses OSMReader which imports OpenStreetMap data. After importing data, graphhopper uses its routing package which contains several shortest-path algorithms such as Dijkstra and A* to calculate the shortest path. For less complexity and time efficiency, the delivery bot only uses the uni-directional path-finding algorithm of graphhopper (GraphHopper, 2018). After getting the directions and readings, the delivery bot will start moving toward the destination.

The generated shortest path may not always be the best feasible path. For example, there could be some blockage along the path which will not be notified in the generated map. For this reason, the delivery bot uses continuous path observation and path planning to avoid obstacles. The base functionality of the bot's obstacle detection is dependent on live images. The bot has a camera module installed for navigation. The delivery bot takes live images and processes them to detect if the path forward is okay or not. The system has a decision-making algorithm that can take proper measures in case an obstacle is detected, without any human interaction.

The delivery bot's system has decision-making capability. The data feed onto the system by the camera and sensors go through a comparison where it is checked if the readings from these components are illustrating the same scenario or not. If a set of data matches the system will take that into account and discard other dissimilar readings. Now analyzing the selected set of data which explains the condition of the path to the bot, the bot makes the decision if it will follow the path or take an alternative route. The decision-making is done by a trained virtual model and is completely autonomous. The model is explained in the later part of this paper.

After reaching the destination, the product chamber can be accessed by only the registered receiver. The sealed chamber can be opened by OTP code sent to the receiver's phone.

## 3.2 Description of the data

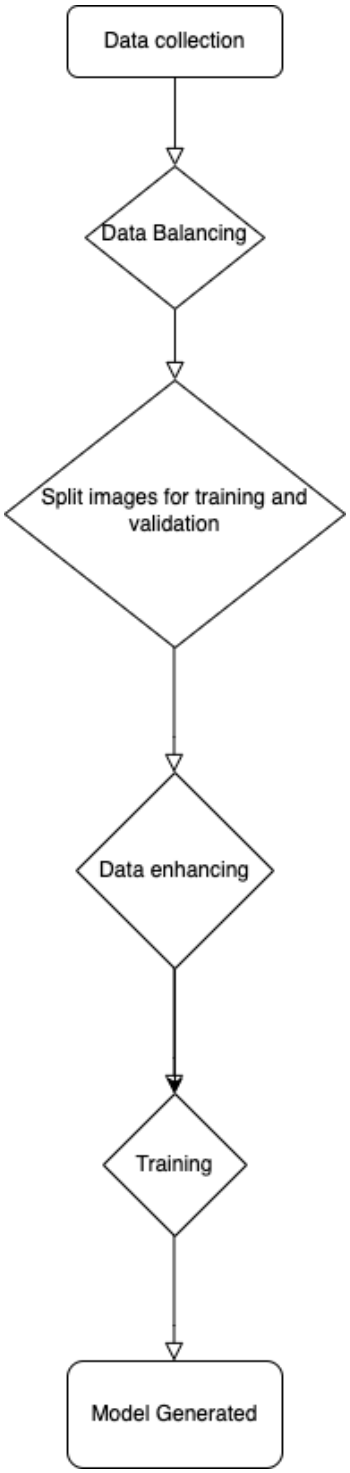



Figure 3.2: Data Processing

1. **Data Collection:** The bot will be driven manually by the operator and simultaneously the camera will record the images at 10 frames per second. Along with the pictures, the steering angle will also be captured and stored on a . CSV file to be used later on.

2. **Data Balancing:** The steering data will be processed by binning to remove minor errors. As most of our driving is going forward, we have to limit the number of samples taken when going forward to process the data efficiently.

3. **Splitting:** The image data has to be split into two parts. One is to train the neural network and the other is for validation. When training the validation data set would be used to check if the predicted model is working or not.

4. **Data Enhancing:** The images we collected, in the beginning, are not enough to train the data. Hence we have to artificially translate, pan, and apply zoom to the image. The image has to be augmented as well, where the color space of the image will be changed to YUV color space from RGB, and blur will be added and cropped accordingly.

5. **Training:** The images are now processed and the data would be used to train according to Nvidia's model.

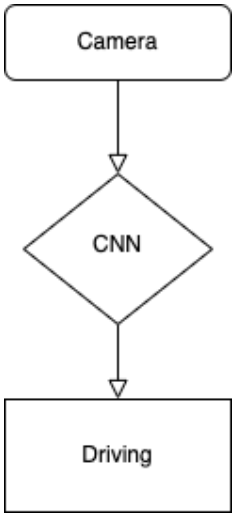



Figure 3.3: Driving System

**Autonomous Driving:** In the application phase, the camera will capture images and feed them into the trained model. The model will then predict the steering output of the robot. The onboard sensors will work along with the output from the computed driving data to move.

**Network Architecture:**

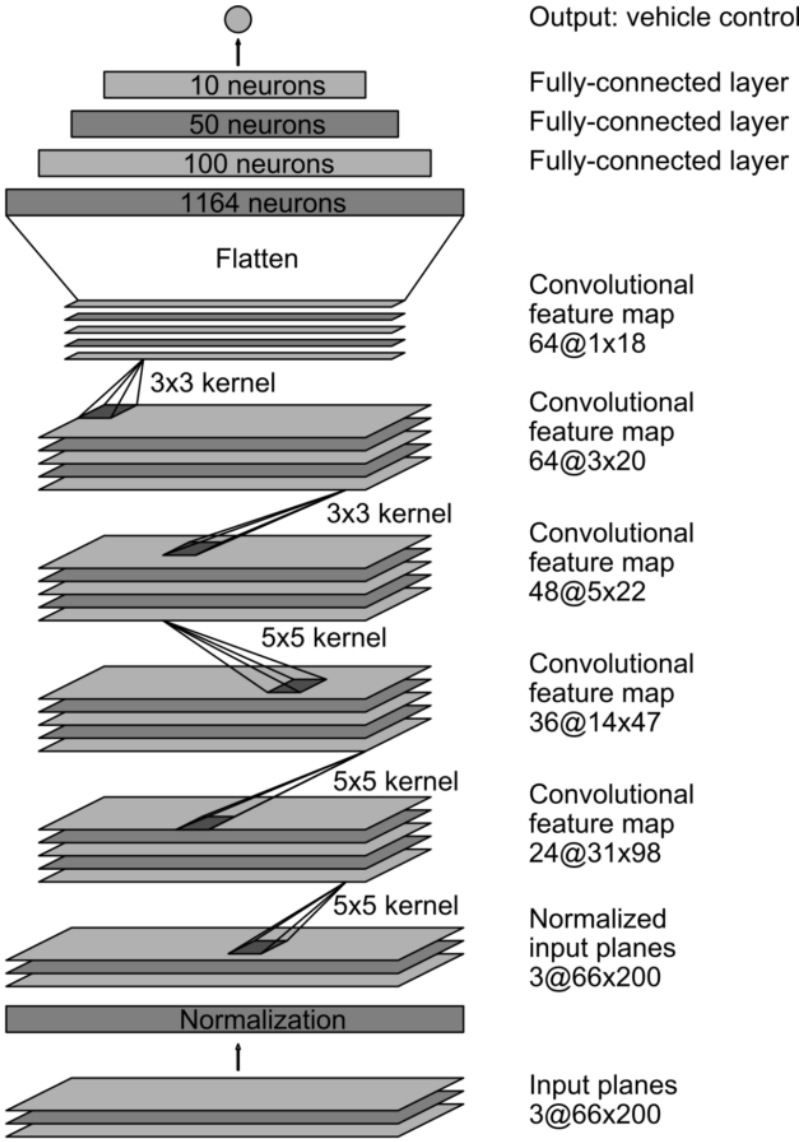



Figure 3.4: Model Breakdown

The model designed by Nvidia consists of 9 layers. There are 5 convolutional layers, 3 fully connected layers or dense layers and one normalization layer. The normalization layer is the processing of the data mentioned above and is not performed by the learning process. The first three layers have a strided 5X5 kernel with a 2x2 stride. The last two layers have a non-strided 3x3 kernel. The dot product of the kernel is calculated with the corresponding input values at each position. This feature allows the network to learn the edges or textures on the image.

## 3.3 Preliminary analysis

**Chassis:**

The rover chassis being used in our project is a metal body with two, gear reduction electric brushed motors. The motors are fitted with gears that have a belt attached to them for driving. The motors need a power of 100 watts to drive forward. The friction on the bearings along with the gears and the belt makes the motors draw more power to rotate. This will be fixed later on by using proper bearings and calibrating the dimensions of the vehicle.

**Mapping and Planning:**

The first step in an automated delivery robot's mapping process is gathering data about the environment. This is typically done using sensors, such as lidar or cameras. In this case, the robot is equipped with a camera connected to a Raspberry Pi. The camera captures images of the environment, which the Raspberry Pi then processes to create a map of the area. The robot will also use localization techniques, such as GPS, to determine its current position and update its map as it moves. The user will provide the end destination. The starting and end coordinates will be used as the input.

The next step in the mapping process is to use the information gathered by the camera to plan the robot's route. The Raspberry Pi uses the map of the environment to identify the most efficient path to the robot's destination. We will also use a routing engine called GraphHopper, which is based on a graph data structure and uses various algorithms to find the optimal route between two points. The API provides a variety of endpoints for calculating routes, searching for addresses, and retrieving information about the road network. It gives an output in the form of an XML file. Then the Raspberry Pi will extract relevant information from the images, such as the location of obstacles and the environment's layout along with navigation routes from the XML file. This information will then be used to instruct the robot to avoid obstacles and navigate around them to reach its destination as quickly and efficiently as possible. This allows the robot to adapt to environmental changes, such as new obstacles or closed roads, and to navigate efficiently.
As the robot moves, it communicates with a central system to receive updates on its delivery location and to update its map with new information. This allows the robot to plan its route and adapt to environmental changes quickly.

**AI Driving:**

The number of pre-trained data sets for the streets of Bangladesh is not widely available, hence we have to manually drive the vehicle and collect as much data as possible. Since the streets are busy we have to perform our experiment early in the morning. This means that there will be less traffic data on the model. The image obtained has to be processed according to the model that was discussed previously.

**Sensor limitations**

**Sonar:**

- Multiple reflections
- Cross-talk
- inflexible scanning methods

**GPS:**

- Not always accurate/reliable due to obstacles like large buildings and structures. GPS cannot penetrate solid walls or structures

# Chapter 4

# Implementation

## 4.1 Design

The chassis was repurposed from a different project. As seen in the image, the vehicle is made of solid steel and aluminium frame. The conveyor belt tracks are rolled over two gears. One gear is connected to the motor and another is free moving. The same is repeated on the other side consisting of two independent motors. Connected to the frame is a wooden base, where all the electronic components are housed. The motor and battery housing is located underneath the wooden base. A vertical wooden piece was attached to the base to secure the camera on top of it for the best possible results.

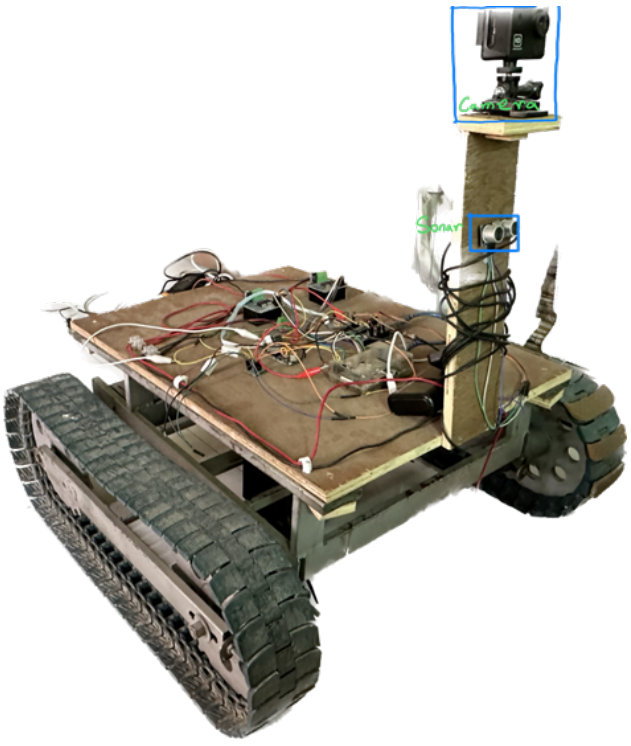



Figure 4.1: Front view

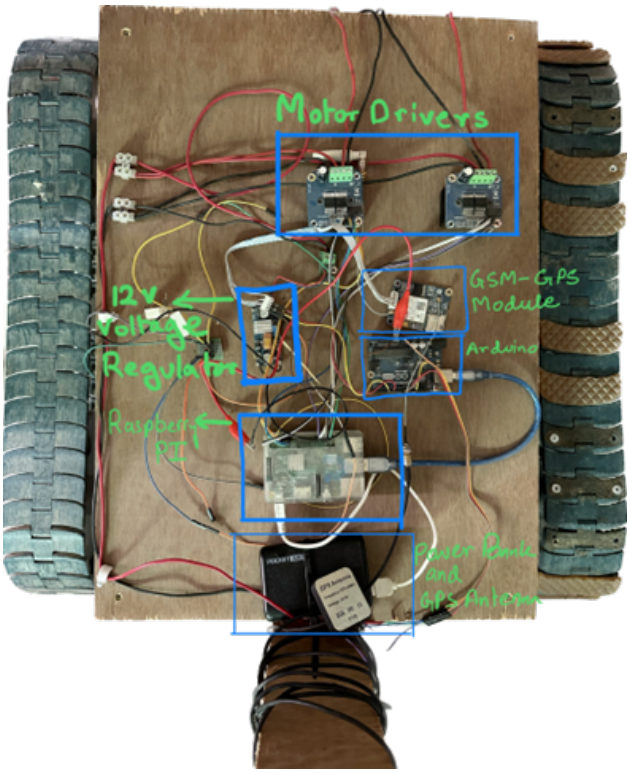



Figure 4.2: Labelled top down view

## 4.2 Technical Specifications

**Motor:** The motors are brushed gear reduction motors that consume 24 volts and 14 Amps of current. Typically used on bicycles these motors were the perfect choice for the vehicle.

**Motor Driver:** The motor drive being used to drive the motors is the "BTS7960". This is an H-Bridge High-Power Stepper Motor Driver Module. The motor driver can handle the high voltage and current required by the motors. The integrated driver IC, which has logic level inputs, current sense diagnosis, slew rate adjustment, dead time generation, and protection against overtemperature, overvoltage, undervoltage, overcurrent, and short circuit, makes it simple to interface with a microcontroller. The Double BTS7960 43A H-Bridge High-Power Stepper Motor Driver Module offers a highly space-efficient, cost-optimized solution for protected, high-current PWM motor drives.

**Voltage Regulator:** The XL4015 DC-DC Step Down power supply is needed to supply power to the GPS Module. Since the supply voltage from the battery ranges from 24-25 volts and the low-power modules require a much less voltage. The voltage regulator is a 180 KHz fixed frequency PWM buck (step-down) DC/DC module capable of delivering currents up to 5 Amps.

**Sonar:** The HC- SR04 Sonar was used for distance measurement. This is a cheap way to measure the distance from the rover to the obstacles. The range of this sonar is 20cm to 400cm which is good enough for our use case on the vehicle.

**Arduino and GPS Module:** The GPS module was used to find the position of the vehicle. Since we have used sonar, all the devices were connected to Arduino which was used to establish communication between the GPS sonar and the raspberry pi.

**Camera:** The camera used is a generic computer camera connected to the raspberry pi using a USB cable.

**Batteries:** The vehicle has two power systems. Two lead acid batteries of 12Volts are paired up in series to make the 24 volts required for the motor. A powerbank is used to power the raspberry pi. The reason for two separate power sources is that the lead acid batteries drive the motors and lose their charge very fast. If the batteries lose enough charge the raspberry pi shuts down on its own which results in loss of data. Hence, a separate power supply is used to power the raspberry pi for reliability.

**DUALSHOCK 4 Controller:** The PS4's DualShock 4 controller was used to manually control the rover for data collection and maneuvering. Using the analog stick of the controller gives a value from -1 to +1 which indicates the direction and the angle of the turn. The x button is configured for the rover to go forward, and the "O" button moves the vehicle in reverse. The analog stick on its own rotates the vehicle and when going forward the analog stick moves the vehicle to the right or left.

**Hardware Connections and onboard computer:** The raspberry pi was used as the main computing unit of this entire project. The pi directly controlled the motor drivers which provided power to the motors. The Arduino was used to interface with the GPS module and Sonar. The calculations and data modulation of the sonar was done on the Arduino. The Arduino is connected to the raspberry pi via USB where the data is transmitted between the Arduino and PI using serial communication. The 12Volt buck converter is used to power the GPS module. Raspberry Pi was powered using the powerbank and the camera module connected to the raspberry pi via USB where the power is taken directly from the raspberry pi over USB.

# Chapter 5

# Methodology

## 5.1 Data Collection

**Choosing the correct place to train:** To train the vehicle, a place with proper asphalt road with buildings and light traffic was chosen. These criteria are based on the fact that the vehicle would be navigating a neighbourhood as the main objective of this vehicle is to be the last mile of the delivery hub. For this, a neighbourhood in Uttara was chosen with relatively less traffic and a well-paved road. Although this is a quiet neighbourhood, there were times fast driving cars and rickshaws posed threats to the vehicle and the person driving the vehicle.

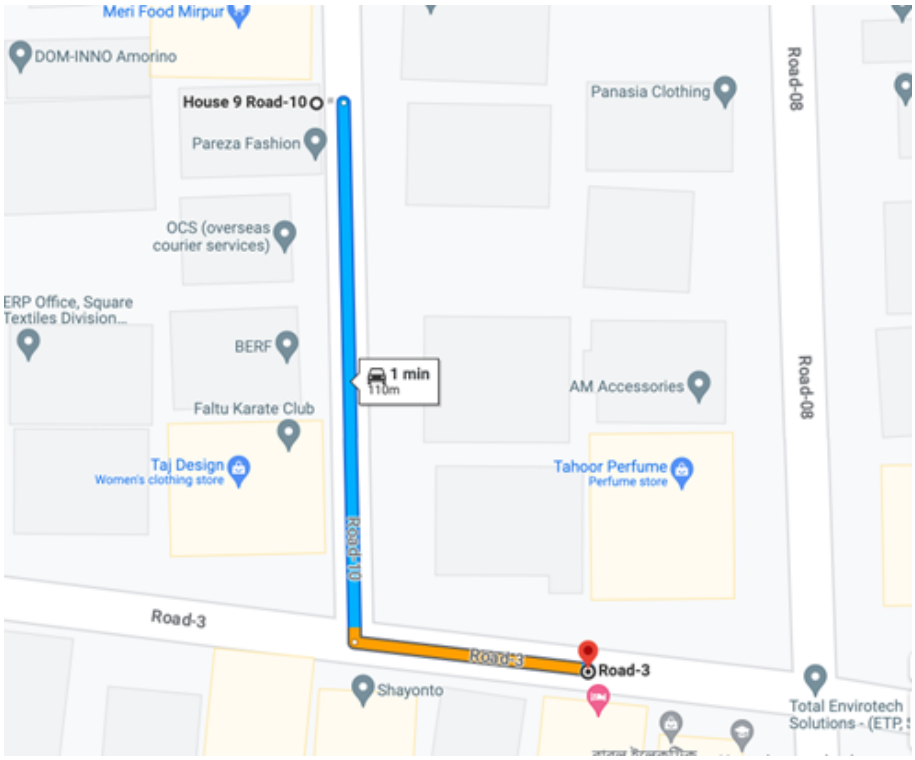



Figure 5.1: Training Route

To train the vehicle, the rover was driven manually using the controller on the map shown above. The vehicle was driven for a total of 2 laps throughout the road, and the Raspberry Pi program captured the images along with their corresponding steering data.

**Data Storage:** Each image collected from the camera has to be given a unique name. This is why the time and date were extracted from the Raspberry Pi system and set as the name of the image. The program utilizes the pandas library to store this information along with the steering data. The OpenCV library was used to capture the images from the camera, and they were converted to a resolution of 240x120 pixels. The image path and the corresponding steering value were stored in the .csv file, which was used for the training process.

Pressing the button on the controller would start the data collection process. Once we have completed the necessary number of laps on the street, pressing the button again would save the data to the Raspberry Pi. Each time this process is performed, the program creates a new folder and stores the necessary images and the CSV file containing the steering values.

| | | |
|---|---|---|
| 96 | /home/srzavin/main_files/version_1/DataCollected/IMG6/Image_16763626067148.jpg | 0 |
| 97 | /home/srzavin/main_files/version_1/DataCollected/IMG6/Image_1676362606993506.jpg | 0 |
| 98 | /home/srzavin/main_files/version_1/DataCollected/IMG6/Image_1676362607271619.jpg | 0 |
| 99 | /home/srzavin/main_files/version_1/DataCollected/IMG6/Image_1676362607550292.jpg | 0 |
| 100 | /home/srzavin/main_files/version_1/DataCollected/IMG6/Image_1676362607826693.jpg | 0 |
| 101 | /home/srzavin/main_files/version_1/DataCollected/IMG6/Image_1676362608105864.jpg | 0 |
| 102 | /home/srzavin/main_files/version_1/DataCollected/IMG6/Image_1676362608381362.jpg | -0.05 |
| 103 | /home/srzavin/main_files/version_1/DataCollected/IMG6/Image_167636260865789.jpg | -0.26 |
| 104 | /home/srzavin/main_files/version_1/DataCollected/IMG6/Image_1676362608933721.jpg | -0.43 |
| 105 | /home/srzavin/main_files/version_1/DataCollected/IMG6/Image_1676362609211765.jpg | -0.38 |
| 106 | /home/srzavin/main_files/version_1/DataCollected/IMG6/Image_1676362609490413.jpg | -0.65 |
| 107 | /home/srzavin/main_files/version_1/DataCollected/IMG6/Image_1676362609769366.jpg | -0.93 |
| 108 | /home/srzavin/main_files/version_1/DataCollected/IMG6/Image_1676362610047194.jpg | -1 |
| 109 | /home/srzavin/main_files/version_1/DataCollected/IMG6/Image_1676362610324182.jpg | -1 |
| 110 | /home/srzavin/main_files/version_1/DataCollected/IMG6/Image_1676362610600957.jpg | 0 |
| 111 | /home/srzavin/main_files/version_1/DataCollected/IMG6/Image_1676362610878486.jpg | 0 |
| 112 | /home/srzavin/main_files/version_1/DataCollected/IMG6/Image_1676362611157524.jpg | 1 |
| 113 | /home/srzavin/main_files/version_1/DataCollected/IMG6/Image_1676362611437214.jpg | 0 |
| 114 | /home/srzavin/main_files/version_1/DataCollected/IMG6/Image_1676362611715783.jpg | 0.21 |
| 115 | /home/srzavin/main_files/version_1/DataCollected/IMG6/Image_167636261199425.jpg | 0 |
| 116 | /home/srzavin/main_files/version_1/DataCollected/IMG6/Image_1676362612272619.jpg | 0 |
| 117 | /home/srzavin/main_files/version_1/DataCollected/IMG6/Image_1676362612551655.jpg | 0 |
| 118 | /home/srzavin/main_files/version_1/DataCollected/IMG6/Image_1676362612831351.jpg | 0 |
| 119 | /home/srzavin/main_files/version_1/DataCollected/IMG6/Image_1676362613113476.jpg | 0 |
| 120 | /home/srzavin/main_files/version_1/DataCollected/IMG6/Image_1676362613394682.jpg | 0 |
| 121 | /home/srzavin/main_files/version_1/DataCollected/IMG6/Image_1676362613677007.jpg | 0 |

Figure 5.2: Raw Data for Training

**Future work:** In the future, we could add many other cameras, braking values, speed and throttle control, and other parameters to train the model and get better results.

# Chapter 6

# Model Training

As a vision-based autonomous bot, the primary functions of the system is greatly dependent on the data collected through a single front-facing camera. A CNN was trained to map raw pixels from the camera directly to steering commands. A CNN, or Convolutional Neural Network, is a type of deep learning model designed for analyzing visual data such as images and videos by image classification, object detection, image segmentation and other computer vision tasks. CNNs apply filters across the input data to produce feature maps that highlight important patterns. The system learns to drive in traffic on roads and highways and even in areas with unclear visual guidance with minimum provided training data. The implemented CNN learns the whole processing pipeline needed to steer a vehicle. The project is greatly inspired by a project done in "End to End Learning for Self-Driving Cars" by NVIDIA Corporation in 2016 (Bojarski et al., 2016). Training the CNN model is done in 4 major steps which are data collection, preprocessing, enhancing data and training-testing.

## 6.1 Data Collection

The primary source of data was used for the project. The data was collected as a video format through a single-mounted [camera model] unit. Then each frame of the video was converted to image format and stored in the destination folder. Along with the visual data, the steering angle of the vehicle was collected. The location of each image and the steering angle derived from them were stored in a CSV format file. The file was then imported to the programmed CNN model for data initialization.

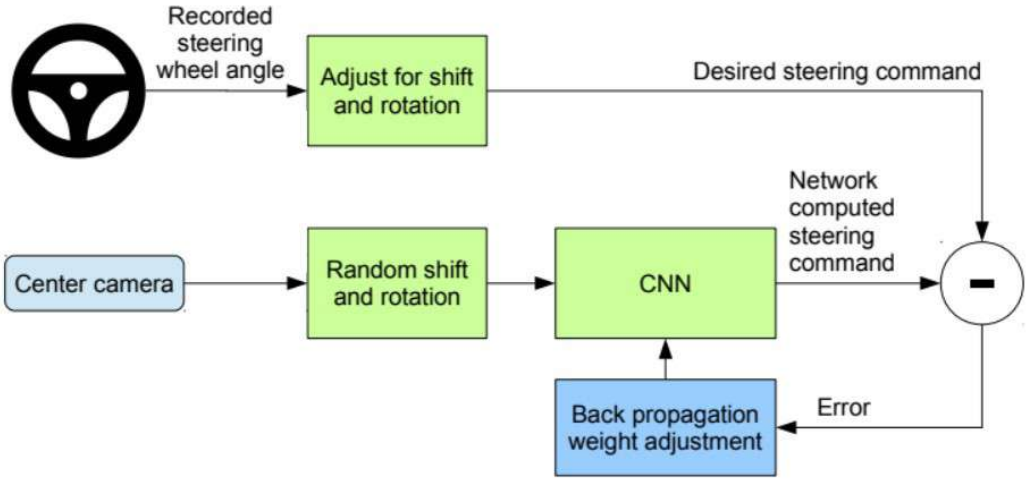



Figure 6.1: System Structure

## 6.2 Preprocessing

Firstly, the CSV file with the image's locations and steering angle is imported into the model. The model loads the data column-wise. For this task, the read_csv() function of the panda's library of Python programming language was used. Through this process, the CSV file is DataFrame format which makes it eligible for the model to process effectively.

Secondly, the imported data might be imbalanced which can rise problems training process. For example, in the collected data the left curve of the vehicle is present in great quantity than the right curve. If the data is not balanced and corrected then the training process will prioritize the left curve more than the other and may produce faulty results. So, the model balances the data so that the training process is not biased. To do this, the model sets a cutoff value and any value exceeding that cutoff limit is excluded from training data. This method reduces the redundant and similar quality of data from the dataset which results in a better-balanced quantity of data.

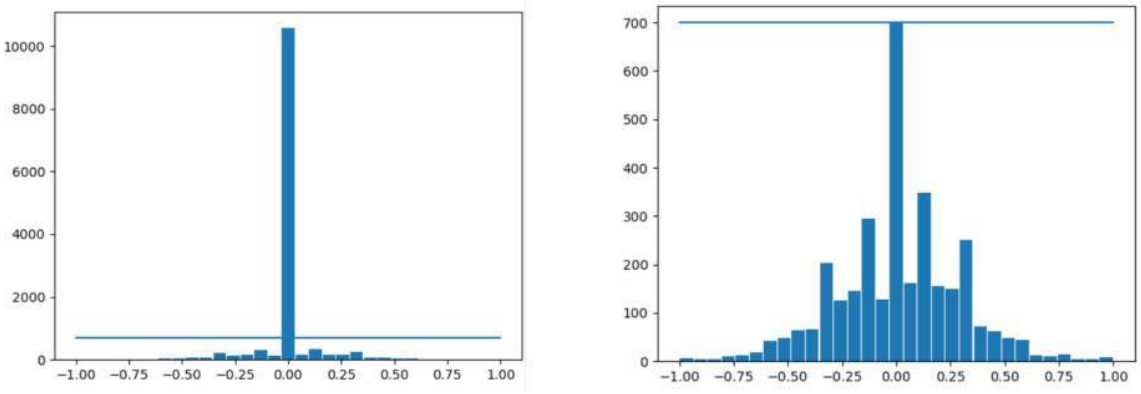



Figure 6.2: Before and after data balancing

Lastly, the model prepares the data for processing and split it up into training and testing data. Once the data is balanced, the model splits the data for training and validation. The training data is for the model to train itself and the validation data is used to test the performance of the model after each epoch. So, each time the model completes one training it will test the performance of the training with validation data and then will train again. To split the data the model uses the Python programming language's Scikit-learn library also known as sklearn. Using the sklearn.model_selection package's train_test_split function the model splits the whole data to an 80:20 ratio for training and validation. This means the model uses 80% of the data to train and 20% of data to test the performance of the training.

## 6.3 Enhancing Data

Since the data is divided into training and validation sets, the model will now perform augmentation on the training data set. The augmentation process will change the image data slightly from the original one and store it as new data. Generating new transformed versions of images from the given image dataset increase the dataset's diversity which benefits the training process. The augmentation performs zooming in, shifting left or right which is referred to as panning and changing the brightness and contrast. Doing these will create an altered version of the original image and include it in the training dataset. Thus enriching the quantity of training data. The model uses imgaug python library to do these augmentation tasks.

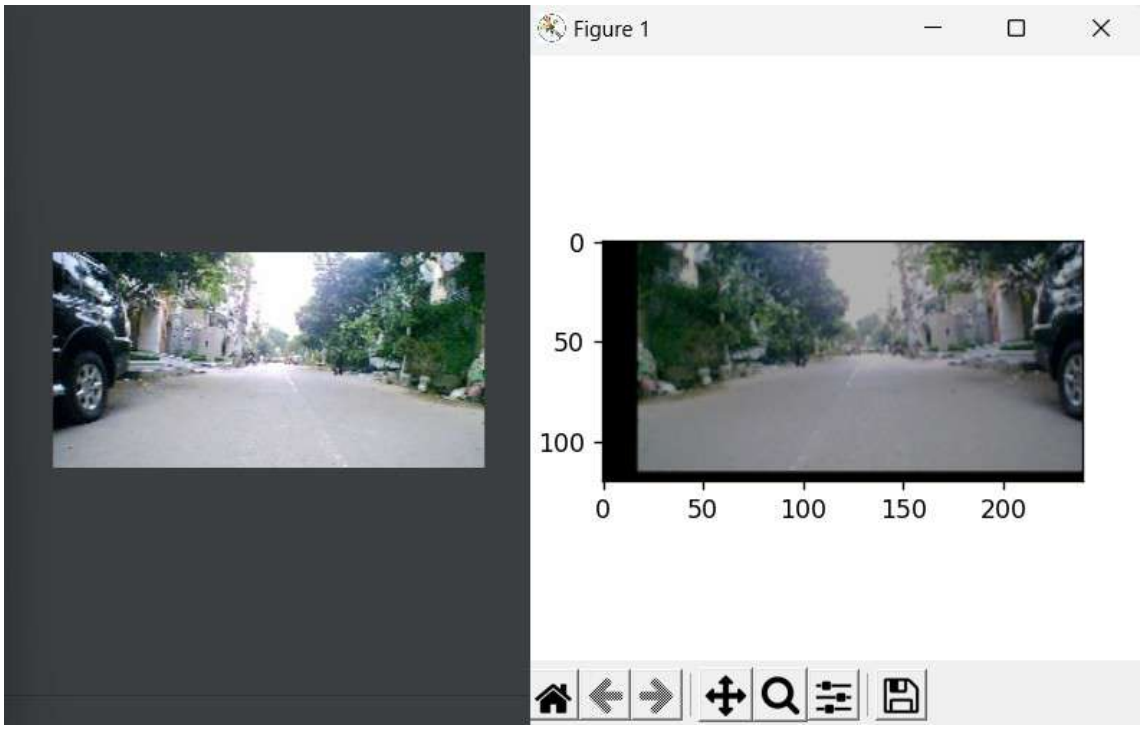



Figure 6.3: Before and after performing Augmentation on primary data

After the augmentation process, the model will now enhance the updated training data. First, the model will crop the original image data from 60 to 135 pixels. The reason for doing this is to remove unnecessary objects and terrains from the vehicle's view.


Figure 6.4: Cropping the original image data

Now the model has the image data which only contains the lanes and nearby objects. For the next step, the model will now change the colorspace of the cropped image data for simplified visual representation. The model changes the colorspace from RGB to YUV. The YUV is a color model for the color image pipeline. Its greatly efficient for encoding images or videos as it takes human perception into account (Microsoft Corporation, 2021). The end result of this colorspace transformation is more generalized image data for the model to train.

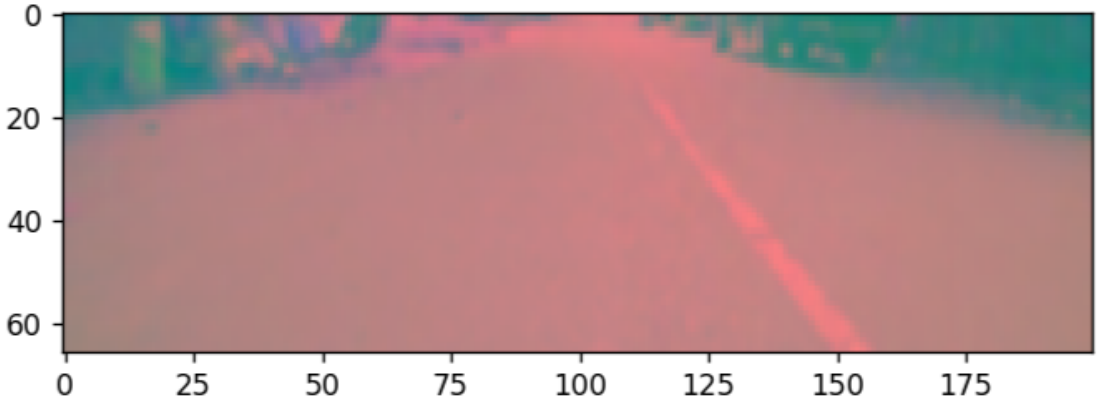



Figure 6.5: Changing colorspace to YUV

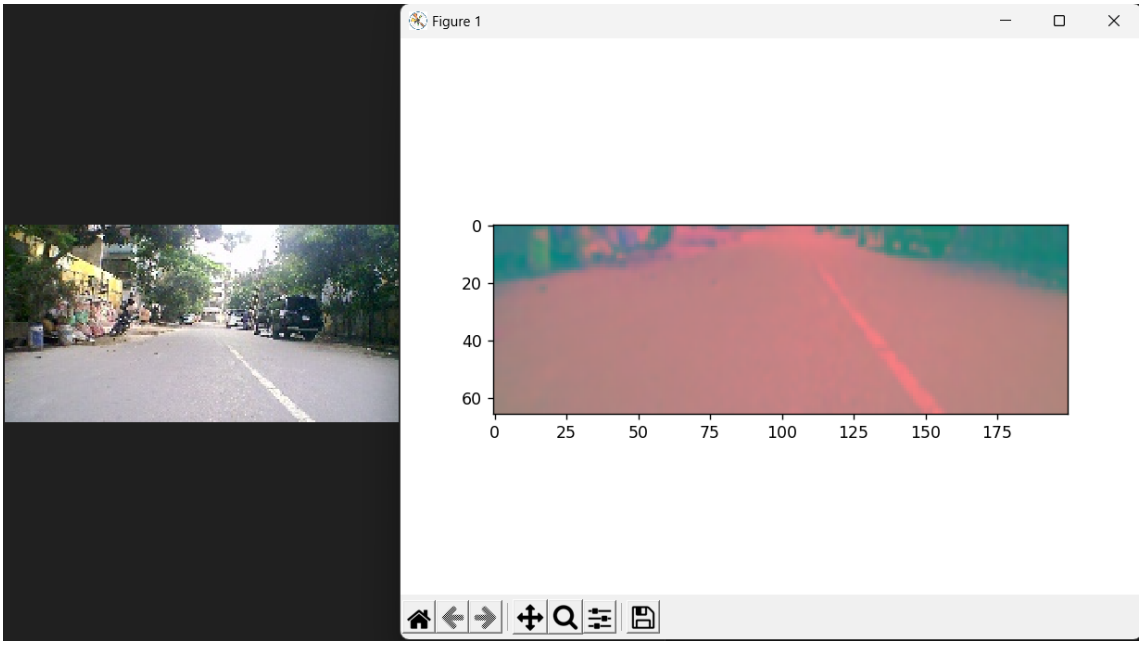



Figure 6.6: From original image data to finalized image data

As now the model is done preprocessing training data completely, it will work on batch generation. The idea of batch generation is that, the model does not send all of its data altogether into training. Rather, it sends the data into batches. The model creates 100 images batch and sends them simultaneously into training. However, the model does not include validation image data while making batches as the original validation data needs to stay unchanged for testing properly.

## 6.4 Training and Testing

After all the preprocessing tasks are completed, the data is fully ready to be sent for training. The training model is of 9 layers. 5 of them are convolutional layers, 3 of them are dense layers and one normalization layer. The Sequential model was used for training. The sequential model is an appropriate model while working on plain stack of layers where each layer has exactly one input and output tensor (TensorFlow, 2022). As the created CNN model uses data batches one at a time to train and also follows layer architecture, the sequential model is the effective one for training. The tensorflow.keras.model package of Python was used for Sequntial model and tensorflow.keras.layers was used for implementing convolution, flatten and dense layers.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 31, 98, 24)        1824
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 14, 47, 36)        21636
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 5, 22, 48)         43248
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 3, 20, 64)         27712
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 1, 18, 54)         31158
_________________________________________________________________
flatten (Flatten)            (None, 972)               0
_________________________________________________________________
dense (Dense)                (None, 100)               97300
_________________________________________________________________
dense_1 (Dense)              (None, 50)                5050
_________________________________________________________________
dense_2 (Dense)              (None, 10)                510
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 11
=================================================================
Total params: 228,449
Trainable params: 228,449
Non-trainable params: 0
```

Figure 6.7: Model Summary

The model creates 100 images batchs and send them to the training process where each epoch have 300 steps. So, each epoch generates 100X300 images for training. And the model trains like this for 10 times, so the total number of epoch is 10.

```
----------------------------------------------------------------
Epoch 1/10
300/300 [==============================] - 430s 1s/step - loss: 0.4405 - val_loss: 0.5183
Epoch 2/10
300/300 [==============================] - 389s 1s/step - loss: 0.4179 - val_loss: 0.6010
Epoch 3/10
300/300 [==============================] - 355s 1s/step - loss: 0.3947 - val_loss: 0.5929
Epoch 4/10
300/300 [==============================] - 320s 1s/step - loss: 0.3718 - val_loss: 0.6707
Epoch 5/10
300/300 [==============================] - 325s 1s/step - loss: 0.3491 - val_loss: 0.6102
Epoch 6/10
300/300 [==============================] - 322s 1s/step - loss: 0.3253 - val_loss: 0.5339
Epoch 7/10
300/300 [==============================] - 322s 1s/step - loss: 0.2898 - val_loss: 0.5568
Epoch 8/10
300/300 [==============================] - 318s 1s/step - loss: 0.2657 - val_loss: 0.5845
Epoch 9/10
300/300 [==============================] - 324s 1s/step - loss: 0.2380 - val_loss: 0.5767
Epoch 10/10
300/300 [==============================] - 323s 1s/step - loss: 0.2161 - val_loss: 0.5881
Model Saved
```

Figure 6.8: Training complete

In the training process the loss in training and the loss in validation are decreasing in each following epoch, which is considered good result as the training is producing more and more accurate results.

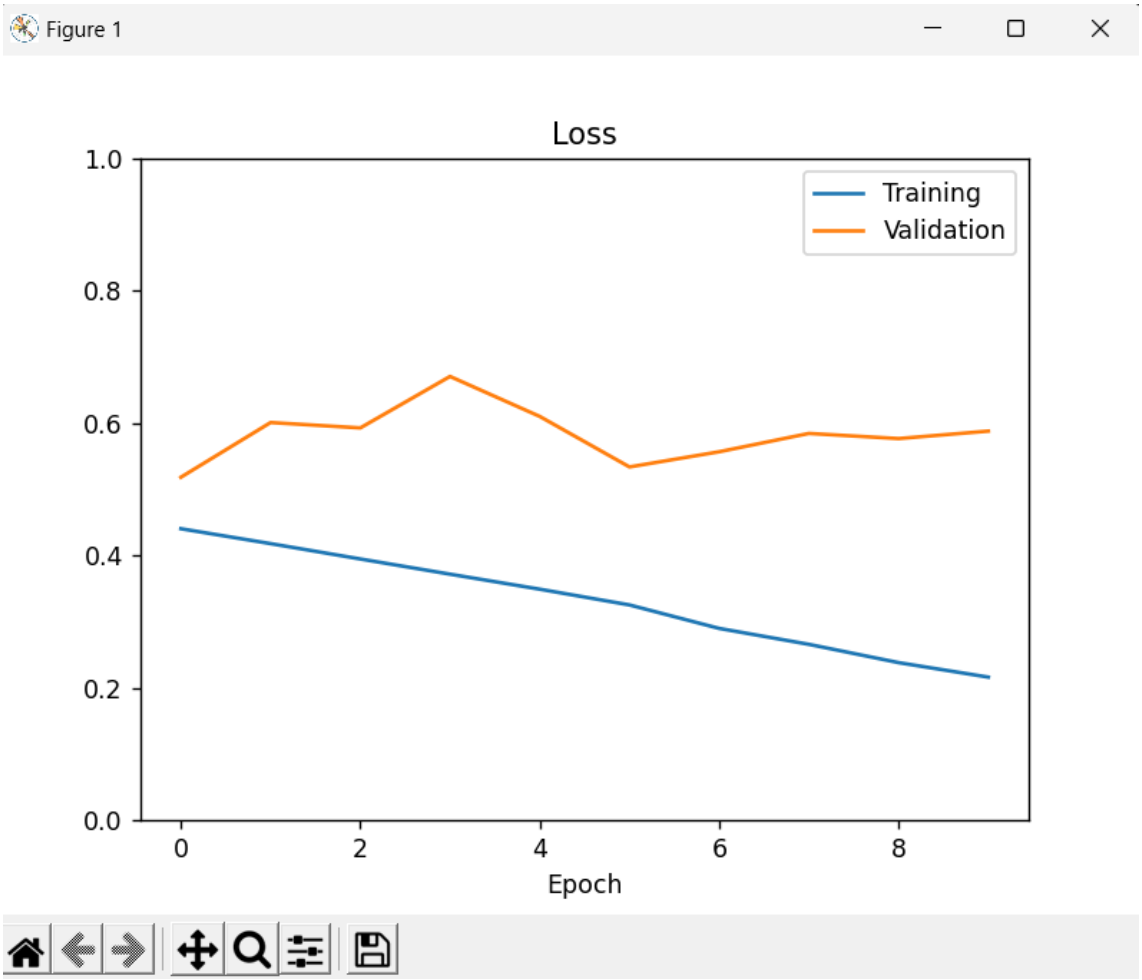



Figure 6.9: Graphical representation of loss in training and validation

As the training is complete the program stores the trained model in a Hierarchical Data Format file (.h5). Loading this saved model.h5 file the automobile can run autonomously.

# Chapter 7

# Path Planning

## 7.1 Introduction to Path Planning:

The importance of path planning cannot be overstated when it comes to the effectiveness of autonomous systems such as delivery robots, and locating the best way to travel from one location to another is necessary. Due to the presence of several variables like time and potential obstacles, this is a challenging task. However, the role of the path planning algorithm is crucial in maintaining safety as well as efficiency when it comes to helping robots navigate through an environment autonomously for applications such as a delivery system by avoiding obstacles and traveling within minimal time.

## 7.2 The A* Algorithm:

The effectiveness and precision of the A* Algorithm make it a choice method for many when finding pathways, and the informed searching method of this algorithm relies on heuristics for guidance. The initial starting position serves as an anchor for constructing an array of paths in the form of a tree with every subsequent node representing another possible route. To come up with a nodes's total cost we first add up the expenses associated with all edges going towards it and then estimate additional costs needed to reach our objective. With a continuous focus on picking nodes with limited costs to expand upon and repeating this technique until reaching its destination node - this is how the algorithm works. This approach ensures that the first time the destination node is selected for expansion, the path found to it is the shortest possible.

## 7.3 Modified A* Algorithm in GraphHopper :

For the purposes of our research study, we made use of a modified version of the A* algorithm as implemented in GraphHopper. The modified version runs almost

identically to the original A* algorithm except that it factors in a new variable: how many times certain edges have been used on previous paths. The goal of this alteration is to improve the efficiency of the autonomous delivery robot's navigation by steering clear of likely crowded or troublesome routes, which is especially valuable in urban settings where gridlocks have a severe effect on how quickly robots can deliver.

## 7.4 The Alternative Route Algorithm :

The alternative route can be found by using a modification to the A* algorithm, which allows one to generate various alternative routes between an origin and a destination. At first, this algorithm detects the most efficient path from point A to point B. The generation of supplementary paths is done keeping in mind avoidance of congested localities and probable hindrances along with maintaining resemblance with the shortest path. The algorithm considers many different factors including the number of turns taken into account when choosing an optimal alternative route. Balancing both optimality and diversity in approach allows the robot to have multiple viable routes hence increasing its resilience to dynamic environmental changes.

## 7.5 Advantages and Disadvantages of the Alternative Route Algorithm

The alternative route algorithm offers several advantages: · The system generates different paths between 2 specific locations which increases flexibility and adaptability in robots' navigational methods. Dynamic establishments find this particularly advantageous when the conditions change rapidly. · Steering clear of high-traffic areas or any other potential problems increases the efficiency and safety of the robot. · By considering different factors and variables an optimal alternate route is selected to optimize the robot's path planning.

Nevertheless, some limits should be taken into account while using the alternative route algorithm. Additionally, the additional computational requirements for generating alternate routes can make it slower than when using the shortest route algorithm. Complex environments with a lot of obstacles can pose challenges when trying to find a route. Furthermore, it may only sometimes identify the best alternative route, especially in dynamic environments where conditions change rapidly.

# Chapter 8

# Lane Detection

While autonomously driving, the chassis finds the path using color detection or edge detector. It gets the curve using the summation of pixels in the y direction which can also be referred to as a histogram. The chassis's system will first detect the path and then find the curve present in the road. Then the system will send commands to the chassis's motors based on a found curve. The lane detection task is completed in 5 steps which are Thresholding, Warping, Creating Histogram, Averaging, and Displaying. The OpenCV library of Python is being used in the majority of the tasks.

## 8.1 Thresholding

The chassis gets the path using Color Detection. The roads and its surroundings have different textures and color depth. The system will differentiate the lane by taking only the color of the road. For this, the system will convert the image to HSV colorspace. A ColorPickerScript gives us the range of values that the system needs to use to find the lane. After the values are received, the thresholding task continues which detects the lane and other terrains.

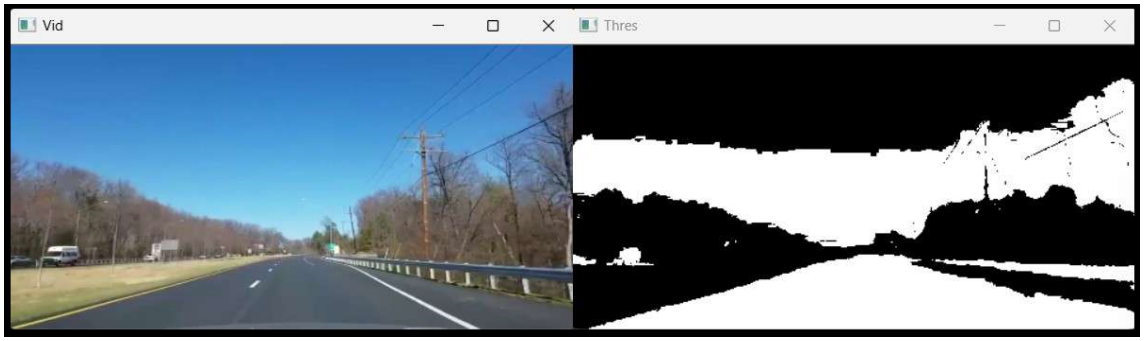



Figure 8.1: Image data thresholding

## 8.2 Warping

As the chassis won't need to process the whole image because it just needs the curve on the path right at the moment, so it simply crops the image, but this is not enough since it needs to look at the road as if it were watching from the top. This is known as a bird eye view and it is important because it will allow the chassis to easily find the curve. The system will achieve this through the warping process. To warp the image the system will define the initial points. The idea is to get a rectangle shape when the road is straight. The program uses the warpPrespective function of the OpenCV libraby to warp the image.


Figure 8.2: Different trackbars for different initial points

Figure 8.3: Processed images according to different warping initial points

After the warping process, the chassis will now have a bird eye view of the lane only. Thus, it can easily find the curve when appeared.

## 8.3 Creating Histogram, Averaging and Displaying

After the warping process, the chassis now has a binary image i.e. it has either black or white pixels. Now the program can sum the pixel values in the y direction. The reason for doing this is that the program finds the curve through the summation of pixels.

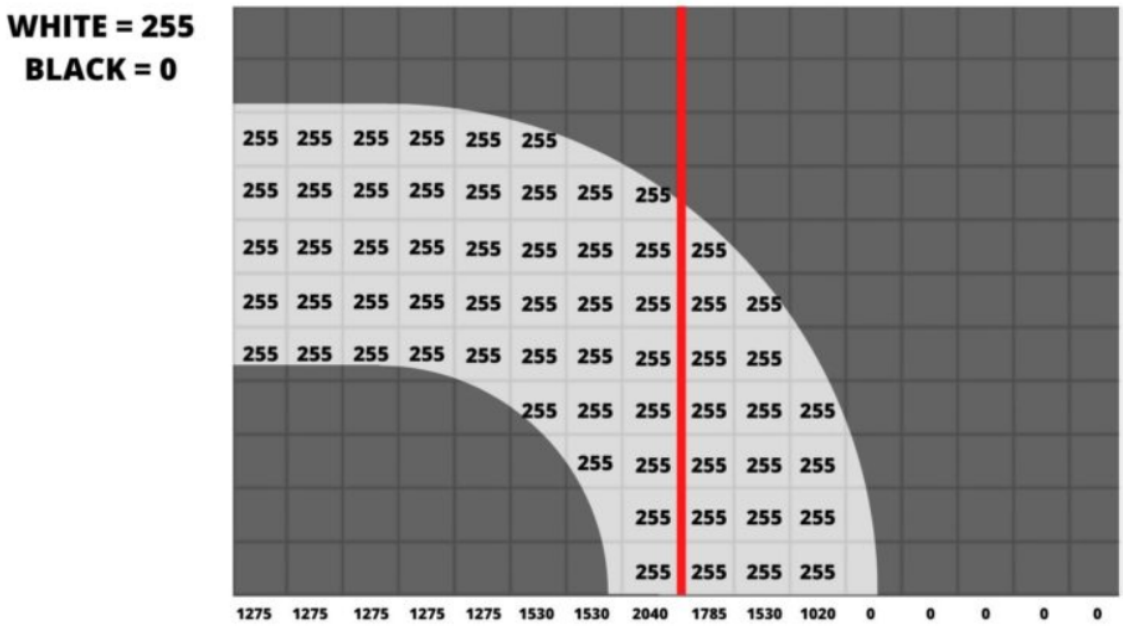



Figure 8.4: Pixel Summation

The picture above shows all the white pixels with 255 values and all the black with 0. Here, all the pixels in the first column together sum up to 1275. This method is applied to each of the columns. In the binary image data above, it has 240 pixels in width. Therefore the chassis will have 240 column values. After summation, the program will look at how many values are above a certain threshold on each side of the center red line. In the above example, there are 8 columns on the left and 3 columns on the right. This means the curve is towards the left. While doing pixel summation the program also keeps the center line in check. If the program uses a fixed center line in every case then it will result in fault detecting curve.

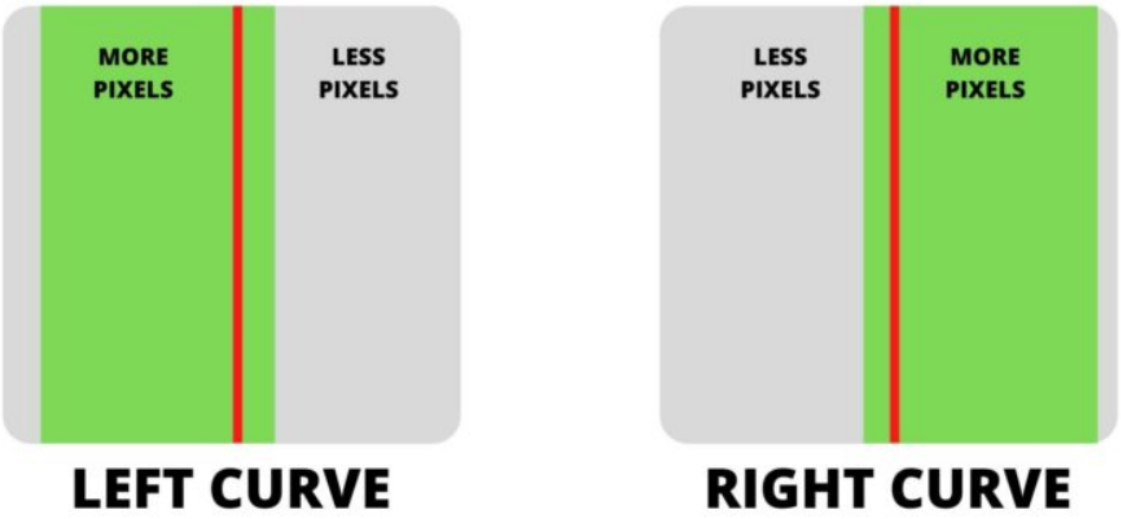



Figure 8.5: False curve detection dew to wrong placing of center line

The program solves this problem by working on the histogram first. The program has 240 values from 240 columns. Now few pixels in the image might just be noise and are not needed in the calculation. Therefore the program sets a threshold value acting as the minimum value required for columns to qualify as part of the path and not noise. Thus excludes the noises from the calculation. The calculated maximum sum value is multiplied by the user-defined percentage (0.5) to create the threshold value. Now, if the curve is left the program needs to know how much the curve is leaning left. To get the curvature value, the program finds the indices of all the columns that have a value more than the calculated threshold and then will average the indices. Meaning that, if the indices of the pixels started from 40 and ended at 310, our average would be ((310-40)/2) + 40 = 175. The average base point is drawn for better visualization.

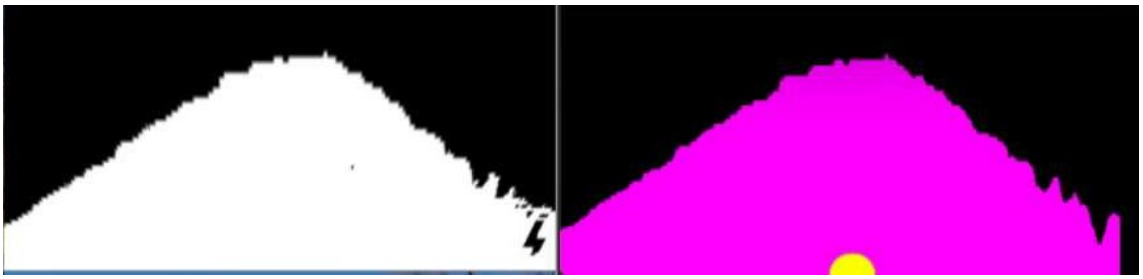


Figure 8.6: Histogram (Right) of the Warp Image (Left) with average base point (Yellow Dot)

For the next part, the program needs to find the middle or center of the histogram. But this time the program will apply the histogram technique only on the bottom 1/4 part of the image. This is because the average of the base is needed so there is no need to average the pixels above 1/4 the image. After finding the middle value, the program will find the difference between the average base point found at the first calculation and the middle value found now. For our 240-pixel image, if the curve is on the left side then the average base point value found in the previous calculation will be less than the middle value. So the difference between them will be negative.


Figure 8.7: Detecting curves (Negative for left and positive for right curve)

The chassis completes all the processes explained above for each of its image data to successfully detect lanes and perform turns on curves.

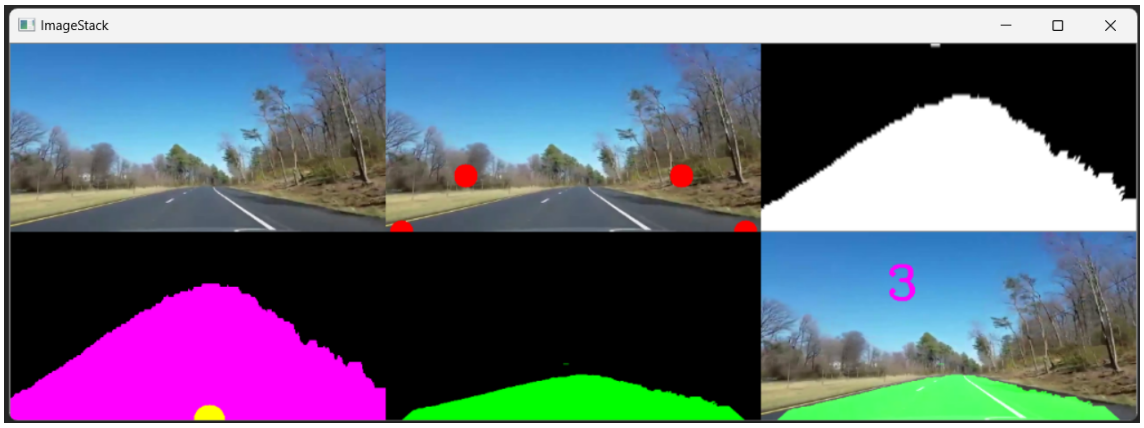



Figure 8.8: Complete process of lane detection

# Chapter 9

# Object Detection

## 9.1 Introduction

The object detection mechanism is a crucial subsystem that helps the autonomous delivery robot sense and perceive its surroundings. This module allows the robot to recognize obstacles in its path, such as vehicles, pedestrians, animals or other objects and autonomously decide on evasive actions or an alternative route that adapts to avoid collisions and ensure safe navigation. With this enhanced perception, the bot is capable of mitigating dangers in real-time guaranteeing the safety of the bot and the people around it. For accurate navigation and localization, this subsystem's data is important, especially in challenging and complex situations such as densely populated places where GPS signals are unreliable and so by employing this data, the robotic system can modify its mapping of surroundings and determine which way to navigate. Therefore, this capacity effectively increases the bot's adaptability, dependability, and autonomy while facilitating interactions between the bot and the environment, ensuring correct and effective delivery.

## 9.2 Object Detection Tools

In implementing object detection for our study, we relied on the OpenCV library, which has become highly popular in the field of real-time computer vision as it provides an open-source solution. The bot we designed has an object detection subsystem implemented using a Python script utilizing the OpenCV library using a pre-trained model, a camera sensor. The pre-trained model in question is the SSD MobileNet V3 Large model which is a lightweight convolutional neural network architecture designed for mobile and embedded devices. This model has been trained using a combination of deep learning techniques, including convolutional neural networks (CNNs) to learn discriminative features and object detection algorithms like anchor-based methods that use anchor boxes to detect and localize objects in an image and match them against the known objects of the COCO dataset and relaying the name of the object with the highest degree of resemblance.

## 9.3 Advantages of SSD MobileNet V3 Large Model

This model is used because it contributes:

- **Generalization:** The model is trained on a large-scale dataset COCO (Common Objects in Context) which contains a wide range of object categories ranging from animals, vehicles, people and everyday object images from various sources captured in different contexts and environments. This ensures that models trained in COCO can handle different lighting conditions, viewpoints, and backgrounds.
- **Ease of Use:** The availability of pre-trained models like SSD MobileNet V3 Large simplifies the development process saving time and effort required for collecting and labelling a large training dataset to design and train a custom model.
- **Performance:** The model was designed to strike a balance between computational efficiency and accuracy while being lightweight allowing it to be suitable for implementation into real-time applications on resource-constrained mobile devices.
- **Speed:** The model utilizes a single-shot detection approach which means it can detect objects in a single pass through the network. This design enables faster inference times compared to models that require multiple passes or more complex computations.
- **Community support:** This popular model has a wide user-base and community that supports the development and improvement of the model. This also means there are resources, tutorials, and community discussions available to help developers troubleshoot issues, optimize performance, and explore best practices.

## 9.4 Disadvantages of the SSD MobileNet V3 Large Model

The drawbacks of using this model are:

- **Limited generalization to unseen objects:** The COCO dataset on which the model is trained may not classes for unknown objects specially when detecting objects from a different domain or context restraining the performance of the model. The model could also be impacted by false positives or negatives as a result of this factor.
- **Trade-off between speed and accuracy:** This model is designed to prioritize speed and efficiency over ultimate accuracy. While it provides real-time inference, it may not achieve the same level of accuracy as larger, more computationally expensive models.

- **Sensitivity to input image quality:** This model is more sensitive to variations in input image quality meaning it may struggle to handle low-resolution or fine-grained details, leading to reduced performance in such scenarios. The model also has a difficulty of detecting small objects or objects that are heavily occluded.
- **Limited flexibility for model customization:** As the model is pre-trained, its architecture and parameters are fixed. This means there is limited control over the underlying architecture and adjusting some aspects is just not possible.

For future work, a custom model will be made by training with a custom dataset suited for specialized domains which in this case is the streets of Bangladesh and fine-tuning parameters to optimize performance.

## 9.5 Code Implementation

For the delivery bot, the Python script sets up importing necessary libraries, such as ‘cv2‘ for computer vision operations. A confidence threshold for object detection of 60% is assigned, and an error threshold of 85% is assigned. The determination of whether or not an object is valid is dependent on whether or not its confidence rating meets or exceeds the defined threshold. Variables are initialized for capturing video frames from the camera sensor, loading class names, and defining a boundary line on the frame. Each video frame has a width of 1280 and a height of 720, and the brightness of each frame is set to 70.

The pre-trained SSD MobileNet V3 Large model is loaded using the `cv2.dnn_DetectionModel` class with its configurations and weights. Class names are loaded from the `coco.names` file, and the model configurations are loaded from the `ssd_mobilenet-v3_large_coco_2020_01_14.pbtxt`.

The code enters a loop where video frames are continuously read from the camera sensor. Object detection is performed on each frame using the loaded model, and detected objects are visualized by drawing bounding boxes or anchor boxes and labels on the frame. A red boundary line is also drawn on each frame so that when an object crosses the boundary line, an ”ALERT” text is displayed to establish that the object is approaching the bot.

When objects are detected, the code loops over the detected objects and draws bounding boxes and labels on the output frame, meaning that the class name, confidence score, and an ”ALERT” text are displayed near objects. This real-time feedback is crucial for monitoring the performance of the object detection system and identifying any potential issues.


Figure 9.1: Object Detection in open roads

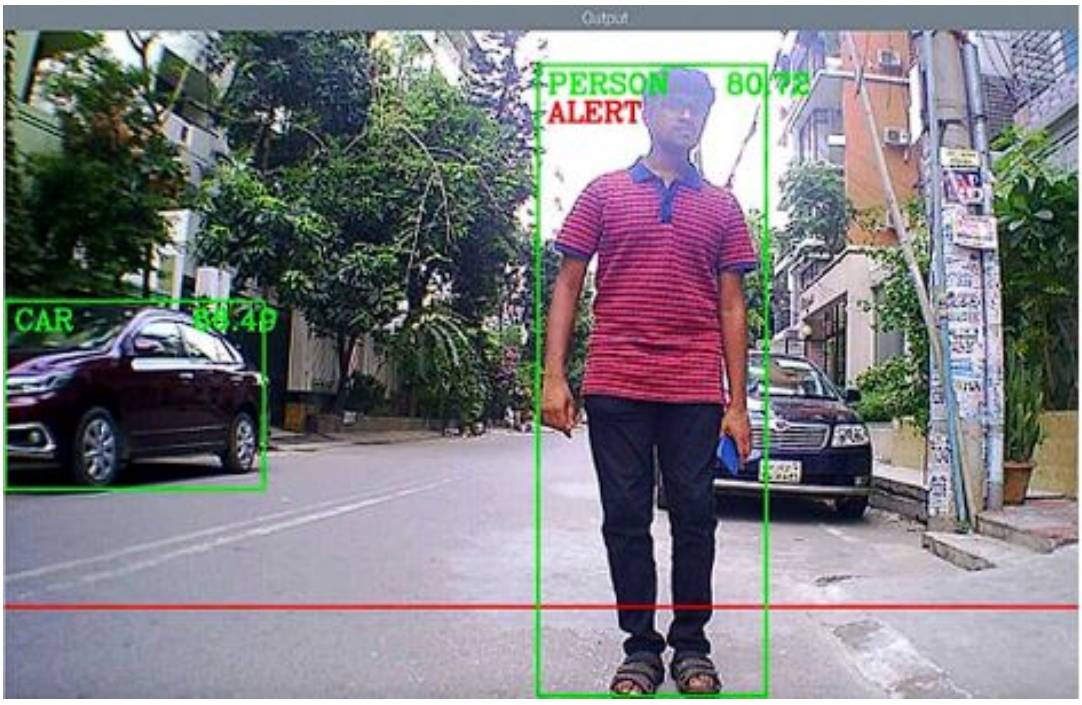



Figure 9.2: Proximity Awareness

## 9.6 Obstacle Detection

Obstacle detection is another critical subsystem that enables the bot to comprehend depth and, therefore, distance from obstacles. The delivery bot we designed has a sonar sensor that enables obstacle detection. By leveraging data from the object detection and obstacle detection modules, the bot can accurately detect, identify, and navigate its way around obstacles. The sonar sensor provides distance information by emitting ultrasonic waves and measuring the time it takes for the waves to return. Paired with the object detection component, the information is enough to create a detailed map of its surroundings and identify potential obstacles in real-time. The incorporation of this subsystem into the design allows the bot to maneuver through crowded spaces, narrow corridors, and other dynamic surroundings. The addition of this sensor is advantageous because it is low-power, relatively inexpensive, and has a wide detection range and field of view that provides robust situational awareness in a variety of environmental conditions.

# Chapter 10

# Results and Analysis

## 10.1 Testing the trained programs

**Obstacle Detection:** To test the obstacle detection system, a person was allowed to stand in front of the vehicle. The rover was manually moved forward and if the system worked properly the vehicle was stopped. Initially, all of the trial runs were a failure where the rover would stop for a moment but would continue after a short pause. Since the image processing power of the raspberry pi was limited, it was noticed that the vehicle was detecting the person after it had crossed its initial stoppage point. This was solved by reducing the speed of the vehicle. On every other trial after adjusting the speed, the program successfully detected the person and stopped, maintaining a proper distance from the obstacle.

Figure 10.1: Starting Position

Figure 10.2: Vehicle Stops after detecting a person over a distance

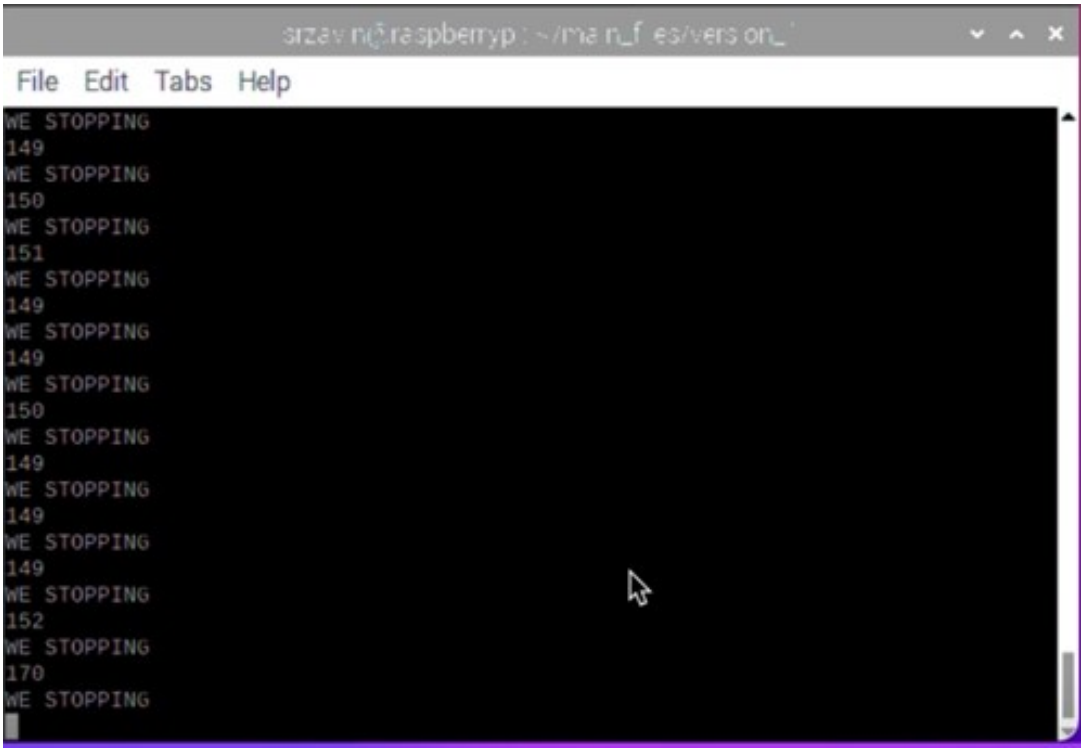



Figure 10.3: Terminal displays the distance to the obstacle and displays the stoppage message

**Autonomous Driving:** The main autonomous driving program uses the tensorflow model library. This library would take in the image from the camera and give the steering value as output. When training the data the image had to be converted by changing the color space, induce blur and crop out the ineffective parts of the image. As the model was trained using the augmented image we make the image similar to what it was trained with. Using the model function from the tensor flow library, the image from the camera is used to predict the steering value. The previous program of manual control was utilized here as well, instead of steering value from the controller the predicted value was given. The test was done on the same street the model was trained with. The throttle control was left to be manually controlled to avoid any accidents. The vehicle was aligned manually with the street and allowed to go forward. The predicted values were close to what it actually has to be. Detecting cars and pedestrians, the vehicle was seen to correct its course accordingly and stay on the road at all times.

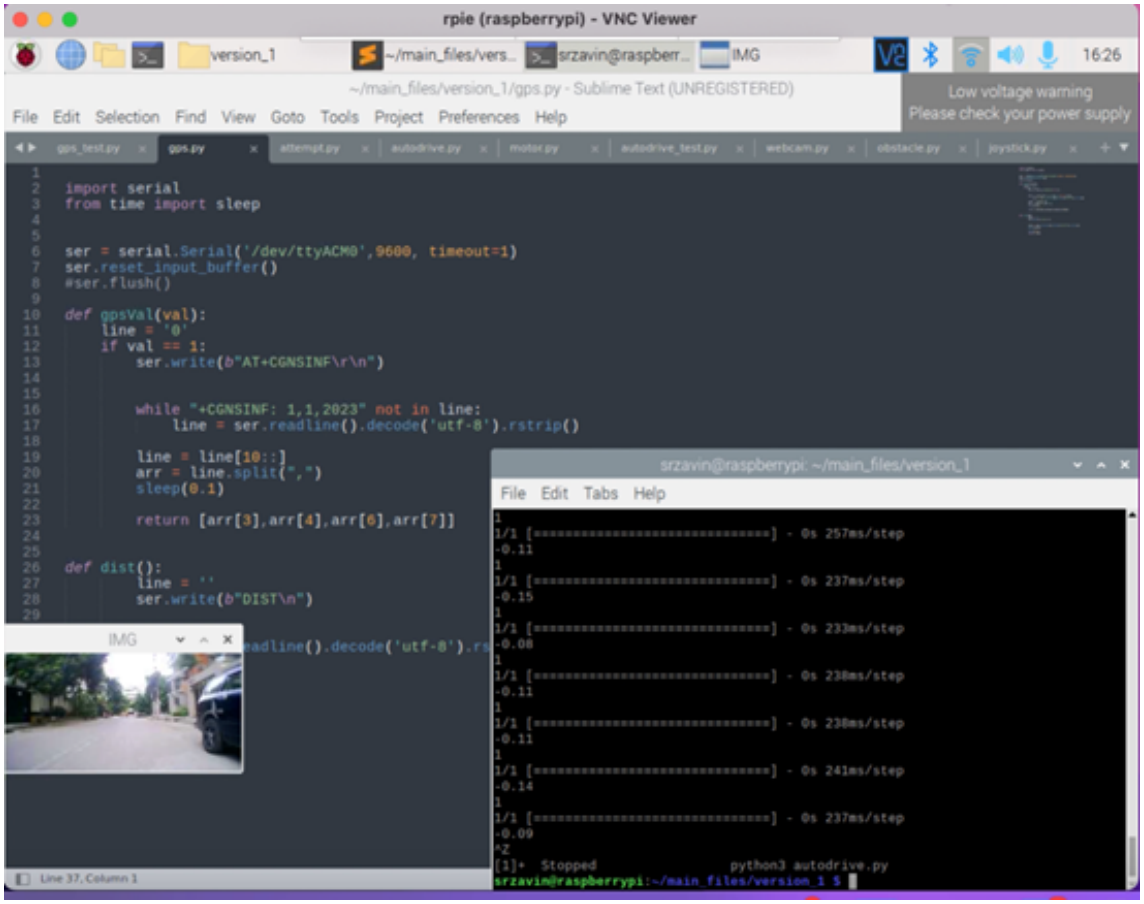



Figure 10.4: Real-time image processing and steering prediction

**GPS Navigation:** The GPS program extracts the longitude and latitude, heading and speed of the vehicle. Using these and data from graphhopper the vehicle would know where to make a turn and where to stop. The turns and the destination coordinates are stored on the program, once the vehicle is close to a turning point the program would slowly move the vehicle towards the turning direction. Later adjustments would be done by the machine learning model that was discussed earlier. Similarly, as the vehicle approaches the destination, the vehicle would stop.

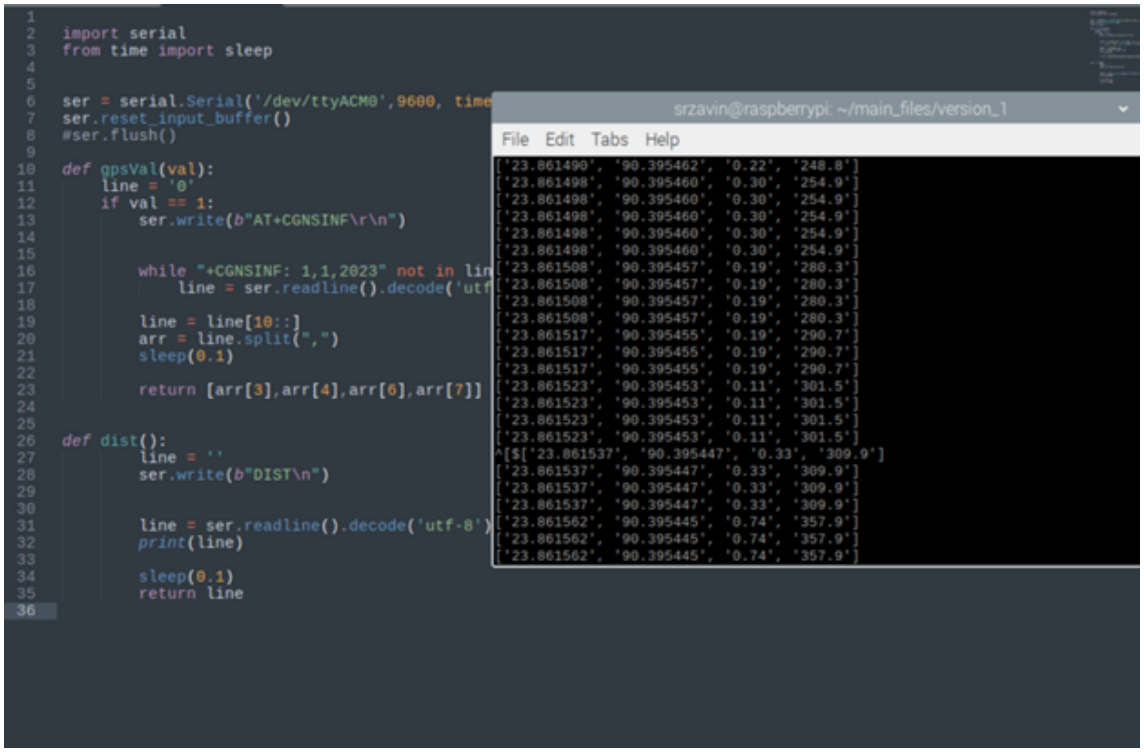



Figure 10.5: Terminal displaying the coordinates, heading and speed

# Chapter 11

# Conclusion

## 11.1 Current Limitations

1. The model needs better and more powerful hardware components to train given a massive dataset. However, it is quite efficient with a moderate amount of data for successful training. The amount of data needed to define the environment effectively is within the model's capability to process and train.

2. The batch generation process makes the training more accurate but takes a little more amount of time. However, the response time of the chassis is not affected by that.

3. Being a cost-efficient chassis, the input hardwares implemented upon the chassis are not very powerful. So the gathered data have noises in them. However, the system excludes these noises while performing data processing as explained above.

4. The model runs fine in a normal environment but might find problems processing data under exceptional environments and circumstances.

## 11.2 Future Development

1. By installing better input hardware, the stored data will be more noise-free and of better quality.

2. Running the model on better processing hardware will make the training faster.

3. More diverse dataset from the environment is needed for better training and understanding for the model. For which, many test runs will be required.

# References:

[1] Wang, J., Huang, H., Li, K., Li, J. (2021). Towards the Unified Principles for Level 5 Autonomous Vehicles. Engineering. https://doi.org/10.1016/j.eng.2020.10.018

[2] Janebäck, E., Kristiansson, M. (2019). Friendly robot delivery: Designing an autonomous delivery droid for collaborative consumption. Odr.chalmers.se. https://hdl.handle.net/20.500.12380/257174

[3] H.T.L. Chiang, B. HomChaudhuri, L. Smith, L. Tapia Safety, challenges, and performance of motion planners in dynamic environments N.M. Amato, G. Hager, S. Thomas, M. Torres-Torriti (Eds.), Robotics research, Springer, Cham (2020), pp. 793-808

[4] Badue C, Guidolini R, Carneiro RV, Azevedo P, Cardoso VB, Forechi A, et al. Self-driving cars: a survey. 2019. arXiv:1901.04407.

[5] Ö.Ş. Taş, F. Kuhnt, J.M. Zöllner, C. Stiller Functional system architectures towards fully automated driving Proceedings of 2016 IEEE Intelligent Vehicles Symposium (IV); 2016 Jun 19–22; Gotenburg, Sweden (2016), pp. 304-309

[6] G. Chen, H. Cao, J. Conradt, H. Tang, F. Rohrbein, A. Knoll Event-based neuromorphic vision for autonomous driving: a paradigm shift for bio-inspired visual sensing and perception IEEE Signal Process Mag, 37 (4) (2020), pp. 34-49

[7] Last Mile Delivery Logistics Explained: Problems Solutions. (2022, April 15). Insider Intelligence. Retrieved September 18, 2022, from https://www.insiderintelligence.com/insights/last-mile-delivery-shipping-explained/

[8] Patel, R. (2022, April 7). Last-mile Delivery Challenges and How to Overcome Them. Upper Route Planner. Retrieved September 18, 2022, from https://www.upperinc.com/blog/last-mile-delivery-challenges/

[9] Ross, S. (2021, March 26). Seven Last-Mile Delivery Challenges, and How to Solve Them. Supply Chain Brain. Retrieved September 18, 2022, from https://www.supplychainbrain.com/blogs/1-think-tank/post/32800-last-mile-delivery-challenges-and-how-to-solve-them

[10] GraphHopper. (2018, February 15). Technical Overview of GraphHopper. GitHub. Retrieved January 14, 2023, from https://github.com/graphhopper/graphhopper/blob/master/docs/core/technical.md

[11] Bojarski, M., Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L., Monfort, M., Muller, U., Zhang, J., Zhang, X., Zhao, J., Zieba, K. (2016, April 25). End to End Learning for Self-Driving Cars. NVIDIA. https://images.nvidia.com/content/tegra/automotive/images/2016/solutions/pdf/end-to-end-dl-using-px.pdf

[12] Microsoft Corporation. (2021, January 7). About YUV Video. Microsoft Learn. https://learn.microsoft.com/en-us/windows/win32/medfound/about-yuv-video

[13] TensorFlow. (2022, January 10). The Sequential model. TensorFlow. Retrieved May 20, 2023, from https://www.tensorflow.org/guide/keras/sequential_model