A Project Report on

Machine Learning based Autonomous Riding System (M.A.R.S)

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

Bachelor of Engineering

in

Information Technology

by

Fenil Bhimani(19204012)
Om Bheda(19204011)
Amey Mohite(18104067)

Under the Guidance of

Prof. Apeksha Mohite

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

Academic Year 2021-2022

Approval Sheet

This Project Report entitled ***"Machine Learning based Autonomous Riding System"*** Submitted by ***"Fenil Bhimani"(19204012),"Om Bheda"(19204011),"Amey Mohite"(18104067),*** 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***.

Prof. Apeksha Mohite
Guide

Prof. Kiran Deshpande
Head Department of Information Technology

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

CERTIFICATE

This is to certify that the project entitled ***"Machine Learning based Autonomous Riding System"*** submitted by ***"Fenil Bhimani"(19204012),"Om Bheda"(19204011), "Amey Mohite"(18104067),*** 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 2021-2022.

Prof. Apeksha Mohite
Guide

Prof. 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:

Acknowledgement

We have great pleasure in presenting the report on **Machine Learning based Autonomous Riding System.** We take this opportunity to express our sincere thanks towards our guide **Prof. Apeksha Mohite** Department of IT, APSIT, Thane for providing the technical guidelines and suggestions regarding line of work. We would like to express our gratitude towards her constant encouragement, support and guidance through the development of project.

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

We thank **Prof. Vishal S. Badgujar** 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.

Student Name1: Fenil Bhimani
Student ID1: 19204012

Student Name2: Om Bheda
Student ID2: 19204011

Student Name3: Amey Mohite
Student ID3: 18104067

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.

(Fenil Bhimani - 19204012)

(Om Bheda - 19204011)

(Amey Mohite - 18104067)

Date:

Abstract

Vision begins in eyes, but truly takes place in the brain. So, in today's world with the best high-definition cameras, high-speed computers, and artificial intelligence, computer vision was introduced. Computer vision is one of the latest advancements in technology that helps on giving the abilities of vision and understanding of the environment to computers so that they can extract high-level understanding from digital images and videos.
In several industries machines are being trained using artificial intelligence to reduce human errors and perform high-precision tasks with accuracy. Autonomous vehicles are the application of computer vision with main motive of eliminating human errors and drive the vehicles with high precision and remove the efforts taken by a person to drive and to take advantage of the technology for travelling around safely and securely. It will be implemented over a small scale on a RC-sized car. The car would traverse autonomously in a demo map where we can simulate dynamic conditions and assess the model's performance accordingly.
This is the main motivation behind the project. To create a completely autonomous system capable of navigating around on its own. Detect, identify and follow traffic signs, signals and rules. Take appropriate actions for dynamic conditions and avoid any sort of loss. Provide user with an application interface to experience a safe journey to selected destination. The proposed system will be using LiDAR for mapping the telemetry and detecting surroundings of the vehicle, a camera module to identify the objects detected, a development board with the deployed autonomous driving model, being the brains of all the operations, and actuators for moving the car around. The autonomous driving model to be deployed, will be trained using Nvidia's DiscoBox for semantic segmentation of the surroundings and classifying each pixel in the input along with the dynamic decision-making subsystem to take some decisions based on the perception to reduce the severity of certain accidents or even avoid them.

Contents

1 Introduction **1**
- 1.1 Problems Identified . . . 1
- 1.2 Solutions Proposed . . . 2
- 1.3 Objectives . . . 3

2 Literature Review **4**

3 Project Design **8**
- 3.1 Hardware . . . 8
- 3.2 Mechanical Design . . . 9
  - 3.2.1 Steering . . . 10
  - 3.2.2 Base Plate . . . 10
- 3.3 Autonomous Riding . . . 11
  - 3.3.1 Semantic Segmentation . . . 12
  - 3.3.2 Control System . . . 12
- 3.4 Mobile Application . . . 13
  - 3.4.1 Front-End . . . 13
  - 3.4.2 Back-End . . . 14

4 Project Implementation **16**
- 4.1 Hardware . . . 16
- 4.2 Autonomous Riding . . . 17
  - 4.2.1 Semantic Segmentation . . . 17
- 4.3 Mobile Application . . . 22

5 Testing **25**
- 5.1 Unit Testing . . . 25
- 5.2 Integration Testing . . . 25
- 5.3 Load Testing . . . 25
- 5.4 Model Testing . . . 26

6 Result **28**

7 Conclusions and Future Scope **33**
- 7.1 Conclusion . . . 33
- 7.2 Future Scope . . . 33

Bibliography **35**

Appendices **36**
Appendix-A 36

Publication **38**

List of Figures

3.1 Hardware Interfacing with Jetson Nano 9
3.2 Steering Geometry 10
3.3 Steering System Assembly 10
3.4 3D printer in action 11
3.5 3D printed Base Plate and mounts 11
3.6 Segmentation by DiscoBox 12
3.7 Autonomous Riding System 13
3.8 Application Flow 15

4.1 Complete Vehicle Assembly - Back View 16
4.2 Complete Vehicle Assembly - Isometric View 17
4.3 DeepLabV3-R50-D8 Segmentation Model 18
4.4 Improvement in output of models trained on IDD dataset 19
4.5 Customized Dataset Training 19
4.6 Script to convert custom dataset to cityscape supported format 20
4.7 Train.py 21
4.8 Main 22
4.9 Home 22
4.10 Home 23
4.11 Sign-in 23
4.12 Sign-up 24
4.13 Splash-screen 24

5.1 Input Image 26
5.2 Custom Trained Model output 27
5.3 Pre-Trained Model output 27

6.1 Sign-in Page 28
6.2 If login credentials entered are wrong 29
6.3 Sign-up page 29
6.4 Home Screen after successful login 30
6.5 Pre-Trained Model results in college basement 31
6.6 Custom Trained Model results in college basement 32
6.7 Custom Trained Model output 32

7.1 Side-View R742 34
7.2 Isometric View R742 34

List of Tables

4.1 Performance comparison of different Methods 17

List of Abbreviations

IDS:	Intrusion Detection System
WSN:	Wireless Sensor Network
MANET:	Mobile Ad-Hoc Network
AODV:	Ad-Hoc On-demand Distance Vector Routing
DSR:	Dynamic Source Routing Protocol
NS2:	Network Simulator 2
ACK:	Acknowledgement
AGT:	Agent
RTR:	Router

Chapter 1

Introduction

The fast-moving progress of applied artificial intelligence (AI) and the predicted importance of autonomous vehicles on the future, from independent mobility for non-drivers and low-income individuals, reduced pollution, traffic and parking congestion, increased safety on the roads, and the ability to create such technologically advanced systems that are self-dependent in decision making, is the main motivation behind the entire project. Autonomous vehicles are also predicted to be relied on in some of the most complex human planned endeavors, such as space exploration. The meteoric rise of AI along with deep learning (DL) methods and frameworks, have made possible the creation of such an autonomous vehicle without expensive laboratories and years of research.

The system that we have proposed is to built a self-driving car that is scaled-down to the size of an RC Car. Along with learning the new technologies and exploring this domain, goals from the project is to build a model capable of navigating through the map autonomously, while demonstrating the capability to perform behaviors such as lane following, dynamic actions with respect to environment changes. The project goes through the entire process of building such a vehicle, starting from the RC car model and the embedded hardware platform, to the orchestration end-to-end machine learning algorithms and a user application.

1.1 Problems Identified

All traditional systems have a scope for improvement and change. Driving a motor-vehicle is one of the basic tasks that takes place at some point in every person's life. Driving a car requires judgement, control over body, and constant observation of the things happening in the surrounding to traverse through the roads safely, not only for the driver but everyone around.

This task seems easy but a lot of human error takes place around us causing accidents. [9] Every year, traffic accidents account for 2.2 percent of global deaths. That stacks up to roughly 1.3 million a year - 3,287 a day. On top of this, some 20–50 million people are seriously injured in auto-related accidents each year. The root of these accidents? Human error. Humans are prone to errors may it be drivers or pedestrians there are many examples that we have faced personally that are caused due to over-confidence, lack of judgement and other distractions. Unintentional or foolish interruptions caused by the pedestrians, domestic animals and other drivers on road results in accidents and loss of both health and wealth.

Lack of road sense, taking the traffic rules lightly, and disregard to the authorities also results in traffic or vehicle environment difficult to travel in. [10] In 2020, 10,010 four-wheelers were towed away for wrong parking and a fine of Rs 85,73,642 was recovered.

Specially abled people who are physically unable to travel alone, require drivers to carry them around or have to modify their cars with attachments which costs them a lot of money.

1.2 Solutions Proposed

We have seen many implementations of self-driving car by major companies like Waymo, Tesla, etc however the scope for improvement still exists and will always continue to exist. The system that we have proposed is a small step for improvement of more efficient and secure commercial self-driving cars. In which we use a scaled down model to the size of an RC-car. This RC-car will be easier in management, easier to test on and extremely cost efficient in prototyping and initial stages. Additionally, the cost and risk of damage to property and life in testing environment is negligible. Our proposed system architecture will be a completely autonomous riding system scaled down to the size of an RC car that will have its own –

- Environment sensing techniques for detection and identification of objects.
- Decision making system driven by traffic rules and selective moral/ethical values.
- Shortest path finding system to find the route with shortest distance to reach the destination
- Hardware system with actuators for smooth movement of car based on commands given by the decision-making system.
- Mobile Application that will have the options for selecting destination, starting and stopping, and video feedback from car's cameras. The rider in the car will also have override access to the cars control for emergency through the mobile application.

We will be using technologies like Embedded Systems for perceiving the environment and actuate the car based on these perceptions, Computer Vision for analysing these inputs and detecting identifying obstacles, road signs and symbols, and take dynamic actions for it [11] [13]. The main control of all these operations will be the Nvidia Jetson Nano development board, with powerful processing and tensor cores making it a perfect use for our application of the RC-sized autonomous vehicle.

LiDAR will be used for setting the telemetries of the surrounding world, and a camera module will be used to label these telemetries identified by the LiDAR into classes.

This entire model will be scaled down to a pre-determined and controlled map with manual exceptions to simulate the working in real life and train the algorithm extensively. The main objective of this is to provide a proof of concept for autonomous driving system for future implementation on an actual sized car that would be ready for deployment in real world.

1.3 Objectives

1. To design, fabricate and develop a completely autonomous scaled down vehicle using latest trends in the field of Computer Vision, AI and Embedded Systems as per the feasibility.

2. To apply an algorithm that can find the shortest path with respect to distance in the scaled down map created.

3. To orchestrate a system that detects, identifies, and follows the traffic signs and rules on the roads.

4. To create a system that morally and ethically takes dynamic action for any implausible condition that occurs.

5. To provide user with an application interface that has an overwrite command with higher priority than the system commands.

6. To understand new technologies, hardware and implement the same in a feasible manner.

7. To manage the entire project in a professional manner by cost management, team management and waste management strategies.

Chapter 2

Literature Review

1. **Title** – Building a Self-Driving RC Car Master's Thesis
   Publication – Master's thesis to University of Zagreb, 2020
   Author Name – Ivan Oršolic´
   Conclusion – In the thesis the author created a self-driving RC car using Embedded Systems, OpenCV, DonkeyCar, Keras methodologies. The paper showed detailed description regarding the hardware and the software used in the project. It included comparisons, setting up of the environment and connections for the components used. It introduced to the DonkeyCar library that is developed for creating self-driving cars similar to that of the scope of our project. The entire project was quite similar to the ideas that we have and implemented some of the objectives that we used. Overall, the thesis paper solved a few doubts and discrepancies regarding the hardware selection and gave a new light on DonkeyCar.

2. **Title** – A Comparative Study on Machine Learning Algorithms for the Control of a Wall Following Robot
   Publication – IEEE International Conference on Robotics and Biomimetics (ROBIO) – 2019
   Author Name – Issam Hammad, Kamal El-Sankary, and Jason Gu
   Conclusion – The methodologies used by in paper were Keras, Scikit-learn, Monte-Carlocross-validation, Machine learning algorithms – Decision tree, Gradient Boost Classifier, Sup-port Vector Machines, KNN, Gausian Naive Bayes. The understanding from the paper included that according to the popular No Free Lunch Theorem, there is no golden machine learning algorithm that can outperform all the other machine learning algorithms in solving all possible problems. Identifying correct algorithm based on application is important. The paper provided comparison between the algorithms with different data-sets and showed the results for the best performing algorithm for the given application i.e. wall following robot.

3. **Title** – Autonomous Vehicles and Embedded Artificial Intelligence: The Challenges of Framing Machine Driving Decisions
   Publication – XIII Conference on Transport Engineering, CIT2018
   Author Name – Martin Cunneen, Martin Mullins, and Finbarr Murphy
   Conclusion – The paper covered the moral decisioning problems faced by algorithms

while driving. How not having an emotional quotient may affect the weighted inputs and their corresponding outputs. This paper focused more on ethical challenges as well as the social dilemma that can revolve around autonomous cars and how over-fitting of data while training can make the AI model biased to certain aspects and how this problem is solved by having appropriate diversified datasets and sufficient training of the model.

4. **Title** – Comparative Analysis of Machine Learning Algorithms on Different Datasets
   Publication – International Conference on Innovations in Computing (ICIC), 2017
   Author Name - Kapil Sethi, Ankit Gupta, Gaurav Gupta, Varun Jaiswal
   Conclusion – In the paper the authors used MATLAB to simulate the working of different algorithms for varying datasets. The sensitivity and accuracy of NN, SVM and KNN are determined from the simulations. The schema used by the authors for the experiment was identified and as per the examination, SVM classifier contrasted better than KNN and NN with accuracy of close to 99.38

5. **Title** – Waymo Driver
   Conclusion – Waymo is an independent autonomous driving technology company established under Alphabet. Waymo Driver is their autonomous driving technology designed with safety and a lot of experience. The Waymo Driver is the world's most experienced driver which has travelled millions of miles on public road and billions of miles in simulation since 2009. The Waymo Driver drives 2 different vehicles i.e. Waymo One ride-hailing service for moving people and Waymo Via focuses on transporting commercial goods.
   The designed system first requires the map of a new territory with high granularity, this not only assists in the autonomous working of the driver but also removes the reliability on external GPS data. The Waymo Driver has detailed custom maps to determine its exact location in real-time. It also uses its own perception system to collect and understand its surrounding using high advanced sensors and technologies like Machine Learning. This real-time data and the previous driving experience that the system is trained on, is used to anticipate the behaviour of other traffic scenarios and possible path that other objects can take. It understands how a car moves different than a pedestrian. Using all of this information the best plan of action is decided for the vehicle.
   To execute the mentioned tasks the Waymo vehicle requires LiDAR sensors for giving a 3D picture of the vehicle's surroundings, RADAR sensors for providing details like an object's distance and speed, Cameras providing 360° view around the vehicle, and onboard computer with latest server grade CPUs and GPUs. It takes information provided by n-number of sensors on the car, identifies the different objects and plans a safe route towards your destination in real time. The carefully designed system also comes with back-up systems namely the secondary compute, Backup collision detection and avoidance system, Redundant steering, Cybersecurity, Backup power systems, and Redundant braking.

6. **Title** – Tesla Autopilot
   Conclusion – Tesla cars come advanced hardware capable of providing Autopilot features, and full self-driving capabilities. Advanced safety and convenience features are designed to assist the drivers with the most burdensome parts of driving, it enables the car to steer, accelerate and brake automatically. Due to legal constraints the current autopilot features require active driver supervision and hence, the cars are not completely autonomous. The tesla autopilot has many different features namely –

   - **Navigate on Autopilot** – Navigate on Autopilot suggests lane changes to optimize your route, adjusts so the vehicle does not get stuck behind slower traffic. It will also automatically steer your vehicle toward highway interchanges and exits based on your destination.
   - **Autosteer+** – Using advanced sensors and computing power, the Tesla will navigate through tighter and complex roads effortlessly.
   - **Smart Summon** – The car will navigate more complex environments and parking spaces, maneuvering around objects to come to the driver in a parking lot/garage.
   - **Full Self-Driving Capability** – Tesla cars have the hardware needed in the future for full self-driving in almost all circumstances. From home to your destination the entire commute would be possible autonomously. The person travelling would just require to put a destination, the car would come to the pick-up and travel to destination choosing the optimal route, traffic and other parameters. On reaching destination the car will also automatically identify and park the car in the accurate spot.

   Tesla autopilot is equipped with safety features that include Automatic Emergency Braking for detected objects that may collide with the car and automatically apply brakes accordingly, Auto High Beams automatically adjusts high/low beams as required, and Front Side Collision Warning that helps warn the driver for collisions that might occur with stationary objects/obstacles or slower moving traffic around the vehicle.
   The cars are covered advanced sensors and Tesla Vision. The car has 8 surround cameras provide 360 degrees of visibility around the car at up to 250 meters of range. They include –

   - 3 narrow forward cameras
     - A Wide 120-degree fisheye lens - captures traffic lights, obstacles cutting into the path of travel and the objects that are at close range.
     - A Main camera covering a broad spectrum of different use cases.
     - A narrow camera providing a focused, long-range view of the objects present at a distant.
   - Forward looking side cameras, these are 90-degree side cameras for alerts against real-time exceptions that might occur.
   - Rearward Looking Side Cameras monitoring the rear blind spots on both sides of the car,
   - for safely changing lanes and merging into traffic. Rear View Camera is useful when performing complex parking maneuvers and also assists the autopilot for enhancement of optics.

Along with the camera modules the car is covered with 12 ultrasonic sensors that allow detecting objects that are both hard and soft to be able detect nearby cars and provide guidance when parking. The 12 sensors are strategically placed to cover all possible areas.
To understand and take actions for the data collected from all these different sensors and camera modules 3 onboard computers are present. These computer runs the Tesla-developed neural net which used to train and develop Autopilot. The tesla developed FSD and Dojo chips are the AI inference and AI training chips powering Dojo systems. The FSD Chip is designed to run the full Self-Driving software. These FSD chips by tesla are implementing small architectural improvements to maximize Performance-to-Power ratio, Robust testing of the trained model along with evaluation for the functionality and performance. It has drivers and programs that focuses on performance optimization and redundancy by communicating with the chip.

Chapter 3

Project Design

3.1 Hardware

The hardware platform for completely autonomous RC-sized car consists of the following components –

- Jetson Nano 2GB Developer Kit
- Battery Operated (BO) DC motors
- Servo motor
- L298D - motor driver
- LiDAR sensor
- Ultra-sonic sensors
- Camera
- GSM Module

These components are interfaced with the Jetson Nano developer kit using the GPIO. It is the brains of the entire car and will have all the control commands and autonomous systems deployed on it. Jetson Nano is a very advanced high-performance controller. It has dedicated Maxwell GPU, Quad-core ARM CPU, high speed RAM, and many other latest features in an affordable price range, providing better cost-to-performance ratio. The board is designed with the goal to deliver incredible AI performance at low price ranges and make the fields of AI and robotics more accessible. The board uses python for communication with the GPIO and programming use-cases. [Ivan]Hence, making it the perfect choice for the application and scope of our project.

The LiDAR and the camera are the sources for all the observations being made in the surroundings of the car. The USB I/O ports will be used to interface these sensors respectively. Python being open-source has easy to use and configurable libraries for interfacing these sensors to the board and start the perception of data instantaneously. The ultra-sonic sensors will be used to monitor objects in near range and also assists during parking the car. The sensors are digital sensors mounted on the GPIO pins of the Jetson Nano, and alert

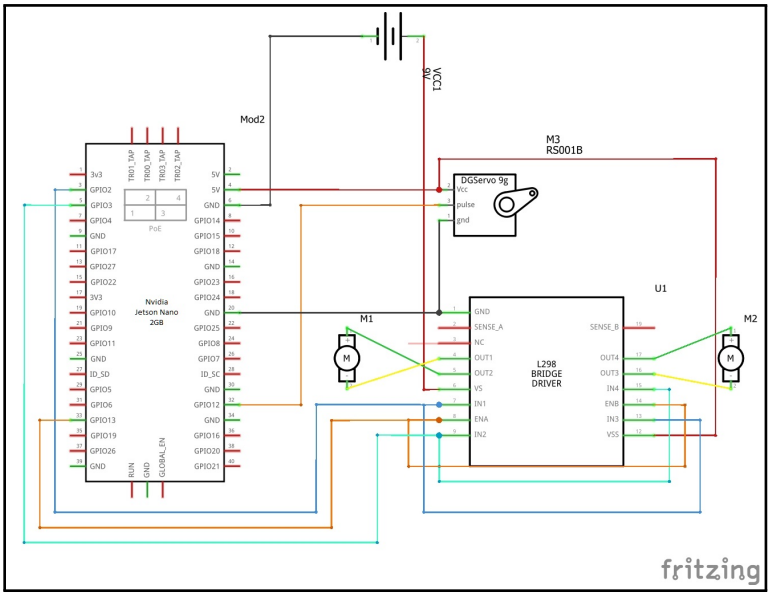

Figure 3.1: Hardware Interfacing with Jetson Nano

the decision-making system when they are in a particular range, for detecting and avoiding collisions.

After providing the eyes for our computer vision system, we require to provide actuators to move the car around. This is taken care by 2, BO DC motors with 100 rpm and 2kg/cm of torque to carry the weight of our entire car around, and a servo motor in the front to steer and manoeuvre the car. Thus, making the car a rear-wheel drive system. To control these parts, we will be using L298D - motor driver.

L298D is a powerful and simple to use motor driver. This driver communicates with the Jetson Nano using PWM and Digital pins the on-board PWM controller drives the motor speed by changing the duty cycle There are 2 output pins OUT1, OUT2 , and 3 input pins IN1, IN2, ENA for each motor respectively. We can drive 2 DC motors using L298D. Based on the precepted information the commands to move as per the respective situation will be given by Jetson Nano, this will then be converted into actuating signals for controlling longitudinal movements, the servo to control the cars steering is directly connected to the on-board PWM output pin.

To connect the car with the rider's app we will be using the GSM-Module for providing peer-to-peer connectivity to the Jetson Nano. The car would take commands and act upon them and it will continuously feedback live data from the camera to the mobile app.

3.2 Mechanical Design

The design of the mechanical sub-system plays a very crucial role in the development of the autonomous car. Irrespective of the size of the car the dynamic behaviour has to be considered during the designing of the car. We have applied very basic Vehicle Dynamics concepts for making the car mechanically capable to move. The design is section 2 parts which are as follows -

3.2.1 Steering

The steering system is based on the Bell-Crank Steering. Bell Crank systems have 2 moving pivots and 1 fixed pivot. The fixed pivot will stay at the center of the servo motor, while the other two moving pivots are connected to the either two of the front wheels respectively.

For proper turning it is required to apply force on the tires with a certain offset so that effort to turn the wheel is reduced. This is known as the kingpin offset. The steering system achieves around 81 percent of ackerman, meaning the slippage of the tires around sharp turns is very less. The components to create the steering system include wheels, servo motor, tie rods, and a central hub to assemble all the components.

The central hub and the bell crank lever were designed in SolidWorks and 3D-printed in the in-house 3D printer at the college with PLA material. The .stl part file was used in the Ultimaker Cura sotware to create .gcode files for the 3d printer. The two hubs and the lever weighed a total of around 100gms and took 2 hours and 13 minutes to print.

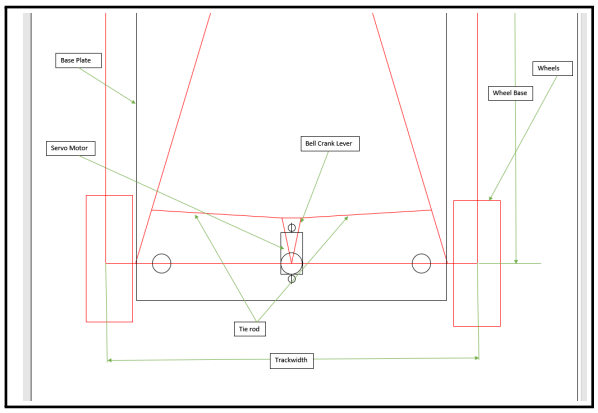

Figure 3.2: Steering Geometry

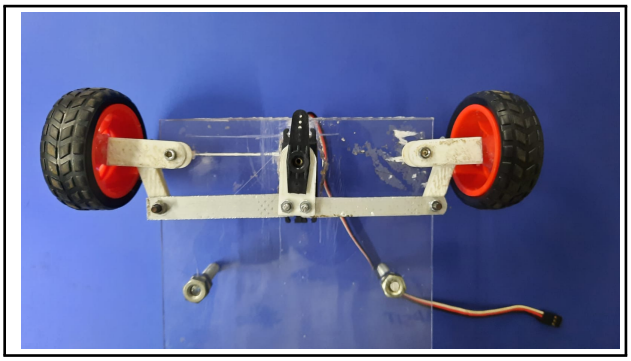

Figure 3.3: Steering System Assembly

3.2.2 Base Plate

The base plate is custom designed with proper measurements of the components and mounting points. The base plate was 3D printed with PLA material. The bed size of the available

printer is smaller than the entire base plate size, so the plate was divided into 6 sections. Each section weighing around 45-60 grams and taking print time of around 6-7 hours each.

The base plate consists of mounting points with extrusions that align with the holes on the boards and hardware. The entire structure of the chassis has 3 layers from top to bottom. The topmost being the mounting for LiDAR sensor, the middle layer being the mounting for all the electronics, and the last bottom layer being a plastic sheet for all the actuators. All of them clamped together with nuts and bolts.

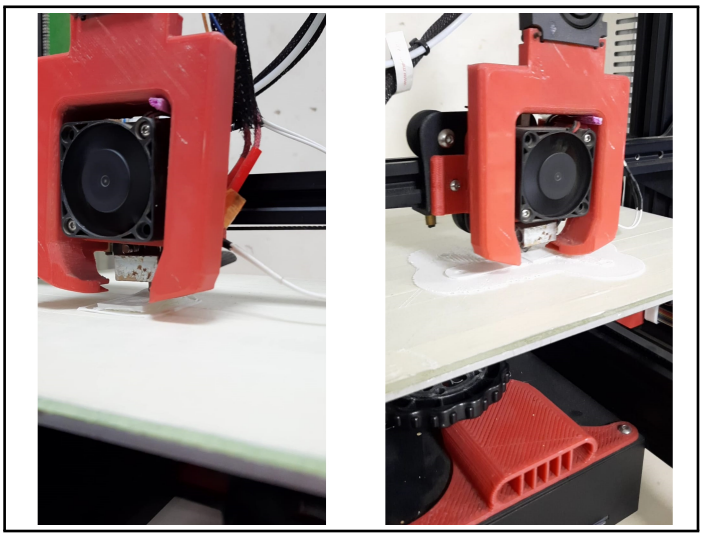

Figure 3.4: 3D printer in action

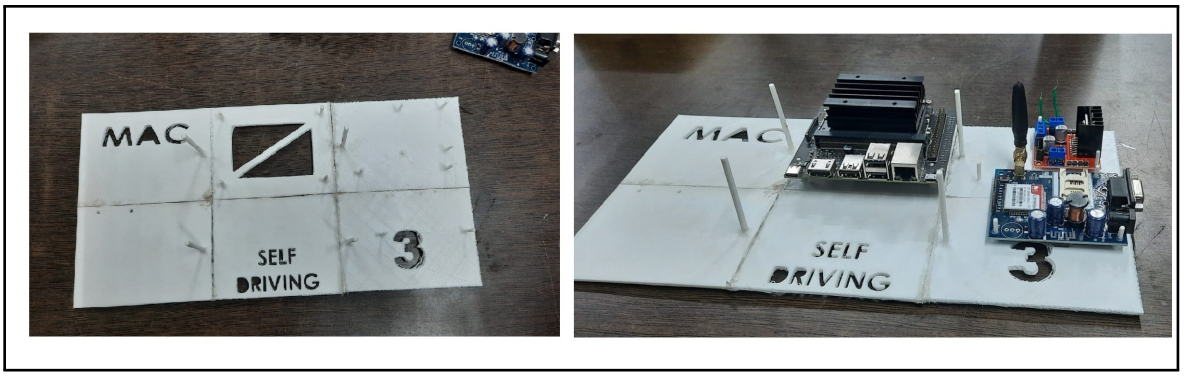

Figure 3.5: 3D printed Base Plate and mounts

3.3 Autonomous Riding

The autonomous riding system gives the car the capability of driving autonomously. It takes inputs from sensors, detects and segments objects, obstacles, road signs, traffic rules, and based on these received inputs it predicts the behaviour of the traffic around the car and chooses the most efficient and safe path to travel to the destination. After choosing the path

it commands the connected actuators that are the motors, to move according to the selected travel path.

3.3.1 Semantic Segmentation

For the first task of making the vehicle see and understand objects in its surrounding we use Semantic Segmentation. Images are nothing but a collection of pixels. Image segmentation is the process of classifying each pixel in an image belonging to a certain class and hence can be thought of as a classification problem per pixel. One of the type of segmentation is Semantic segmentation,it is the process of classifying each pixel belonging to a particular label.It treats multiple objects of the same class as a single entity.

There are two basic conventions followed in an image segmentation task which are as follows:

1. Any countable entity such as a person, bird, flower, car etc. is termed as a **thing**.
2. An uncountable amorphous region of identical texture such as the sky is termed as **stuff**

We will be using [nvidia] Nvidia's DiscoBox framework released in GTC 2022 for the task of semantic segmentation. The model uses Open MMLabs Semantic Segmentation as the base. The main focus of the framework is to reduce the effort and cost taken in creating masks and annotations for training. They do this by using a weakly supervised learning algorithm that can output high-quality instance segmentation without mask annotations during training. The framework generates instance segmentation directly from bounding box supervisions, rather than using mask annotations to directly supervise the task. Each rectangle encodes the localization, size, and category information of an object.

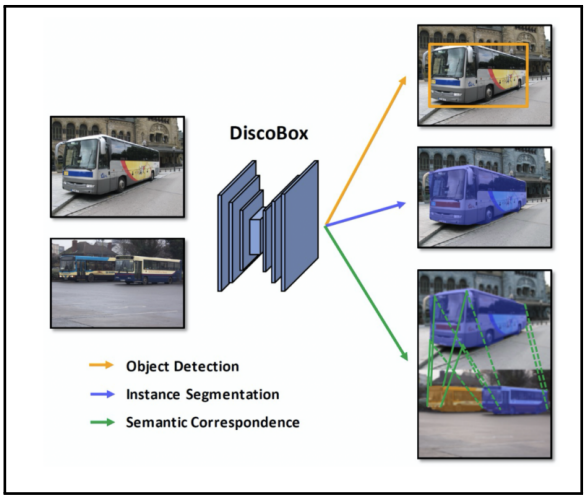

Figure 3.6: Segmentation by DiscoBox

3.3.2 Control System

The control system of the vehicle is designed to be modular and easily scalable. It uses the principles of PID controller to identify and minimize errors in the predicted and actual

system. First, the control system takes the input from the segmented video of camera and using the classes it identifies drivable area and overlays a computed path on top of the segmented video. The control system is aware about the vehicle and environment telemetry. This both input are used to send actuating signals to the actuators. Based on the movements of the vehicle and the feedback from sensors the actual path line of the vehicle is also overlayed on the output of segmentation. The computed path line and the actual path line must always have an offset of close to 0. The PID controller with proper tuning is used to achieve the error rate close to 0 and always follow the computed/predicted path.

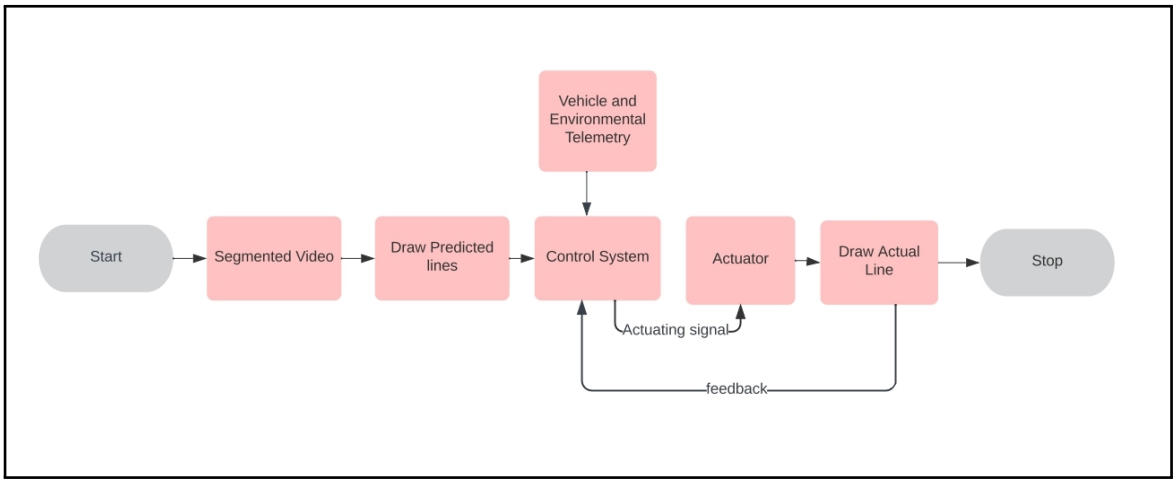

Figure 3.7: Autonomous Riding System

3.4 Mobile Application

To provide connectivity between the rider and the car, with the features of setting up source and destination for the journey, controlling the car movements in case of emergency and provide the required security from external threats particularly the cyberspace. The Mobile application is developed using flutter and Firebase. The flutter app will provide the required GUI for controls, camera feedback, map layout, and flutter will help implement the cross-platform app compatibility. Firebase will be used in the back-end to authenticate rider credentials, before starting the journey.

3.4.1 Front-End

UI

For our UI development we have used Flutter framework. It is an open source UI software development kit. It also helps in cross platform development that means it can run on both Android and IOS. It has multiple packages that helps to design our app like buttons, check-boxes, text, radio buttons, list-view, grid view, etc. We are using youtube live channel stream to show our live camera feed display directly from the car.

Camera Feed

The camera feedback from the vehicle is transmitted using Remote Transport Protocol. The video-viewer tool provided by the jetpack SDK directly transmits the video feed from camera over to the host at port number 1234. The user app then accesses this live video stream and displays it on to the user's screen. The user end can either have gstreamer application or an .sdp file for viewing content in the VLC media player software.
The video-viewer provides different bitrates and encoding types for the video stream. The user can select latency based on the application of the system and also in a file format desired by the user. The gstreamer has the least latency and jitter and stream has the least delay.

3.4.2 Back-End

We have used Google Firebase as our backend for our application. Google Firebase is a Google-backed application development software that enables developers to develop iOS, Android and Web apps. Firebase provides tools for tracking analytics, reporting and fixing app crashes, creating marketing and product experiment. It offers number of services such as

- Analytics
- Authenctication
- Cloud messaging
- Real Time database
- Crashlytics
- Performance

We haved used Authenctication and Firestore database. In authenctication we have fetched the details of user to give access to our application through the signup page provided by it. The details that are fetched are email id and password. We also have added an snippet which checks that whether the email provided is valid or not.
In Firestore Database, when the user registers via the application, document is been created in database with reference to his credentials the user gave. The attributes of the database are the id and users. Use of id is that it has the host URL of the Live Feed camera display and the users contains the email id of the user and other details.

Working of Application: The system proposed by our team will have an application which can control the car. The application will have following features –

1. It will have a GUI interface through which the user can select its destination.
2. As soon as user selects the destination, user needs to be validated. Once user is authenticated, then only user can start its journey. Validation will be done through the database.
3. User can stop the car or can overwrite the system commands; this feature is given by our team so that if any implausible condition occurs and if the machine gets confused on which thing to act-on then the user can take the decision.

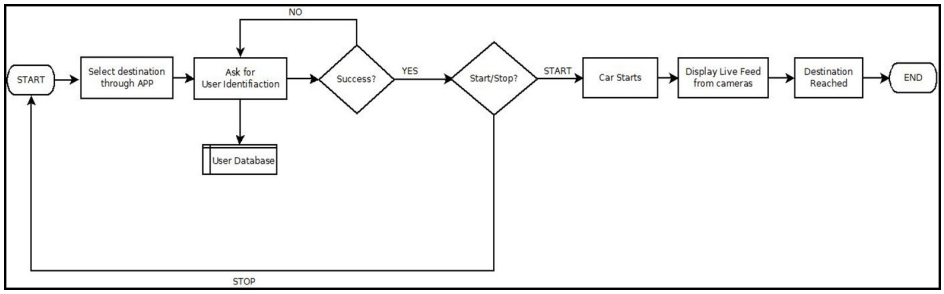

Figure 3.8: Application Flow

Chapter 4

Project Implementation

4.1 Hardware

The hardware implementation was done in parts. First, the components were decided on the basis of cost, availability, compatibility with other components. Accordingly a budget was formed and the components were procured by online and offline means.
After procurement the dimensions were taken and the placement was decided to design the base plates and the steering parts were 3D printed. After all the parts were ready the assembly was done and all the parts were integrated.
The 3D print process was very crucial as the mounts extruded were quite delicate and had broken several times. The previously acquired motor were not able to output the required torque so new motors with higher torque were procured. The weight of the entire system weighs approx 2kgs.

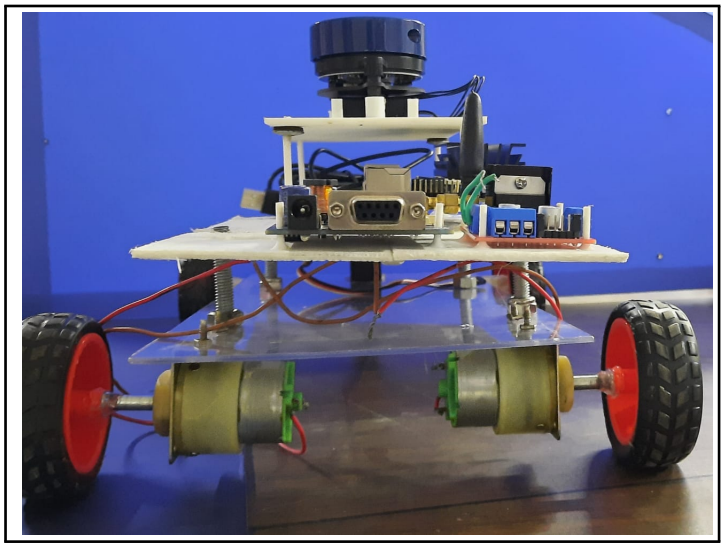

Figure 4.1: Complete Vehicle Assembly - Back View

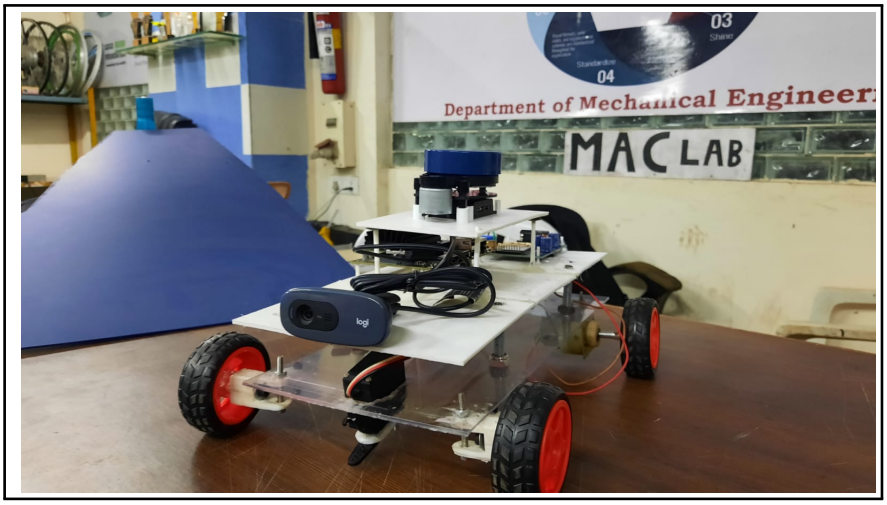

Figure 4.2: Complete Vehicle Assembly - Isometric View

4.2 Autonomous Riding

The project first started with implementation with YOLO object detection. However, in compliance with the first objective, when we became aware of the potential and scope of Semantic Segmentation and Nvidia's DiscoBox Framework we shifted to Semantic Segmentation as the method for giving the vehicle visual understanding of the surrounding. This change was feasible because the implementation had not started on the Jetson Nano board and hence we could adapt the change.

4.2.1 Semantic Segmentation

The Model

Model	Backbone	Crop Size	LR Schedule	Memory	Inf time(fps)	mIoU
DeepLabV3	R-50-D8	769x769	40000	6.9	1.11	78.58
FastFCN + DeepLabV3	R-50-D32	512x1024	80000	5.67	2.64	79.12
DeepLabV3+	R-50b-D8	512x1024	80000	7.4	3.94	80.28
DeepLabV3(FP16)	R-101-D8	512x1024	80000	5.75	3.86	80.48

Table 4.1: Performance comparison of different Methods

The comparison table shown above compares the different models with their required GPU memory in training, their inference time in fps, and the metric of measurement i.e. mIoU. These methods are based on backbone Fully Convolution Neural Network. The backbones in the compared methods are different versions of ResNet. The first difference in the backbone network is the dilation rate or down-sampling rate. and the other difference is the depth of the layers of the neural network. The Learning rate scheduler defines the iteration based runner with maximum number of iterations the model went through while training.

Based on our application and scope we require less inference time as the system has to be very responsive and capture details from environment faster. We used the DeepLabV3

model with backbone R-50-D8 and Inference time of 1.11fps. Along with that we are limited by the GPU memory. The creators and researchers of the backbone networks use workstations and multiple GPUs for training which was not feasible for us hence we went with a model that could meet our hardware requirements.

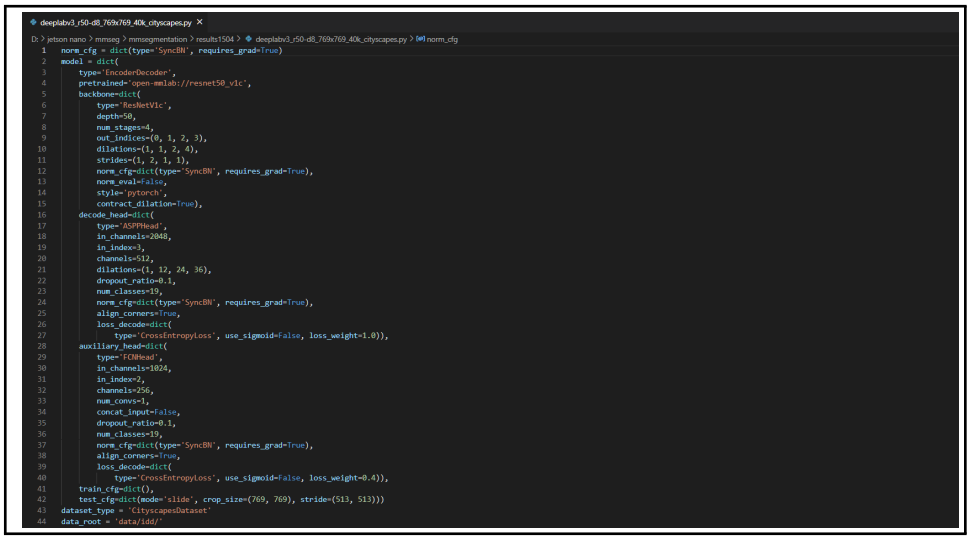

Figure 4.3: DeepLabV3-R50-D8 Segmentation Model

Dataset

The dataset that the selected model was trained on is the cityscapes dataset. Cityscapes is a large-scale dataset that contains a diverse set of stereo video sequences recorded in street scenes from 50 different cities, with high quality pixel-level annotations of 5000 frames in addition to a larger set of 20000 weakly annotated frames. The dataset is largely based on structured/disciplined environment and when tested on the Indian road conditions which are largely unstructured and consists of undisciplined behviour the models precision dropped.

As a solution we trained the model on a new dataset. We used IDD Insaan Dataset contributed by the International Institute of Information Technolgy, Hydrebad and Intel. The dataset consists of images obtained from a front facing camera attached to a car. The car was driven around Hyderabad, Bangalore cities and their outskirts. The images are mostly of 1080p resolution, but there is also some images with 720p and other resolutions.
They propose a novel dataset for road scene understanding in unstructured environments where the above assumptions are largely not satisfied. It consists of 10,000 images, finely annotated with 34 classes collected from 182 drive sequences on Indian roads. The label set is expanded in comparison to popular benchmarks such as Cityscapes, to account for new classes.

Training

The training of the selected model took around 11hrs to train on 25GB of data on Nvidia's RTX3070 Ti GPU with 8GB GPU Memory.

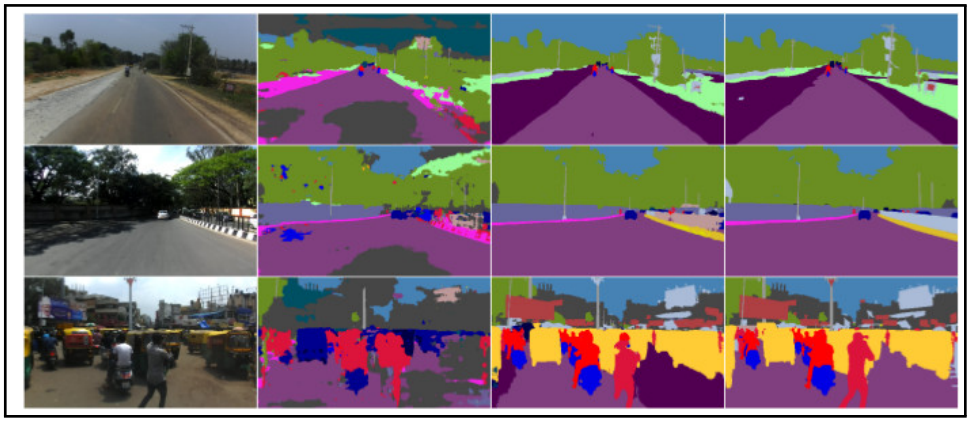

Figure 4.4: Improvement in output of models trained on IDD dataset

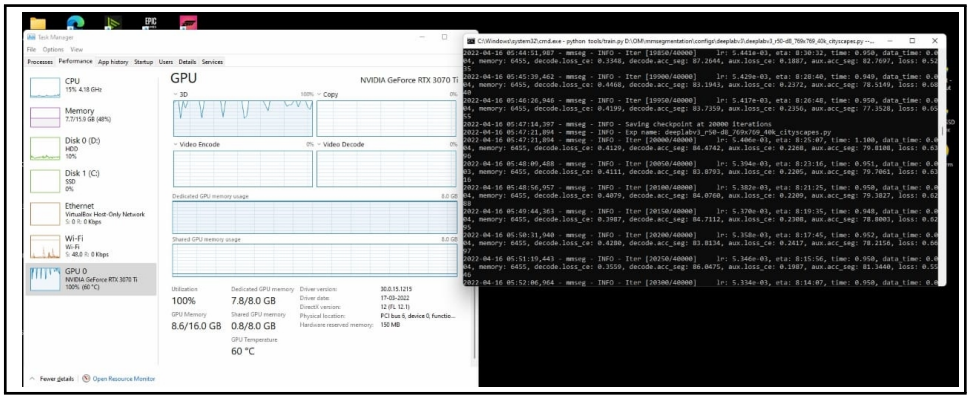

Figure 4.5: Customized Dataset Training

Before the training could start a lot of changes were made in the library files to accommodate the IDD dataset. The train script made available was only suited for the standard datasets namely Cityscapes, PASCAL VOC, ADE20K, etc. The file structure of the datasets was changed. We used the cityscapesscripts library and script to create masks from the json files made avaialble in the dataset. Due to its similarity with the cityscapes dataset we used the models based on cityscape to train our model on IDD dataset.

The train.py script is consits of an argument parser that makes it easier to send commands from the CLI rathar than editing the code again and again. The train function is based on pytorch and also has dependencies of the MMCV and mmseg libraries. The cuda appropriate version of torch is supposed to be downloaded and DiscoBox using Open MMLabs MMSegmetation as codebase we require their libraries to be able to import the apis and function calls for execution of the scripts and tools

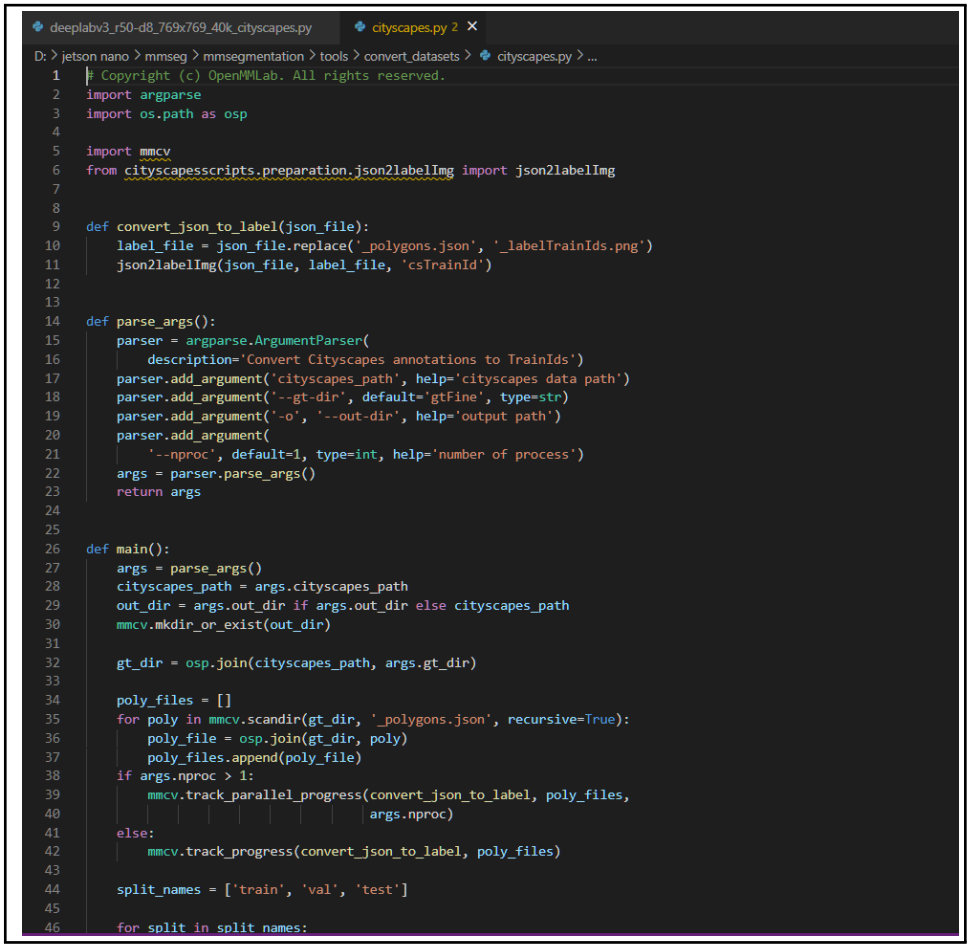

Figure 4.6: Script to convert custom dataset to cityscape supported format

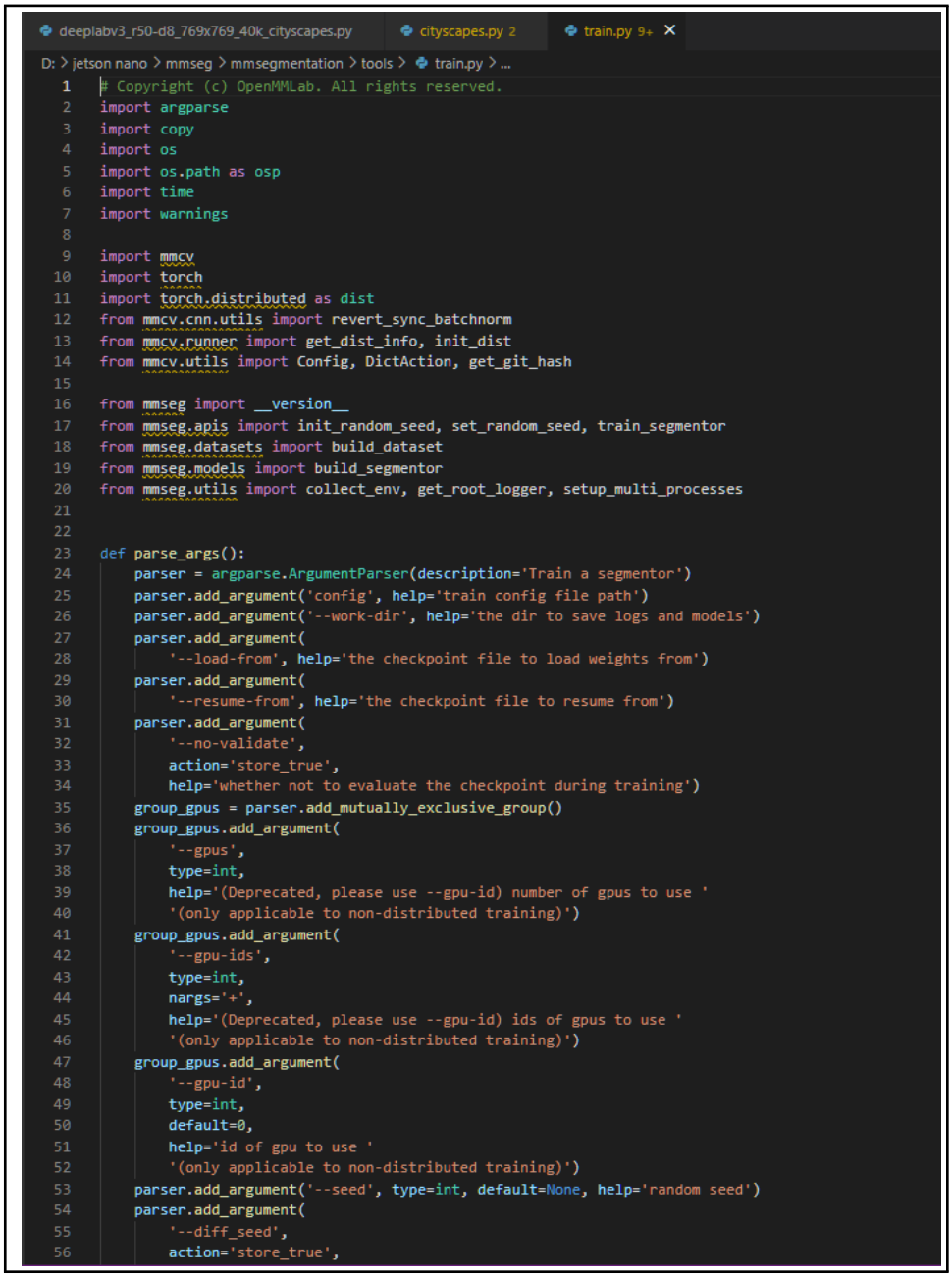

Figure 4.7: Train.py

4.3 Mobile Application

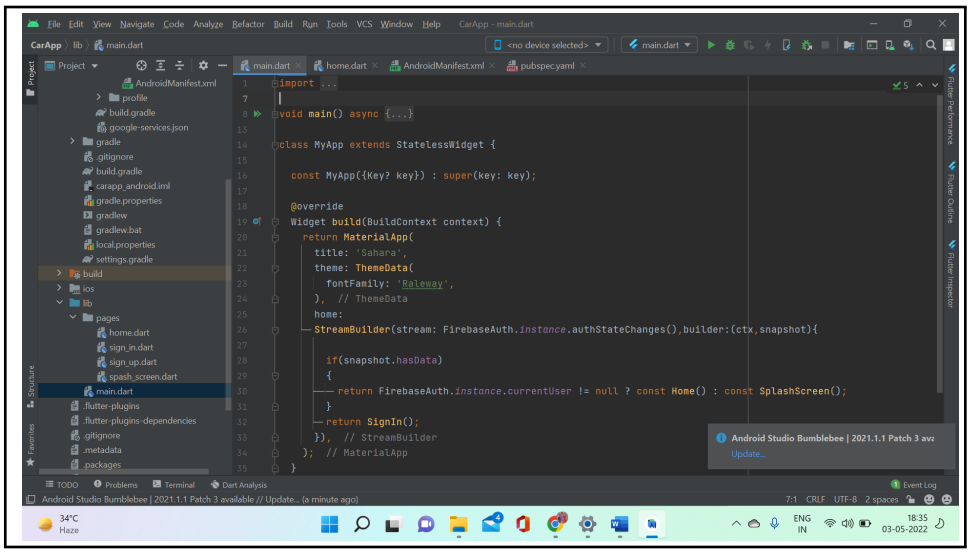

Figure 4.8: Main

For Figure 4.8 all the sub pages of the application are linked to the main.dart file

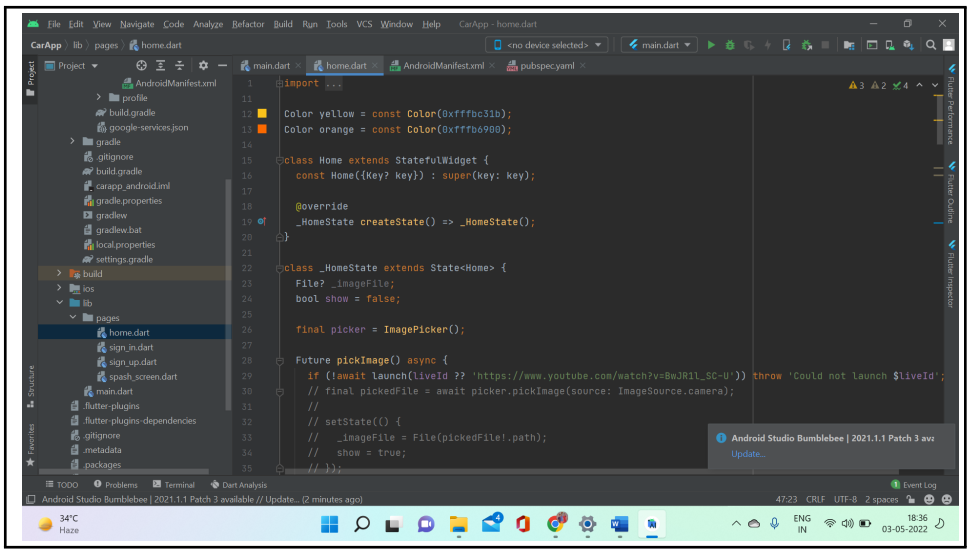

Figure 4.9: Home

For Figure 4.9 the home page consists of a start button and a stop button. As soon as the user presses the start the button the user will be able to see the live feed from camera and the car will start functioning. When the user presses the stop button then the car will stop moving and the live feed of camera will stop.

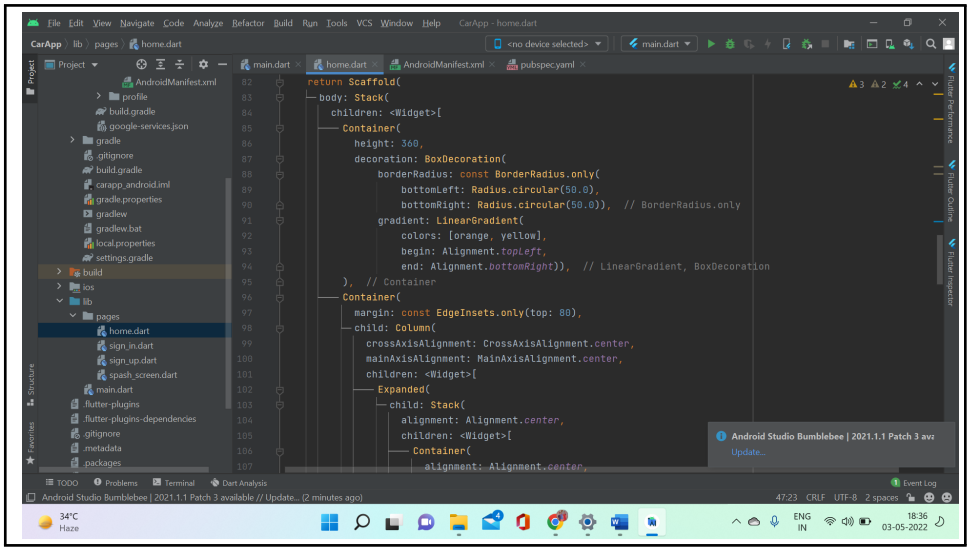

Figure 4.10: Home

The start and stop button are given to user because if the machine gets confused or doesn't function properly then the user has an control over the car.

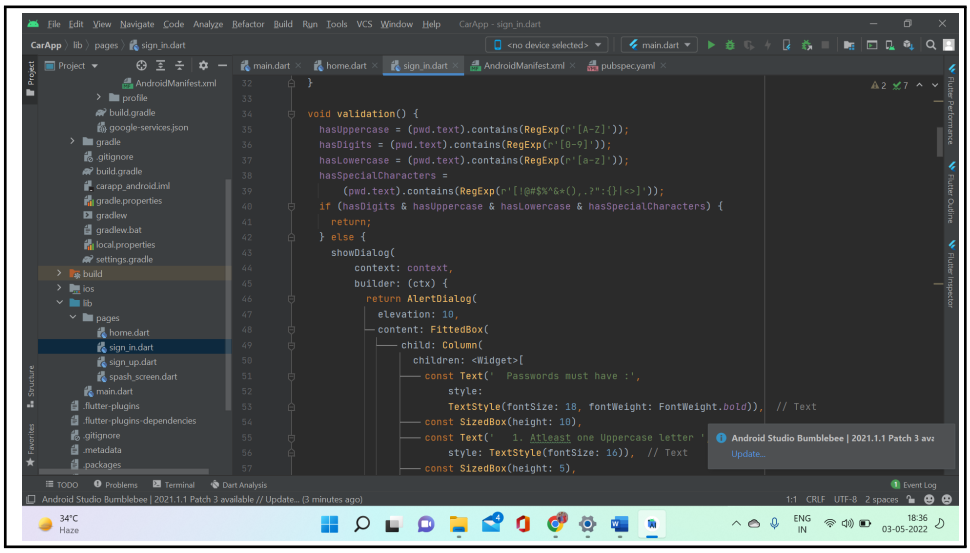

Figure 4.11: Sign-in

For Figure 4.11 that is the sign-in page in which user will be asked for its credentials and if the credentials are valid then the user will be reflected to the home page of the application.

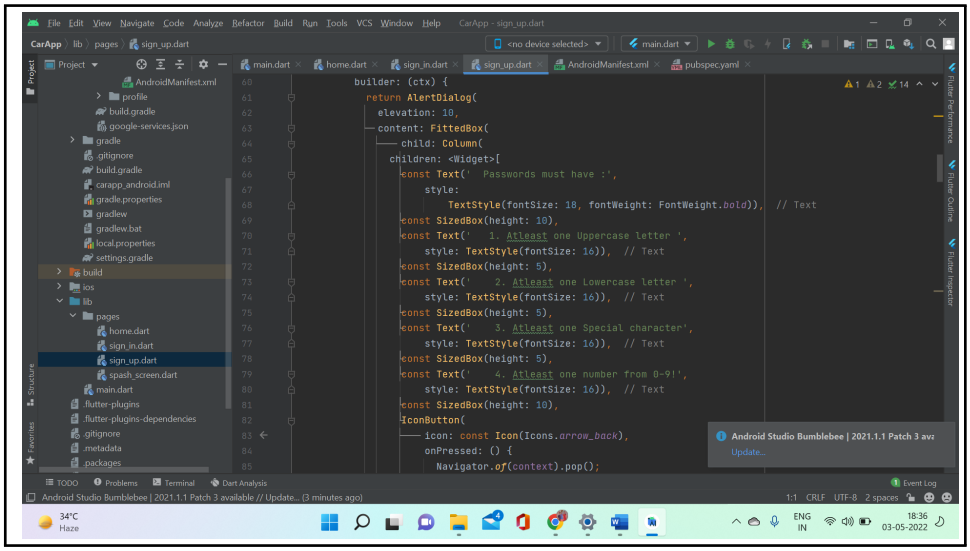

Figure 4.12: Sign-up

For Figure 4.12 that is the sign-up page in which user will be asked for different parameters which are required for sign-in page. As soon as the user enters the value for parameters and submits it, it will reflect in the Fire store Database.

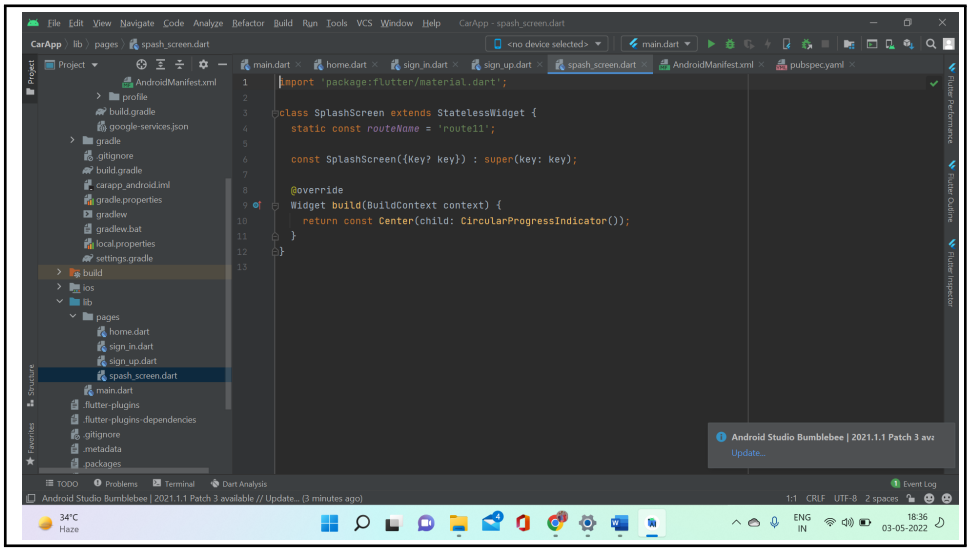

Figure 4.13: Splash-screen

For Figure 4.13 that is the splash-screen page in which is the user enters the credentials and the time till credentials gets validated, the transition which appears in the screen is coded here.

Chapter 5

Testing

5.1 Unit Testing

The individual components of were first implement module wise. The hardware and software both systems were implemented step by step and all functionalities were tested individually. Any faults were eliminated on the component level itself. All hardware components were executed with basic example like led blink, hello world, and similar programs for making sure there are no manufacturing defects in the boards. All those tests were successful and then the next programs and individual functions were tested.

On the mobile application individual modules were designed and tested against their functionalities. The functionalities were tested individually for the application. The faults for the modules were eliminated at that stage only(for example- validation of email). All the functionalities were tested rigorously and if they succeed in it then only we went to other module testing.

5.2 Integration Testing

After successful testing of individual components and functionalities, the crucial part is to integrate all the systems together. Creating a highly modular system is utmost challenging during integration of the entire system. The integration of the jetson nano board and the trained model was very time consuming and full of errors.
The problems faced during deployment on the development board was that the architecture used by the board is aarch64-linux based. These architectures have their different wheel files and installation procedure. This was the major problem faced during the integration and was identified during the testing phase while deploying on the board

5.3 Load Testing

The mechanical and hardware systems were particularly tested for this purpose. During Load testing it was understood that the motors are not capable of working and actuating the car ahead or turn, because of the weight that was being loaded on the motor. This result in the motors getting motor because of resistance and increased heat losses. Which in-turn drained the batteries quicker because of high current demand and increased discharge rate.

The batteries too became hotter over the period of time. Hence, the motors of higher rated torque to work with load of the components.

5.4 Model Testing

For comparison we tested the model on pre-trained models and our own custom trained model. The results are as follows -

Figure 5.1: Input Image

The difference between both the outputs is clearly visible and how model trained on unstructured datasets are more robust and detailed.

Figure 5.2: Custom Trained Model output

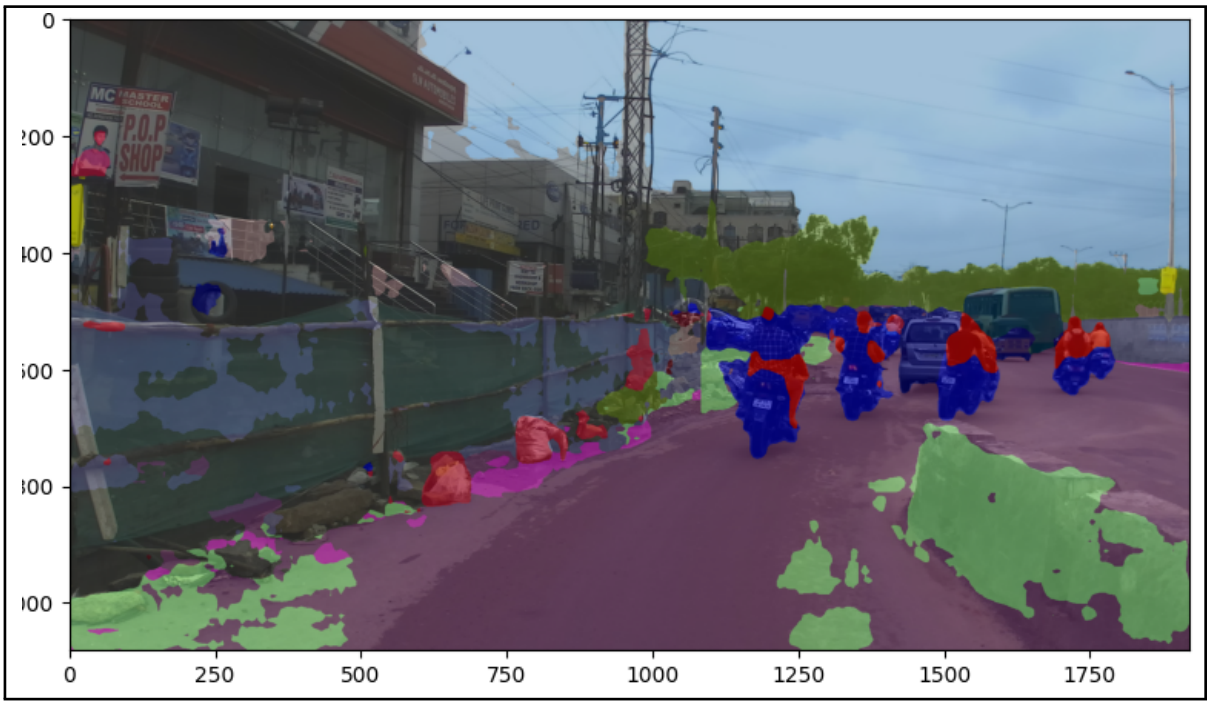

Figure 5.3: Pre-Trained Model output

Chapter 6

Result

The entire project has been result-oriented. The results that were expected from the project were to understand and implement the concepts of Autonomous Vehicles, Computer Vision, Mobile Application Development and Control System on a RC-sized Car so that it is able to navigate autonomously. These results have been achieved during the time of working on the project.
Starting with the Mobile Application the user is able to successfully sign-up, login, use the application to command the vehicle to start and stop, and view the camera feed from the vehicle on the screen. Thus, all the functionalities of the Mobile Application have successful results.

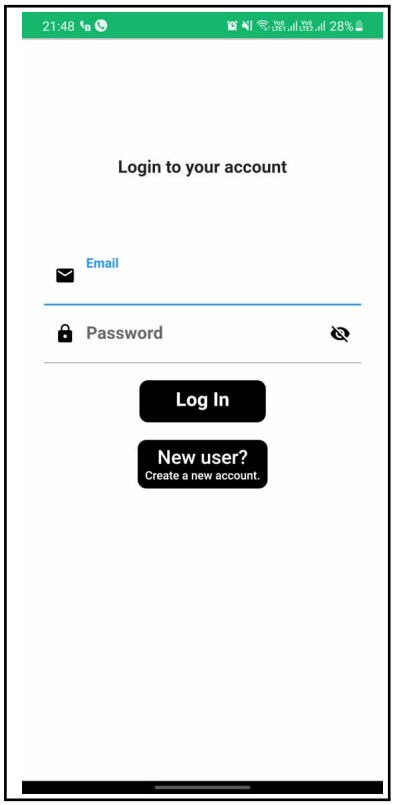

Figure 6.1: Sign-in Page

Figure 6.2: If login credentials entered are wrong

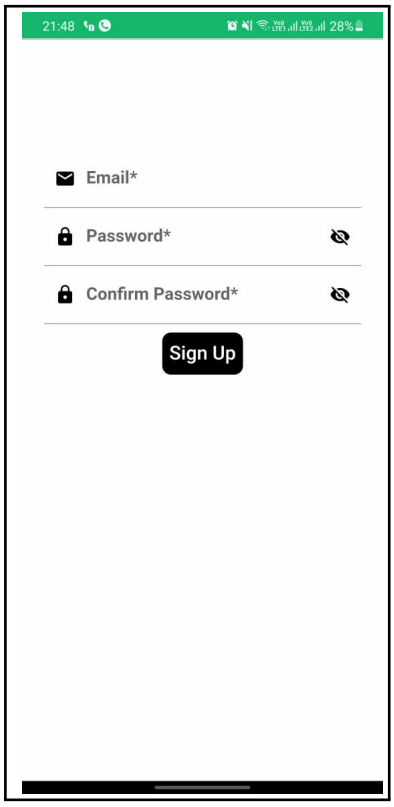

Figure 6.3: Sign-up page

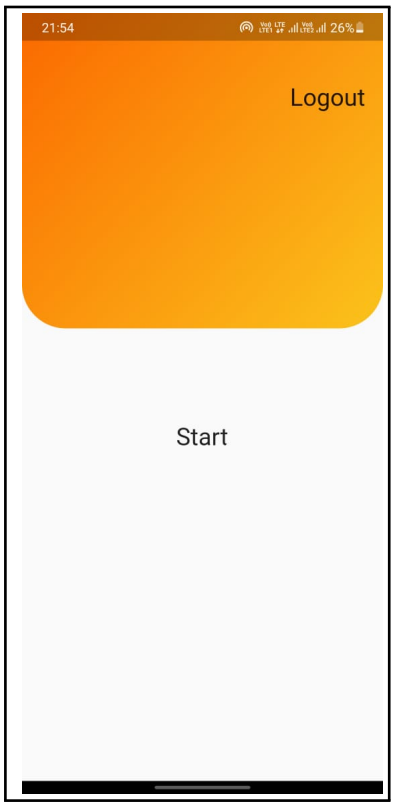

Figure 6.4: Home Screen after successful login

The custom trained model used by us in this vehicle has been implemented with the expectations of having significant advantages over standard data-sets available for a better performance in unstructured environments. Our custom trained model has resulted in a robust and similar or better performance on our target deployment area in the future i.e. India. Having a accurate model did not only improve our segmentation results, but also the results of the entire system. The decision making module has better understanding because of less noise in results, which indirectly affect the error between the actual and computed path, reducing it and, resulting in an even better overall performance.

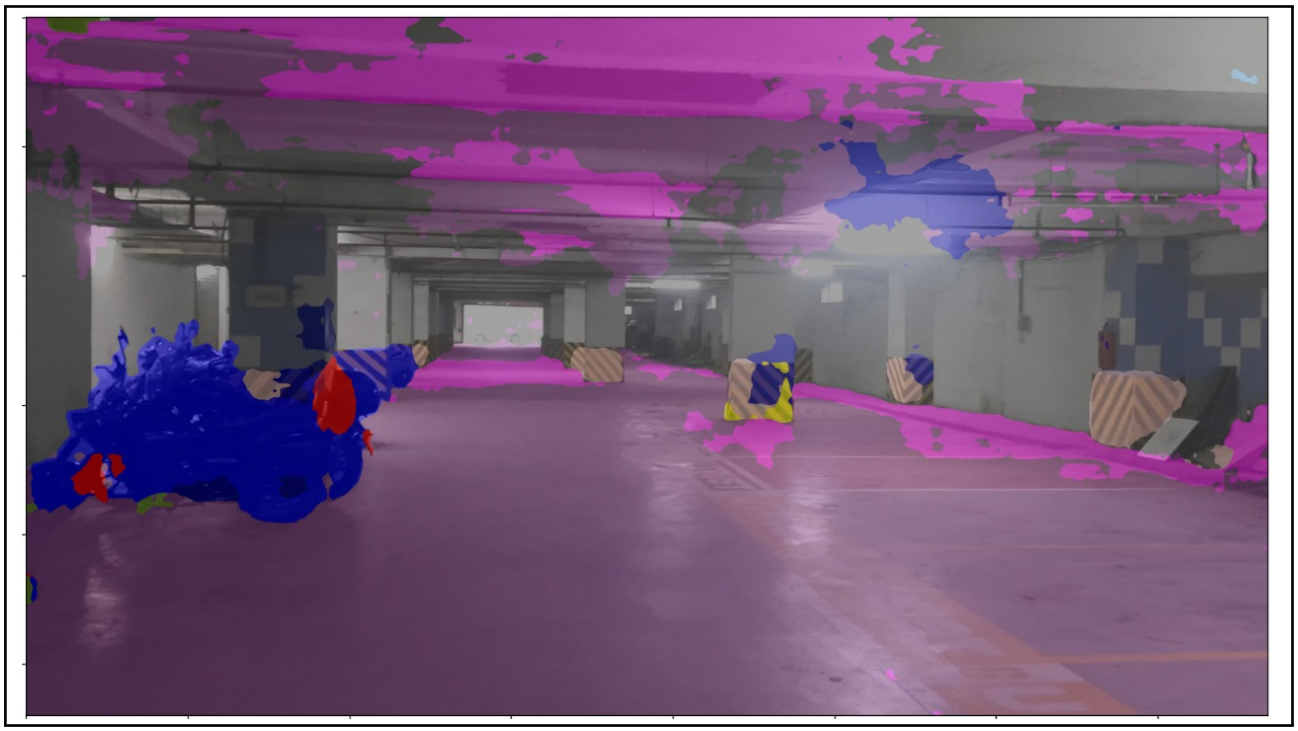

Figure 6.5: Pre-Trained Model results in college basement

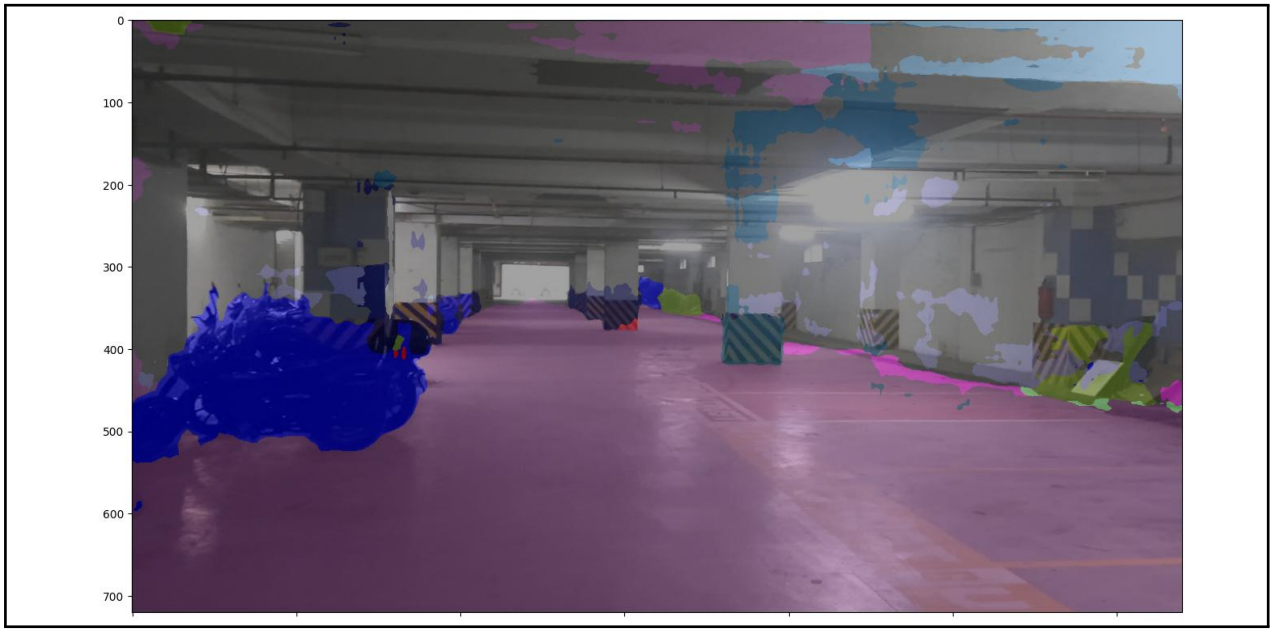

Figure 6.6: Custom Trained Model results in college basement

Figure 6.7: Custom Trained Model output

Chapter 7

Conclusions and Future Scope

7.1 Conclusion

Thus, we now understand the design and methods of implementing a fully autonomous RC-sized car. The technologies that will be required in achieving the objectives set are made clear. This project seems to be a great way in which one can learn about technologies like Embedded Systems, Machine learning, and Autonomous Vehicles as a whole. The project might seem difficult and costly to implement and orchestrate however, taking steps one at a time and careful implementation and integration of all the systems made the things easier and smooth flowing. The use of open-source techniques for the implementation of the subsystems is made clear. The literature review from different published papers gave a clear idea on the different perspectives of implementing autonomous systems. The data sets of traffic agents and different traffic scenarios form these implementations and their experiences is used in training the model and creating rules for the vehicles, making our model be prepared for many scenarios in advance and avoid under-fitting of data to these trained models.

7.2 Future Scope

- Prediction models have become a necessity, a simple autonomous riding system along with faster processing is not enough to fulfil the rising demands of today's traffic scenarios. Predicting future movements increases efficiency and reliability on the system greatly.
  Thus, in future we would like to implement a vehicle behaviour prediction model in the system. Lyft's Level 5 Open Datasets is a great way to start with. The data consists of 1185 hours recording of traffic agent movement. It uses Bird's-Eye View (BEV) rasterization for system's input, with top view of the road. The model assigns unique IDs to the detected objects and tracks them through multiple captured frames. It then predicts and plots the trajectories of these different classes of objects around the car

- Implementation on Real – Scale in APSIT's Modified Auto Club. The R742 solar electric vehicle is a perfect test-bench to test the scaling up the software. With all the resources available in the college itself.

Figure 7.1: Side-View R742

Figure 7.2: Isometric View R742

Bibliography

[1] Abror Abduvaliyev, Al-Sakib Khan Pathan, Jianying Zhou, Rodrigo Roman and Wai-Choong Wong ,"On the vital Areas of Intrusion Detection Systems in Wireless Sensor networks",IEEE Communications Surveys & Tutorials, Accepted For Publications, 2013-in press.

[2] H.H. Soliman, et al,"A comparative performance evaluation of intrusion detection techniques for hierarchical wireless sensor networks", Egyptian Informatics Journal (2012) 13, 225–238.

[3] Giannetsos Athanasios, "Intrusion Detection in Wireless Sensor Networks", Master THESIS, Carnegie Mellon University, April 8, 2008.

[4] K.Fall and K.Varadhan,"The NS Manual", http://www.isi.edu/nsnam/ns/doc/ns_doc.pdf., 1 Feb 2014.

[5] Jae Chung and Mark Claypool,"NS by Example-Tutorial", http://nile.wpi.edu/NS/overview.html , 1 Feb 2014.

[6] Network Simulator blog, http://Mohittahilani.blogspot.com , 1 Feb 2014.

[7] AWK Script for NS2, http://mohit.ueuo.com/AWK-Scripts.html , 1 Feb 2014.

Appendices

Detailed information, lengthy derivations, raw experimental observations etc. are to be presented in the separate appendices, which shall be numbered in Roman Capitals (e.g. "Appendix I"). Since reference can be drawn to published/unpublished literature in the appendices these should precede the "Literature Cited" section.

Appendix-A: NS2 Download and Installation

1. Download ns-allinone-2.35.tar.gz from http://sourceforge.net/projects/nsnam/

2. Place ns-allinone-2.35.tar in your desired directory; like /home/vishal.

3. Go to terminal and do as following commands
sudo apt-get update
sudo apt-get install automake autoconf libxmu-dev build-essential

4. Extract ns-allinone-2.35 and after extracting go to folder ns-allinone-2.35 from Terminal as
$cd ns-allinone-2.35
$./install

5. Path Setting
$ gedit .bashrc

This command will open an existing file in editor. Just put the following path which is given bellow. [Remember that our ns-allinone path is /home/vishal. we will change this path according to our ns-allinone folder's path]

export PATH=$PATH:/home/vishal/ns-allinone-2.35/bin:/home/vishal/ns-allinone-2.35/tcl8.5.10/unix/home/vishal/ns-allinone-2.35/tk8.5.10/unix

export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/vishal/ns-allinone- 2.35/otcl-1.14:/home/vishal/ns-allinone-2.35/lib

export TCL_LIRARY_PATH=$TCL_LIBRARY_PATH:/home/vishal/ns-allinone-2.35/tcl8.5.10/library

After this save and exit.

6. Now type in terminal to check that, is all command we entered in .bashrc is correct or not? And To take the effect immediately
$source .bashrc

7. Then perform the validation test using this command.
$ ./validate

8. Run ns2 using this command
$ns

We will get % prompt in our terminal. Now ns2 has been installed.

Publication

Paper entitled "RC-sized Autonomous vehicle with Prediction Model"
Authors - Fenil Bhimani, Om Bheda, Amey Mohite
Co-author - Prof. Apeksha Mohite
Conference Name - IEEE I2CT
Paper Status - Published.

NOTE:

In Project Report Project Group will have to elaborate sections included as per work done after Sem VII. It can't be same Sem VII. Report must have proper description, Elaboration of Digrams included in Design & Implementation. It can't be just Digram on Page. You will have to write there why this has been included, What it is representing. Same is Applicable to snippets of Project implementation you have included in Report. Include your publications in your Report if You have.