

" SMART PARKING USING MACHINE LEARNING "

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

OF THE DEGREE OF

BACHELOR OF ENGINEERING

In

ELECTRONICS AND TELECOMMUNICATION ENGINEERING

By

Rajiv Iyer 115A2032

Sree Ganesha Chellappa 116A2102

Navin Subbu 116A2105

Muthu Sumathy Thevar 116A2112

UNDER THE GUIDANCE OF

Prof. Hema Raut



Department Of Electronics and Telecommunications

SIES GRADUATE SCHOOL OF TECHNOLOGY

NERUL, NAVI MUMBAI – 400706

ACADEMIC YEAR

2019 – 2020

CERTIFICATE

This is to certify that the project entitled "**SMART PARKING USING MACHINE LEARNING**" is a bonafide work of the following students, submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of **Bachelor of Engineering in Electronics and Telecommunication Engineering.**

Rajiv Iyer	115A2032
Sree Ganesha Chellappa	116A2102
Navin Subbu	116A2105
Muthu Sumathy Thevar	116A2112

Guide Head of Department Principal
(Internal) (Electronics and Telecommunication) (S.I.E.S GST)

PROJECT REPORT APPROVAL

This project report entitled " **SMART PARKING USING MACHINE LEARNING** " by following students is approved for the degree of **Bachelor of Engineering** in **Electronics and Telecommunication Engineering**.

Rajiv Iyer	115A2032
Sree Ganesha Chellappa	116A2102
Navin Subbu	116A2105
Muthu Sumathy Thevar	116A2112

Name of External Examiner: _____

Signature:_____

Name of Internal Examiner: _____

Signature:_____

Date:

Place:

DECLARATION

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I 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.

Rajiv Iyer

115A2032

Sree Ganesh Chellappa

116A2102

Navin Subbu

116A2105

Muthu Sumathy Thevar

116A2112

Signature

Date:

ACKNOWLEDGEMENT

It gives us immense pleasure to thank Dr.Vikram Patil, our Principal for extending his support to carry out this project. We also thank to our Hod Prof. Preeti Hemnani for her support in completing the project. We wish to express our deep sense of gratitude and thank to our Internal Guide Prof. Hema Raut for her guidance, help and useful suggestions,which helped in completing our project work in time.

Also we would like to thank the entire faculty of Electronics and Telecommunication department for their valuable ideas and timely assistance in this project, last but not the least, we would like to thank our teaching and non teaching staff members of our college for their support, in facilitating timely completion of this project.

Project Team

Rajiv Iyer

Sree Ganesha Chellappa

Navin Subbu

Muthu Sumathy Thevar

ABSTRACT

In a country where it took 60 years to acquire 100 million vehicles but added another 100 million in just the next ten years, free and illegal parking has become both a serious urban planning and public health issue.

We are building a system in which we will be making use of live feed from CCTV cameras or pre-recorded video and detect the entry/exit of the vehicle using machine learning and Image Processing algorithms. Detection of number plate of the car is done using Optical character recognition (OCR) or Automatic License Plate Recognition (ALPR) and also store the image of vehicle in the database for ensuring security. To distinguish the entry/exit of vehicle it is determined whether motion of vehicle is towards camera or away from camera i.e. zooming in or zooming out the license plate. After retrieving the number plate, we will match it with the details of the residents stored in the local database. If there is a match then the details are logged into the real-time database and a notification will be sent to the security using the Google Firebase Cloud Messaging with timestamp of entry/exit of vehicle. If the match is not found then an alert will be sent to the security indicating arrival of new vehicle. When the authorization is done, the car driver will be shown the vacant parking lot in the premises where he can park his vehicle. The vacancy of a parking lot is checked using IoT, in which the presence of a car in a parking lot is detected using a proximity sensor/ distancing meter. The vacant spaces data is fed into the Real-time database in Google Firebase.

Contents

1	INTRODUCTION	1
1.1	MOTIVATION	2
1.2	OBJECTIVES	2
1.3	Scope	2
1.4	Organization of the report	3
2	Review of Literature	4
3	Flow of System	6
3.1	Flow Diagram	7
3.2	Methodology	8
3.2.1	Parking Space Allotment	8
3.2.2	Google Firebase	9
3.2.3	At the user end	9
4	System Requirements and Description	10
4.1	Hardware and Software Requirements	10
4.2	Hardware Description	10
4.3	Software Description	11
5	IMPLEMENTATION	12
5.1	Existing System	12
5.2	Proposed System	12
5.3	Software Implementation	13
5.3.1	License Plate Detection Using Tensorflow and OpenCv	13
5.3.2	Image segmentation and Optical Character Recognition (OCR)	17
5.3.3	Firebase Realtime Database and Storage	23
5.3.4	Parking Lot Occupancy management using IoT	24
6	Results	28
6.1	Object Detection Output	28
6.2	Database Output – Machine Learning	29
6.3	Database Output – IoT	30
6.4	Webpage Output	31
7	Conclusion and Future Scope	32
7.1	Conclusion	32
7.2	Future Scope	32

List of Figures

3.1	Flow Diagram	7
5.1	Losses	14
5.2	Tensorflow Initialization	14
5.3	Initialize Webcam and Object detection	15
5.4	Visualizations libraries	15
5.5	Object Detection and draw bounding boxes	16
5.6	Confidence calculations and capture Image	17
5.7	Output of Object Detection	17
5.8	Image Segmentation	18
5.9	Original Image	19
5.10	Grayscale Image	19
5.11	Filtered Image	20
5.12	Canny Edge Detection	20
5.13	Detected Contours	21
5.14	Contour Containing License Plate	21
5.15	7xx Masked Image	22
5.16	Extracted License plate	22
5.17	License Plate Recognised and Displayed	23
5.18	Firebase Real-Time Database	23
5.19	Firebase and WiFi credentials	25
5.20	UDM pins and Distance Calculating Function	26
5.21	One time run Loop for Setting up the system and connecting to Wifi	26
5.22	Main code	27
6.1	Detected License plate and Confidence score	28
6.2	Original image Captured	28
6.3	License Plate Recognised and Displayed	29
6.4	Firebase image store	29
6.5	Realtime database	30
6.6	When Parking Lot 1 is unoccupied	30
6.7	When Parking Lot 1 is occupied	31
6.8	web page front end	31

Chapter 1

INTRODUCTION

License plate numbers are being used to uniquely identify a vehicle. License Plate Recognition (LPR) system plays an important role in many applications like electronic payment system (toll payment and parking fee payment), to find stolen cars, traffic surveillance. For instance, in malls the time difference between the time, the car entered and left the parking lot is used for calculating the parking fee. LPR is convenient and cost efficient as it is automated. In this project we will be recognizing the license plate of the vehicles, time stamping them and carry out the prediction of their entry and exit time. When a vehicle enters the boom-barrier, license plate and make of the vehicle is automatically recognized and a database is created. On exiting, the same are validated. Now in housing societies or commercial complexes, a pre-existing database of residents/employees containing the above info along with name, parking space number, room no./company name will be used to validate. Parking spaces will be allotted to every individual at start and remaining spaces will be kept vacant for visitors. When a particular employee doesn't reach the office campus at the intended time (30-45 mins buffer time), his/her parking space will be then allotted to someone else for that day provided the parking lot is almost full. The parking spaces will consist of UDM sensors linked to NodeMcu/ESP32 Wi-Fi board that detects the presence of car parked. This will give a confirmation that the car has been parked successfully in that parking lot. When the car is moved back from the space, the slot is retained back in the system.

If a visitor approaches the boom barrier, system will perform the usual process of detection plus it will allot the parking space number displayed on the boom barrier. The feature of prediction of entry and exit time is critical for residential societies. So, if an unregistered vehicle parks anonymously or parks for out of bound limit, then the security guards will be alerted.

1.1 MOTIVATION

With growing popularity of Smart Cities, there is always a demand for smart solutions for every domain. There are multiple domains in a smart city and Smart Parking is one of the popular domain in the Smart City Parking industry has seen a number of innovations such Smart Parking Management System, Smart Ge Control, Smart Cameras which can detect types of vehicle, Smart Payment System, Smart Entry System and many more. With increase in development of affordable cars in the modern world, everyone has access to private vehicles and thus increases the requirement of free space for parking them. In a metropolitan city like Mumbai, there's a lack of free space. To overcome this issue, multi storey parking areas were created. But in coming years even this system will become inefficient. Thus, a new system is required to overcome this issue.

1.2 OBJECTIVES

The main objective here is to provide a system which provides a real time process and information of the parking slots. This paper enhances the performance of saving users time to locate an appropriate parking space. It helps to resolve the growing problem of traffic congestion such as reduced traffic as search time for parking is reduced, therefore traffic is reduced, enhanced user experience as user can easily find the parking lot which is vacant, real-time data and trend insight because using ML we can study the time of arrival and leaving of a particular car, decreased management costs because it reduces the labour cost and can be more efficient, increased safety since parking lot employees and security guards will have real-time lot data that can help prevent parking violations and suspicious activity.

1.3 Scope

It is found that in housing complexes and Office buildings, parking lots used are not occupied for the whole day. So, if a system is developed to find the duration for which the car stays in the plot, the same lot can be allocated to a visitor when it is unoccupied. Also, a physical handbook has to be maintained to record the arrival and departure of the vehicles. This increases the error probability of wrong data entries and also missing out some entries. We have proposed the smart parking system using the Internet of Things (IoT) that can be part of a solution for the parking problem. This system help in organizing the parking lot and helps the driver to reach their parking spots easily as they known which space is vacant. The parking space can be detected using an Infrared sensor that connects to the ESP12-E (NodeMCU) module that was programmed through Arduino IDE. This system will help security to know the availability of parking space in real time.

1.4 Organization of the report

Chapter 1 briefs about the problem statement describing the parking problems and how it affects the serious urban planning and how certain technical solutions and their application could help in reducing the search traffic. The scope of the proposed system is also discussed to understand the practicality of the solution and certain applications.

Chapter 2 is about understanding various solutions that preceded to address this problem statement and understanding its efficiency. More or less a dozen research paper references were studied to understand, highlight and compare the uniqueness of each solution presented by the respective authors. Our perspective on each paper is presented without obscuring important details.

Chapter 3 is all about our perception of the initial solution thought out to the problem statement. Understanding the flow of the system succeeding after deciding an efficient flow diagram and methodology i.e. our steps to our approaching a practical solution and a prototype to back it up. The nature of this application is network based - wireless network and our take on the new perspective of the solutions from the literature survey and the user end input.

Chapter 4 deals with all the technical aspects that lie beneath the theory of the solution proposed with hardware and software aspects involved and why we chose the same, keeping in mind the basic requirements that were required.

Chapter 5 explains in brief the working of the system while explaining the concepts involved in all the electronic components involved and how they act cohesively in the proposed solution. Understanding the harmony between the hardware functionality with the software backing the operation. The primary component i.e. the NodeMCU ESP 8266, its working, specifies its necessity for our project along with other components like a microcontroller and other passive components used.

Chapter 6 summarises the working and the results of the tests performed and discussions regarding the implementation details and the analysis of the performance measures

Chapter 7 discusses more on the practicality and feasibility of the solution. The prototype of our project used a lot of sensors and microcontrollers-without an efficient power supply - taking into consideration that we focussed more on the “feedback control” loop that made our solution unique and if focussed more on the research and development, the project hardware and its intricate details and efficiency would be improved.

Chapter 2

Review of Literature

Giuseppe Amato from ISTI-CNR [1] and his colleagues developed a smart parking system using hardware and software based on IoT , and mobile application which is used by the driver can easily check parking information and use mobile transactions to pay the parking fee. Here they have used real time car occupancy detection which uses convolutional neural Network (CNN) which is running on-board of a smart camera with limited resources. The results show that it is robust to light condition changes and even from shadows. Here detection of car parking occupancy is done by deep learning. they exhibit very high accuracy, even in presence of noise. This method is also robust to non-standard parking behaviors, such as cars occupying multiple parking lots. In that case, our solution tends to classify both slots as busy. The goal of our study is to improve the parking process by reducing the time that is required to park a car.

Rachapol Lookmuang's [2] experiments show that Convolutional Neural Networks (CNN) are effective enough to be applied for finding the vacant parking lots using computer vision. The accuracy of the network increases with no of images used for training the model. They have tested their approach using a test set produced with a camera placed in a different place (perspective and viewpoint) and their approach reduces errors caused by natural factors such as light etc. This real time parking system can not only tell if a parking space is available or not but it can all locate the free parking space for the car to be parked. In this system we can also pay the parking fee through mobile which reduces the time for standing in queue and paying the fee for parking. The system will detect the vehicle plate number and use it to inform the driver where his/her car is parked and also for the purpose of security monitoring. The system works accurately even where the parking area have low light. They concluded that CNNs have good generalization capabilities in predicting parking status when tested on a dataset completely different from the training dataset.

Leonardo Dominguez [3] used an open source platform is taken as base and by improving it to evaluate only objects that are moving. The algorithms run on a distributed platform that operate multiple cameras and sensors, to support security management. The initial outputs gave partially good performance, reaching near a real-time LPR detection, even for high resolution images. On the other side, the true positives are less than expected. management. The first results are reasonable in performance, reaching near a real-time LPR detection, even for high resolution images. The first results are reasonable in performance, reaching near a real-time LPR detection, even for high resolution images. The

algorithms required to have the plates more focused even if the text is readable, so they propose using smarter algorithms for better output.

Thanh Nam Pham [4] introduces a novel algorithm that increases the efficiency of the current cloud-based smart-parking system and develops a network architecture based on the Internet-of-Things technology. This paper proposed a system that helps users automatically find a free parking space at the least cost based on new performance metrics to calculate the user parking cost by considering the distance and the total number of free places in each car park. Cost can be reduced by saving time which will reduce emission of fuel and it will save the environment. This system realizes that if we use the percentage of free spaces in each car park as a parameter for planning with regard to forwarding the users, the waiting time of the user for the service will be greatly reduced compared with that in an ordinary network. Thus this system is an efficient system to reduce time and money at the same time

Abhishek Kashyap [5] proposed Automatic Number Plate Recognition system is used for many purposes like tollway authorities uses this system for allowing the vehicle to enter the toll road by detecting their number plate automatically. The concept of ANPR system is based on the matching of templates and exactness (result) of this system was established as 75-85 percent for Indian number plates. model of this gadget can be carried out by way of capturing pictures from stationery clip and choosing the great car border for category of vehicles and spotting the quantity plates the use of neural networks

Chapter 3

Flow of System

3.1 Flow Diagram

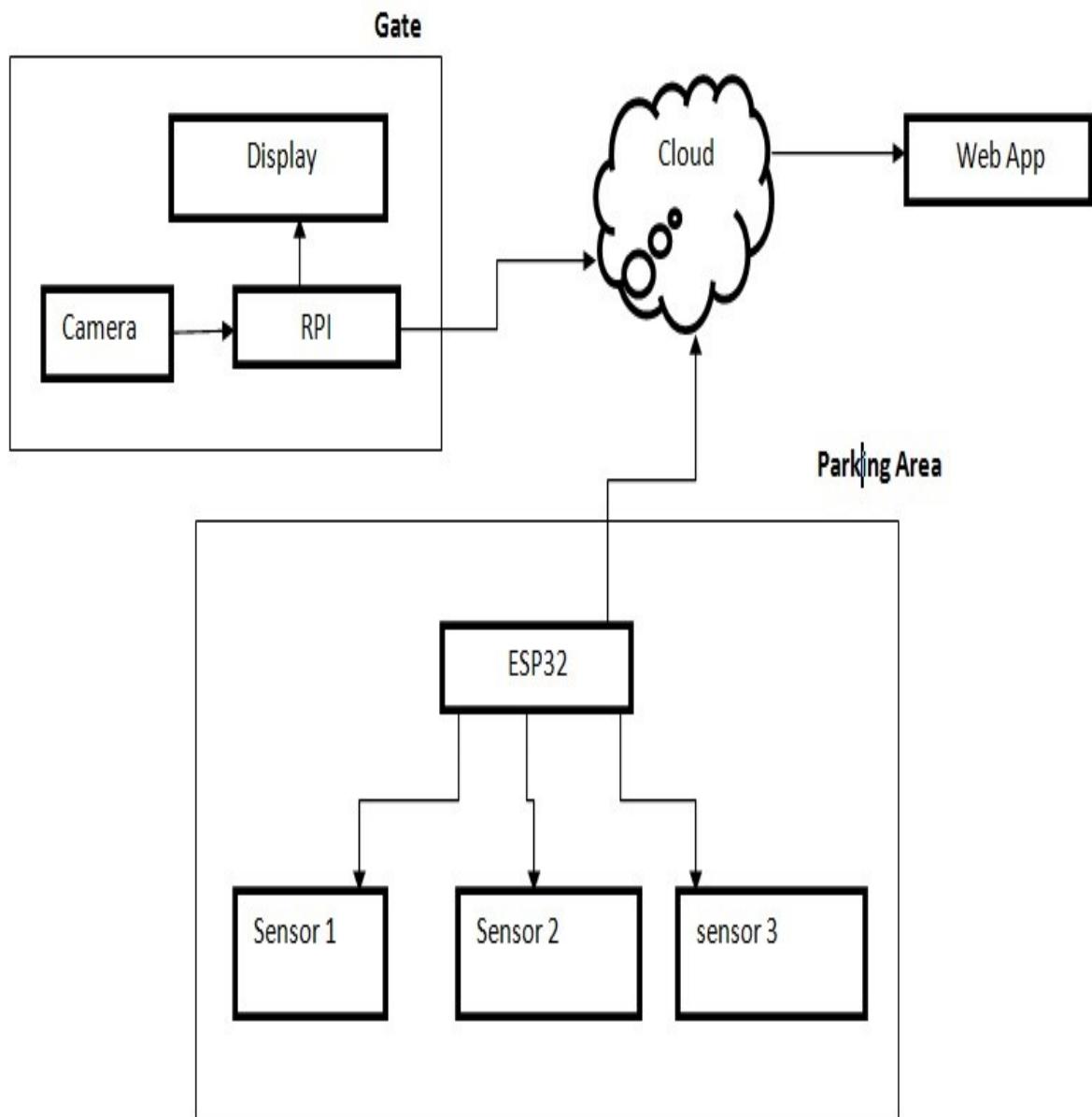


Figure 3.1: Flow Diagram

3.2 Methodology

This project aims to take all the efficient aspects of the projects proposed in the literature survey and integrate it with an efficient feedback to be part of a solution for the parking problem.

When the car enters the gate of the parking space the camera installed in front of the gate detects the car. We have trained the system to identify the license plate by using Tensorflow. Tensorflow is a free and open source software library for differentiable and dataflow programming across various range of task. It is basically used for machine learning application. Thus by using this we train our model to detect license plate from a live video.

In order to detect the license plate, first we will resize the original image to the required size making sure that the number plate remains in the frame after resizing. Then convert the image to gray scale which usually speeds up the following process and also no need to deal with the colour details as it is not required when processing an image. It is necessary to remove the unwanted details of the image as our main focus is to find license plate. Hence, useless information (noise) can be removed by using bilateral filter (blurring). Next step is to perform Edge detection which displays only the edges that have the intensity gradient between the minimum and maximum threshold values. This can be done using the canny edge method. Once the edge detection is done, we need to find the closed surface area in the image in order to extract the license plate for which contours need to be differentiated. While finding the contours we may obtain several results, from which only 10 contours will be filtered. These results may consist of closed surface contours which involves license plate number since it is a closed surface. So we filter those contour images by setting it up in a loop and simultaneously check for a rectangle shape (four sided closed figure). Once detected the contour is saved in a variable and a rectangle box is drawn around the license plate. Now after detection of the license plate, the remaining unnecessary information can be removed by masking the entire image except the area of the number plate. Basically Character Segmentation is done where we will be cropping out the license plate from the image. After identifying the license plate and extracting the license plate from the image, extraction of the characters from the license plate will be processed using OCR (Optical character recognition). OCR initially removes unnecessary parts of the image by converting the colour image into black and white image. Thus the image will be left with black and white text. In the next step OCR does character recognition by scanning pixel by pixel and comparing it with a set of database stored. Thus giving us the characters of the number plate.

3.2.1 Parking Space Allotment

Using ML (Machine Learning) arrival time of resident/employee car in the parking space will be predicted. Initially the residents/employees will be provided a particular lot in the parking area. If the resident/employee doesn't come on time, his/her parking lot will be given to a outsider/visitor. This process is called time stamping. A timestamp is the current time of an event that is recorded by a computer. Thus the computer maintains

accurate current time, calibrated to minute fractions of a second. By studying the arrival and exit time of the car we can check whether the parking lot is vacant or not. As soon as the car enters parking space, the ultrasonic sensor which is used to detect if the parking slot is available or occupied, sends the data to ESP8266 accordingly. If the parking space is occupied it sends ‘1’ and if the space is vacant it sends ‘0’. Simultaneously when the car enters the parking space, arrival time is recorded. This information along with license plate number is stored in the database. Similarly the exit time of the car is also noted and updated in the database.

3.2.2 Google Firebase

Firebase provides a real-time database and backend as a service. Firebase in this project is used to store the data of prototype. Firebase here acts as a cloud. Firebase cloud is a NoSQL document database that lets to easily store, sync, and query data for our web apps. Firebase here is also used as a web application development platform.

The characters obtained from OCR will be stored in the Real time database via RPI microcontroller which is connected to Wi-Fi system. Along with the license plate number, details such as date and time of the car arrival and exit will also be recorded.

We will send data to webserver for looking up the availability of space for vehicle parking. Here we are using firebase as lot database to get the parking availability data. For this we need to find the Firebase host address and the secret key for authorization.

3.2.3 At the user end

--

--

Chapter 4

System Requirements and Description

4.1 Hardware and Software Requirements

To implement the proposed system, it was required that we need electronic components and a suitable bridge to connect us to the medium processing the data i.e. the software and the hardware. Selecting any random microprocessor would affect the budget of the project – but at the same time considering the efficiency and power handling capacity required us to shortlist nodemcu esp8266 after considering its specifications from the data sheet. For the software requirements we needed a suitable Ide to process all the data.

4.2 Hardware Description

Raspberry Pi Camera: This 8mp camera module is capable of 1080p video and still images that connect directly to your Raspberry Pi. This is the plug-and-play-compatible latest version of the Raspbian operating system, making it perfect for time-lapse photography, recording video, motion detection and security applications. Connect the included ribbon cable to the CSI (Camera Serial Interface) port on your Raspberry Pi.

Raspberry Pi: Raspberry Pi is an open source platform for IoT and Credit Card size computer. It is used for projects where it needs low power computing and compact system design. It has its own 32bit microprocessor and all other components required for a computer in a compact design.

Ultrasonic distance measure: An Ultrasonic Sensor is a device that measures distance to an object using Sound Waves. It works by sending out a sound wave at ultrasonic frequency and waits for it to bounce back from the object. Then, the time delay between transmission of sound and receiving of the sound is used to calculate the distance.

NodeMCU ESP 8266:

The central unit used for collecting the sensor output pulses and processing the values, a nodemcu esp8266 is used. The particular hardware was chosen due to cost efficiency and the ability to transmit data over one of its four Wi-Fi modes and connect over LAN. The faster Wi-Fi allowed us to use the data to transmit over a web-interface. Its digital pins are used to take the input from the sensor since the output from the sensor is in the form of pulses. Connected via USB, providing the power supply to the board allowed us to test the computation and analysis of the data. The Access point (AP) mode of the board allowed us to use one of the boards as the central node.

4.3 Software Description

Anaconda Navigator: Anaconda Navigator is an Open Source graphical user interface for conda package and environment manager. It is simple Software interface to work with different platforms in Python.

Spyder: The Scientific Python Development Environment in short Spyder is a powerful scientific environment included in the Anaconda Navigator. This is an environment to code and perform them.

Tensorflow: Tensorflow is an open source Machine Learning and Neural Network platform. With flexible tools and ecosystems and a huge community, it is currently leading in the field of Machine Learning. It provides algorithms and models for ML. there are many pretrained models and also gives facilities for custom model training.

Arduino Ide: The Ide used for the microcontroller nodemcu esp8266 is an Arduino Ide where the basic code for the amount of water flowing through the tap using the flow rate is written. For various nodemcu(s) it was required that the code of same logic be copy-pasted to ensure the basic value that it can measure using different sensors at a given time frame without producing any unnecessary delays in the processing of the system and compiling and uploading the program occupying an efficient amount of memory space.

Web App: The web-interface was designed using various languages and integrating it on notepad ++. The languages used are JavaScript, CSS and Google firebase. The web-interface was modified from the template and the program and logic for the values to be processed and displayed was added in the sections of the web-interface's source program where we can handle all the back-end operations and provide the user with the data he needs.

Chapter 5

IMPLEMENTATION

5.1 Existing System

The existing smart parking is built using an ultrasonic sensor to detect vehicle presence in the allocated lot for the car. When car reaches the gate a sensor senses the presence of a car it opens the gate automatically. The ESP8266 NodeMCU will be used here as the main controller to control all the peripherals attached to it. As the car enters it increments the number of cars in the parking space, and when the car exits it decrements the number of car present. There is no time stamping used in this system.

5.2 Proposed System

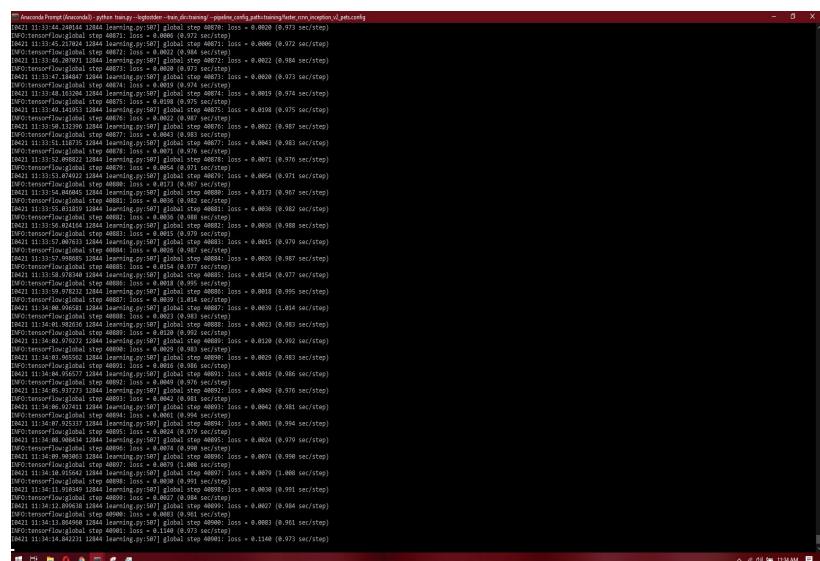
Our main purpose is to build a parking system to save time and to increase security. Our parking system has an organised way to allot resident/employee a permanent parking space. People waste time in finding parking spots to park their car so we can created a system to create an efficient system to overcome this time barrier. when the car enters the parking area the camera detects the number plate noting the time of entry. There will be a database where there will be details about the car owner if he is an resident/employee. If an outsider wants to park the car the system will check for free parking space. We can increase security by using Time Stamping. By using Time Stamping we can predict the resident/employee in and out time from the parking space. If the resident/employee doesn't arrive at that particular period of time the parking space can be held free for an outsider. If the outsider is parked the car for more time than allotted to him an alert message is sent. Thus increasing security and time saving is done.

5.3 Software Implementation

5.3.1 License Plate Detection Using Tensorflow and OpenCv

When a live video is captured in a video camera, usually the whole video is being recorded. This increases the need for more hardware storage. To overcome this, we have used Machine Learning. Using Tensorflow we trained a custom object detection model for detecting the object, in this case the license plate of a car and capture an image. On comparing R-CNN, Fast R-CNN, Faster R-CNN, SSD Mobile Lite and YOLO, we concluded that for our purpose Faster R-CNN was optimal. It was considered as it has better accuracy and speed. The hardware requirement for this is not much, most of the modern computers with a decent GPU will perform well. Faster R-CNN has 3 levels of neural network, Feature Network, Region Proposal Network (RPN), Detection Network. Feature Network is a pretrained classifier which generates good features from the image. RPN is a 3-layer convolutional network, it generates bounding boxes around places with high probability of object being present. Detection Network also known as RCNN network takes input from both the above layers and generates the final class and bounding box. Both RPN and Detection Network needs to be trained.

OpenCV is used to stream a live video from the video camera installed on the boom barrier at the entrance. Usually the whole video is being recorded for security purposes but recording vehicle entry and exit is not much of a use as it'll need more hardware storage which can be used for other purpose. To overcome this, we have used Machine Learning. Faster R-CNN is used for our purpose. The video captured is processed through this model, detection algorithms find the most probable locations of objects and draw bounding boxes. Each box has scores of probabilities, the overlapping boxes are eliminated based on the scores. The box with highest score is retained and others are destroyed. If the object is found, the bounding box that shows the objects passes the coordinates and the score which are then displayed in green on the live video. When the score of the detected object is greater than the desired value, an image is captured with the timestamp of when it was captured and stored in a local storage. Refer figures (ML codes) for the full code



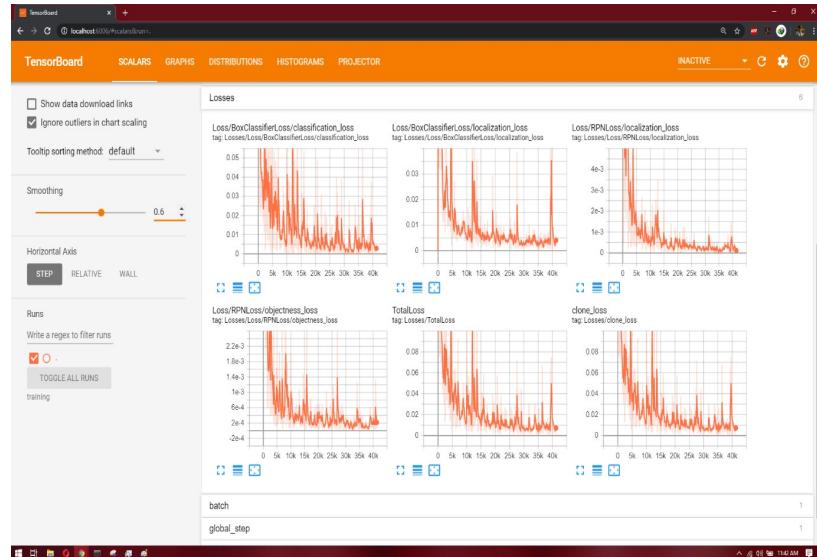


Figure 5.1: Losses

Figure 5.2: Tensorflow Initialization

```

77 # Initialize webcame feed
78 video = cv2.VideoCapture(0)
79 ret, video.set(cv2.CAP_PROP_FRAME_SIZE, 1280)
80 ret = video.set(3,720)
81
82 while(True):
83
84     # Acquire frame and expand frame dimensions to have shape: [1, None, None, 3]
85     # i.e. [1, H, W, C]
86     # cv2.cvtColor expects image in BGR format, so we convert the frame from RGB to BGR
87     ret, frame = video.read()
88     frame_bgr = cv2.cvtColor(frame, cv2.COLOR_RGB2BGR)
89     frame_expanded = np.expand_dims(frame_bgr, axis=0)
90
91     # Perform the actual detection by running the model with the image as input
92     # Inference is done on a single image
93     [detection_boxes, detection_scores, detection_classes, num_detections],
94     feed_dict={image_tensor: frame_expanded})
95
96     # Draw a bounding box around the objects that were detected
97     capture,frame,confidence = vis_util.visualize_boxes_and_labels_on_image_array(
98         frame,
99         detection_boxes,
100        detection_scores,
101        detection_classes,
102        num_detections,
103        category_index,
104        use_normalized_coordinates=True,
105        line_thickness=4,
106        min_score_thresh=0.6)
107
108     # Here we check if the required Confidence value and if it is satisfied
109     # Image is Captured with Time and OCR is processed, followed by data storage in
110     # database
111
112     if confidence > 90 :
113
114         print("Object Found")
115         count = count + 1
116         print(count)
117         if count == 1:
118
119             # Time of Detection
120             time = datetime.datetime.now()
121             print(time)
122             current_time = time.strftime("%d-%m-%Y %H:%M %p")
123             local_time = time.strftime("%d-%m-%Y %H:%M %p")
124             print(current_time)
125             local_image_path = "CAP_{0}.png".format(current_time)
126             img_name = "(0)_{0}/CAP_{1}.png".format(IMAGE_SAVE_PATH,local_image_path,time_image)
127             cv2.imwrite(img_name,frame)
128             print("written-{0}-formatting_name")
129
130             # Storing in Firebase
131             timestamp = time.strftime("%d-%m-%Y %H:%M:%S")
132             db_text = timestamp
133             fb_token = firebase.database().reference()
134             fb_token.push(db_text)
135             fb_token.firebaseio_store(local_image_path,local_image_name)
136
137         else:
138             print("No")
139             count = count + 1
140
141         cv2.imshow('Object detector', frame)
142
143         if cv2.waitKey(1) == ord('q'):
144             break
145
146     # Clean up
147     video.release()
148     cv2.destroyAllWindows()
149
150
151

```

Figure 5.3: Initialize Webcam and Object detection

Figure 5.4: Visualizations libraries

```

77     def draw_bounding_box_on_image(image,
78                                     ymin,
79                                     xmin,
80                                     ymax,
81                                     xmax,
82                                     color="red",
83                                     thickness=4,
84                                     display_str_list=[],
85                                     use_normalized_coordinates=False):
86
87     """Adds a bounding box to an image.
88
89     Bounding box coordinates can be specified in either absolute (pixel) or
90     normalized coordinates by setting the use_normalized_coordinates argument.
91
92     Each string in display_str_list is displayed on a separate line above the
93     bounding box. A single space character separates the box from the strings.
94     If the top of the bounding box extends to the edge of the image, the strings
95     are displayed below the bounding box.
96
97     Args:
98         image: a PIL.Image object.
99         ymin: ymin of bounding box.
100        xmin: xmin of bounding box.
101        ymax: ymax of bounding box.
102        xmax: xmax of bounding box.
103        color: color to draw bounding box. Default is red.
104        thickness: thickness of the lines drawn. Default is 4.
105        display_str_list: list of strings to display in box
106        use_normalized_coordinates: If True (default), treat coordinates
107            as normalized relative to the image. Otherwise treat
108            coordinates as absolute.
109
110    Returns:
111        image: a PIL.Image object.
112        If use_normalized_coordinates:
113            (left, right, top, bottom) = (xmin * im_width, xmax * im_width,
114                                         ymin * im_height, ymax * im_height)
115            else:
116                (left, right, top, bottom) = (xmin, xmax, ymin, ymax)
117            draw.line([(left, top), (right, top), (right, bottom),
118                      (left, bottom)], (left, top), width=thickness, fill=color)
119        try:
120            font = ImageFont.truetype("arial.ttf", 24)
121        except IOError:
122            font = ImageFont.load_default()
123
124        # If the total height of the display strings added to the top of the bounding
125        # box exceeds the top of the image, stack the strings below the bounding box
126        display_str_heights = [font.getsize(ds)[1] for ds in display_str_list]
127        total_display_str_height = (1 + 2 * 0.05) * sum(display_str_heights)
128
129        if top > total_display_str_height:
130            text_bottom = top
131        else:
132            text_bottom = bottom + total_display_str_height
133
134        margin = np.ceil(0.05 * text_height)
135        for display_str in display_str_list:
136            text_width, text_height = font.getsize(display_str)
137            margin_width = np.ceil(0.05 * text_height)
138            draw.text((left + margin, text_bottom - text_height - margin),
139                      display_str, font=font,
140                      fill=color)
141
142        draw.line([(left + margin, text_bottom - text_height - margin),
143                  (left + margin, text_bottom - margin),
144                  (left, text_bottom - margin),
145                  (left, text_bottom)],
146                  fill="black",
147                  thickness=thickness,
148                  text_bottom - text_height - 2 * margin
149
150
151

```

```

151     def draw_bounding_box_on_image_array(image,
152                                         ymin,
153                                         xmin,
154                                         ymax,
155                                         xmax,
156                                         color="red",
157                                         thickness=4,
158                                         display_str_list=[],
159                                         use_normalized_coordinates=True):
160
161     """Adds a bounding box to an image.
162
163     Bounding box coordinates can be specified in either absolute (pixel) or
164     normalized coordinates by setting the use_normalized_coordinates argument.
165
166     Args:
167         image: a numpy array with shape [height, width, 3].
168         ymin: ymin of bounding box.
169         xmin: xmin of bounding box.
170         ymax: ymax of bounding box.
171         xmax: xmax of bounding box.
172         color: color to draw bounding box. Default is red.
173         thickness: thickness of the lines drawn. Default is 4.
174         display_str_list: list of strings to display in box
175         use_normalized_coordinates: If True (default), treat coordinates
176             as normalized relative to the image. Otherwise treat
177             coordinates as absolute.
178
179         Returns:
180             image: a numpy array with the same dimensions as input(image), containing (modified) image.
181             draw_bounding_box_on_image(image,ymin,xmin,ymax,xmax,color,
182                                         thickness,display_str_list,use_normalized_coordinates)
183
184     np.copyto(image, np.asarray(image))
185
186     def visualize_boxes_and_labels_on_image_array(
187         image,
188         boxes,
189         classes,
190         scores,
191         labels,
192         index_to_color,
193         instance_masks=None,
194         instance_boundaries=None,
195         keypoints=None,
196         use_normalized_coordinates=False,
197         line_thickness=4,
198         max_boxes_to_draw=20,
199         min_score_thresh=.05,
200         line_color="blue",
201         skip_scores=False,
202         skip_labels=False):
203
204     """Overlays labeled boxes and labels on an image with associated scores and label names.
205
206     This function overlays labeled boxes and labels on the input image and
207     creates a display string for each detection and overlays these
208     strings on the image. This function modifies the image in place, and returns
209     that same image.
210
211     Args:
212         image: uint8 numpy array with shape (img_height, img_width, 3)
213         boxes: a numpy array of shape [N]. Note that class indices are 1-based,
214               while category names are 0-based. This array contains bounding box
215               coordinates and associated confidence (score) values.
216               For each category, one box should be drawn.
217         scores: a numpy array of shape [N] or None. If scores=None, then
218             this function will only draw boxes. If scores are provided, this
219             function will also draw text labels above each box that
220             corresponds to the score of that box.
221         labels: a list of category names. These names are expected to be
222             in categories_map['name']. Category names are 0-based.
223         index_to_color: a dict mapping categories to colors. This dict must have
224             the same keys as categories_map['name'].
225         instance_masks: a numpy array of shape [N, image_height, image_width]
226             with binary masks corresponding to the boxes that have
227             non-zero scores. The mask has 1's where the object is
228             present and 0 where it's not.
229         instance_boundaries: a numpy array of shape [N, image_height, image_width]
230             with binary boundaries corresponding to the boxes that have
231             non-zero scores. The boundary is 1 where the object
232             is present and 0 where it's not.
233         keypoints: a numpy array of shape [N, num_keypoints, 2] containing
234             keypoint locations in (x, y) format. Note that these are
235             normalized coordinates and not absolute coordinates. If
236             non-keypoints are passed, they are ignored.
237         use_normalized_coordinates: whether boxes is to be interpreted as
238             normalized coordinates or not.
239         line_thickness: thickness of the lines drawn.
240         max_boxes_to_draw: maximum number of boxes to draw on an image.
241         min_score_thresh: threshold minimum score threshold for a box to be visualized
242         alpha: transparency factor for drawing overlapping boxes. For
243             instances with overlap greater than this value, the box with
244             higher score is drawn over the other.
245         line_color: color (hex code) of lines drawn on image
246         skip_scores: whether to skip score when drawing a single detection
247         skip_labels: whether to skip label when drawing a single detection
248
249     Returns:
250         uint8 numpy array with shape (img_height, img_width, 3) with overlaid boxes.
251
252

```

Figure 5.5: Object Detection and draw bounding boxes

```

  # Create a display string (label + color) for box location, group any boxes
  # from the display str map = collections.defaultdict(list)
  box_to_display_str_map = collections.defaultdict(list)
  # box_to_instance_masks_map = collections.defaultdict(list)
  box_to_instance_masks_map = {}
  # box_to_keypoints_map = collections.defaultdict(list)
  box_to_keypoints_map = {}
  confidence_map = {}
  min_scores_to_draw = 0.0
  max_boxes_to_draw = boxes.shape[0]
  for i in range(max_boxes_to_draw):
    if scores[i] < None or scores[i] < min_score_thresh:
      continue
    # Capture image copy
    capture, image = capture_and_image(i)
    boxes = boxes[i].copy()
    class_name = category_index.keys[i]
    instance_masks = instance_masks[i]
    instance_boundaries = instance_boundaries[i]
    keypoints = keypoints_map[box].extend(keypoint[i])
    if box in keypoints_map:
      keypoints_map[box].extend(keypoint[i])
    if box_to_color_map[box] == groundtruth_box_visualization_color:
      display_str_label = ''
    else:
      if not agnostic_mode:
        display_str_label = class_name
      else:
        display_str_label = str(class_name)
    if display_str_label:
      display_str = ('%s: %.2f' % (display_str_label, int(100*scores[i])))
      display_str += ('%s' % format(display_str, int(100*scores[i])))
    boxes_on_image_array = draw_bounding_box_on_image_array(
      image,
      box['bbox'],
      instance_boundaries_map[box],
      keypoints_map[box],
      color=box_to_color_map[box],
      thickness=box_line_thickness,
      use_normalized_coordinates=use_normalized_coordinates)
    draw_keypoints_on_image_array(
      image,
      box_to_keypoints_map[box],
      radius=line_thickness * 2,
      use_normalized_coordinates=use_normalized_coordinates)
    # Capture image copy
    capture, image, confidence
    capture, image, confidence

```

Figure 5.6: Confidence calculations and capture Image

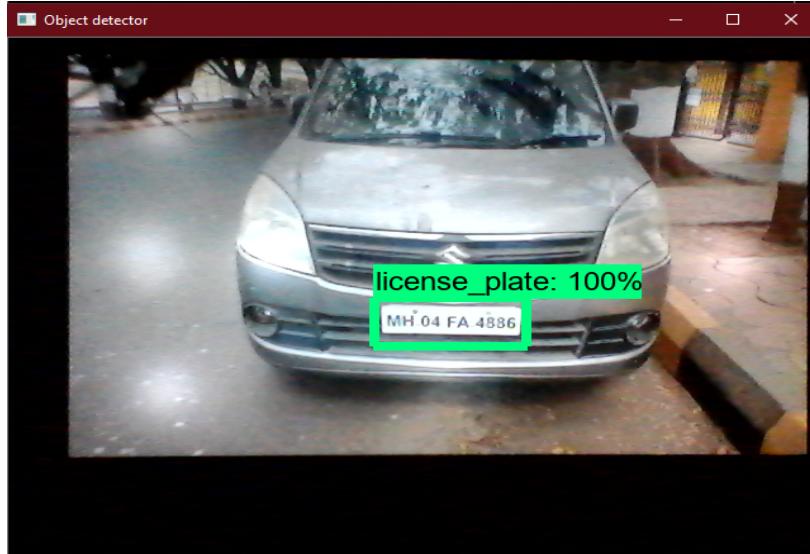


Figure 5.7: Output of Object Detection

License plate detection is done using OpenCV (Open source computer vision) to analyse and process the number plate in the database to know whether the number plate is an registered number in the database or an outsider. After capturing the license plate by OCR (Optical character recognition) number is obtained. OCR is an electronic method of conversion of images of typed, handwritten or printed text. Thus giving us the number which is to be processed.

5.3.2 Image segmentation and Optical Character Recognition (OCR)

Now that the license plate was detected from the live feed and the frame of the video has been captured, the image is processed for increasing the accuracy of OCR. First the image is read from the directory and saved into a variable. Then the image is resized and

converted to grey scale image to convert it into a black and white image. This is then processed through bilateral filtering to reduce noise in the image.

Edge detection is performed to detect the edge of the license plate. There are edge detection methods such as Sobel's Operator method, Canny Edge, Prewitt's method etc., but for our purpose we have used canny edge detection as it gives the best result for our needs. The edge detection process displays only the edges of the different objects in the frame. Contouring is done for drawing out similar areas and forming areas of common features. Among all the contours we need to extract the contour with the license plate. To do that we find contours with maximum white pixels as the license plate is white colour for private vehicle. We can change it to black or yellow if needed. All the contours that satisfy the requirements are then counted. Then arranged in descending order based on the area. The contour with largest area is believed to be the one with license plate.

Then all the other contours are deleted and this contour is retained. Then masking is performed to isolate the contour and the cropped. The cropped frame is then processed for OCR. Optical character recognition in Python is done with the help of Google's Tesseract Library. Tesseract is an open source platform for OCR. It's pretrained library can recognise letters and numbers from an image. Here we use it on the cropped frame. This returns a string which has the characters. For us the string gives us the number plate details. The text string is passed to the main code. Refer Fig (OCR code) for the code of this process.

```

1 import cv2
2 import numpy
3 import pytesseract
4
5 whitelets = string.digits + "abcdefghijklmnopqrstuvwxyz ABCDEFGHIJKLMNOPQRSTUVWXYZ"
6
7 def process_frame(frame):
8     # This function extracts Characters from a static image
9     # defined in the parameters
10
11     # Parameters
12     # frame - Input image
13     # frame - Input image array passed from the main code
14     # choice - To display the whole process, If false display only original image and output image with
15     # license plate found
16     # Returns
17     # correct_string - A string which contains the characters recognised from the OCR, in this case the license plate number
18
19     image = frame.copy()
20
21     image = cv.cvtColor(image, cv.COLOR_BGR2GRAY)
22     image = cv.resize(image, (600,400), interpolation = cv.INTER_AREA )
23     gray = cv.bilateralFilter(image, 10, 100, 100) # to remove noise
24
25     # Retaining only the contour with number plate
26     contours = cv.findContours(gray, cv.RETR_TREE, cv.CHAIN_APPROX_SIMPLE)
27     contours = sorted(contours, key = cv.contourArea, reverse = True)[0]
28
29     # Loop over our contours
30     for c in contours:
31         peri = cv.arcLength(c, True)
32         approx = cv.approxPolyDP(c, 0.01 * peri, True)
33
34         if len(approx) == 4:
35             screenCnt = approx
36             break
37
38     if len(screenCnt) == 0:
39         detected = 0
40         print("No license plate found")
41     else:
42         detected = 1
43
44     if detected == 1:
45
46         cv.drawContours(image, [count], -1, (0, 255, 0), 3)
47
48         gray = cv.cvtColor(image, cv.COLOR_BGR2GRAY)
49         mask = np.zeros(gray.shape, dtype="uint8")
50         new_image = np.zeros_like(gray)
51
52         height, width = gray.shape
53         mask[height/2-10:height/2+10, width/2-10:width/2+10] = 255
54
55         Cropped = gray[mask != 0].copy()
56
57         # Scale image
58         scale_percent = 200 # percent of original size
59         width = int(Cropped.shape[1] * scale_percent / 100)
60         height = int(Cropped.shape[0] * scale_percent / 100)
61         dim = (width, height)
62
63         Cropped = cv.resize(Cropped, dim, interpolation = cv.INTER_NEAREST)
64
65         # Invert image
66         Cropped = cv.bitwise_not(Cropped)
67
68         # Perform OCR
69         config = '--psm 11'
70         text = pytesseract.image_to_string(Cropped, config=config)
71
72         unlist_text = list(text)
73
74         for i in range(len(unlist_text)):
75             if unlist_text[i] not in whitelets:
76                 unlist_text[i] = ''
77
78         text_string = ''.join(unlist_text)
79
80         print(text_string)
81
82         font = cv.FONT_HERSHEY_SIMPLEX
83         org = (420, 350)
84         fontScale = 1
85         thickness = 2
86
87         cv.putText(frame, text_string, (420,350), cv.FONT_HERSHEY_SIMPLEX, 1, cv.LINE_AA)
88
89         cv.imshow("Output", out)
90
91         cv.waitKey(0)
92         cv.destroyAllWindows()
93
94     return text_string
95
96

```

Figure 5.8: Image Segmentation

Image Processing and Optical character recognition:

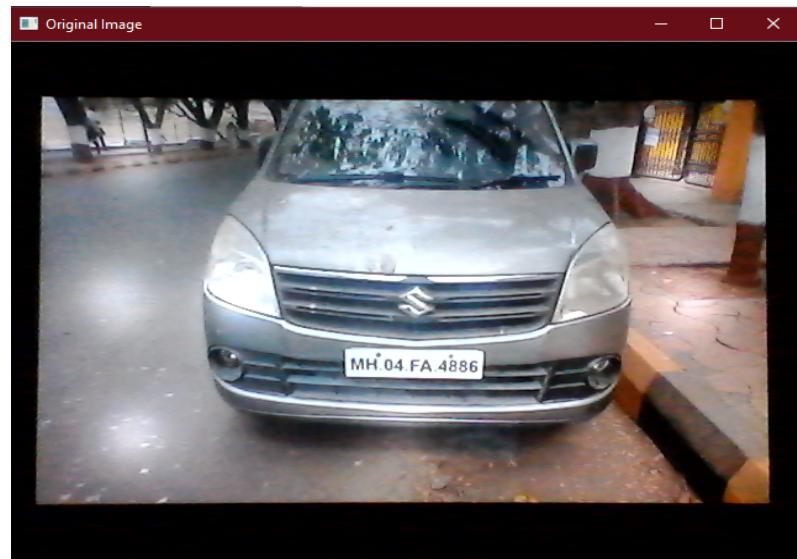


Figure 5.9: Original Image

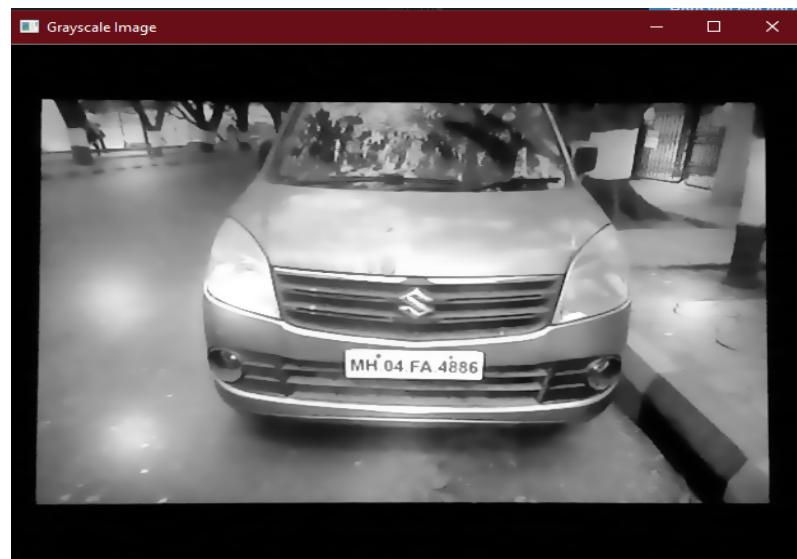


Figure 5.10: Grayscale Image

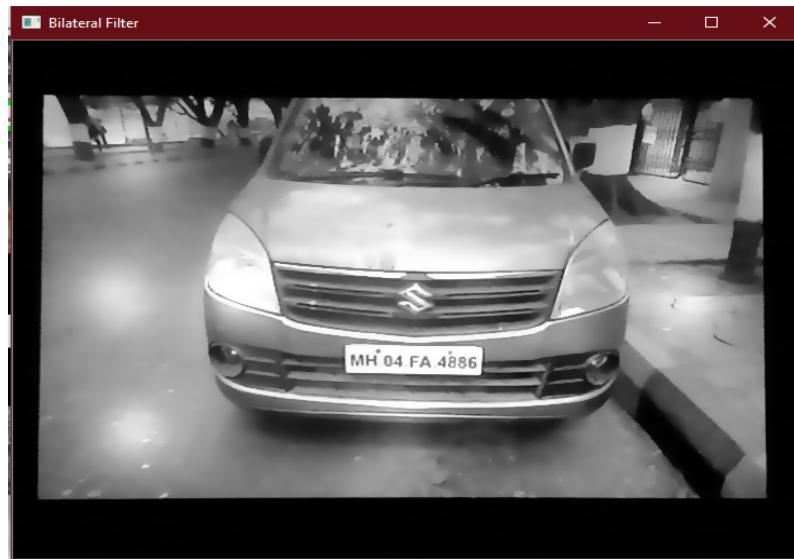


Figure 5.11: Filtered Image



Figure 5.12: Canny Edge Detection

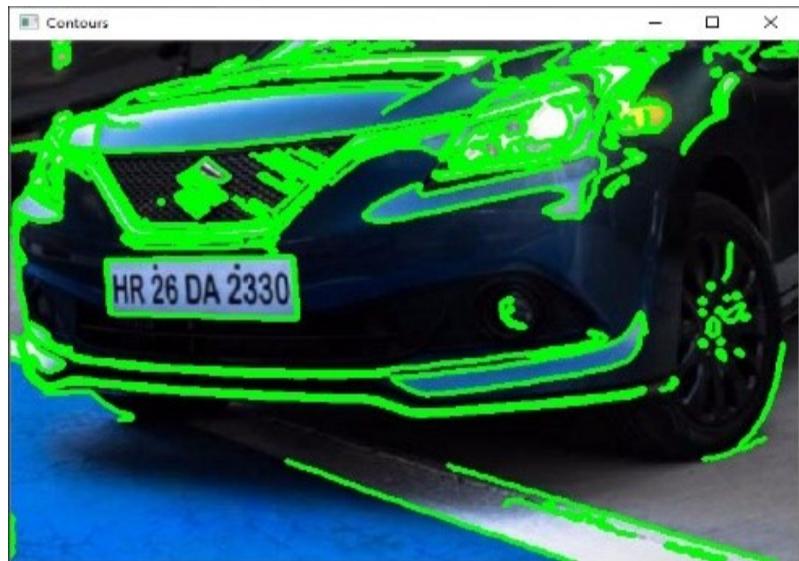


Figure 5.13: Detected Contours

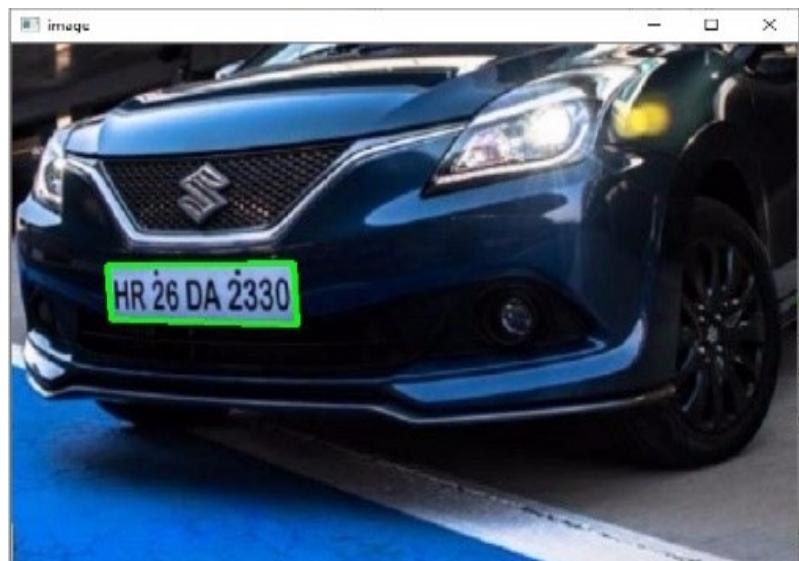


Figure 5.14: Contour Containing License Plate

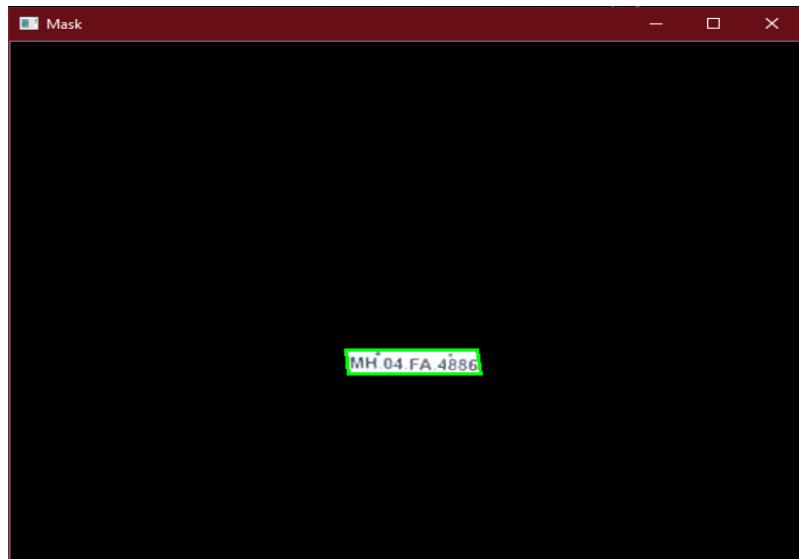


Figure 5.15: 7xx Masked Image

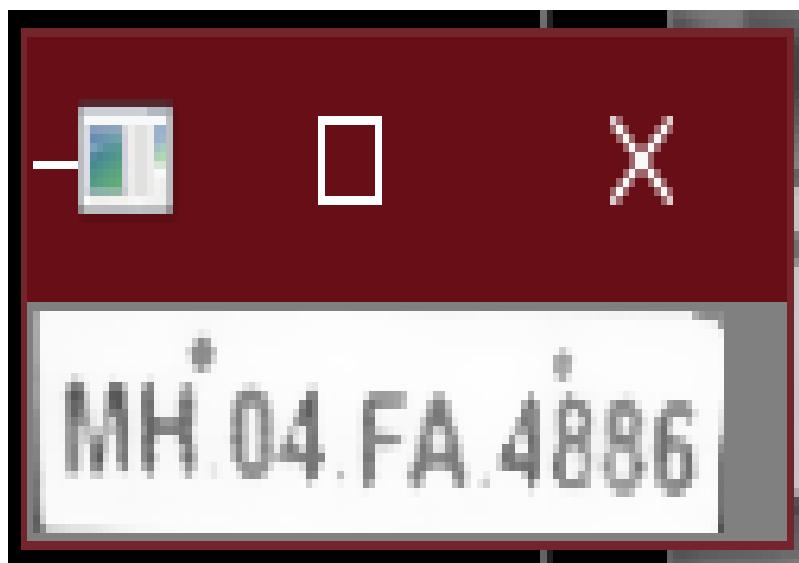


Figure 5.16: Extracted License plate

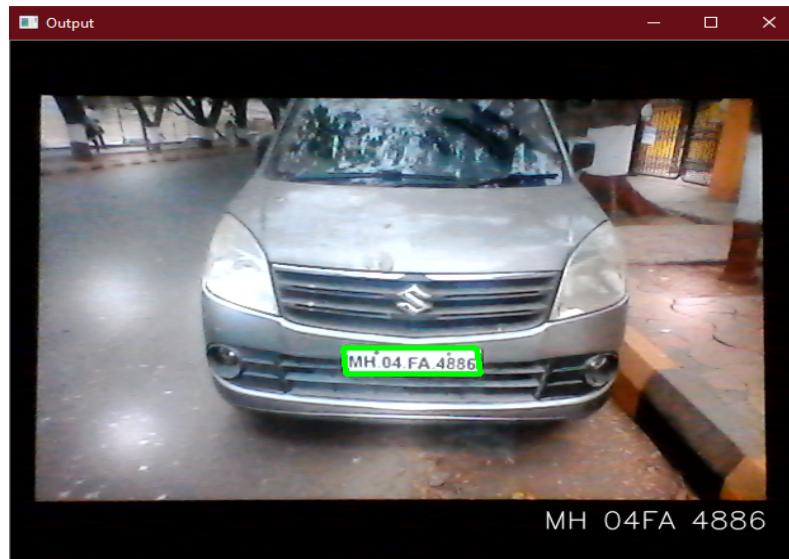


Figure 5.17: License Plate Recognised and Displayed

5.3.3 Firebase Realtime Database and Storage

Google firebase is an online real-time database management system. We are using this feature to record the save the license plate number and the snapshot of the car from the front when it was detected. When the license plate is detected and processed through OCR, the text string that is returned is pushed to firebase. Simultaneously, the image captured is saved in the firebase storage for future references.

The parking lot availability is also fed to the firebase RTDB which is used for WebApp to display the status of the area.

```

1   Created on Fri Apr 17 18:08:19
2
3   @author: Navin Subbu
4
5   """
6
7   from firebase import firebase
8   import pyrebase
9
10
11   # Firebase Initialization with app config
12   # This function initializes the firebase application
13   # config = {
14   #     "projectId": "Ansyshydrus-0edecdf",
15   #     "databaseURL": "https://python-test-1235f.firebaseio.com/",
16   #     "storageBucket": "python-test-1235f.appspot.com",
17   #     "messagingSenderId": "1:70811262681:web:93df50c96e28b114bafef",
18   #     "appId": "G-9YR00NvLbY"
19   # }
20
21
22
23
24   def firebase_store(path,image_name) :
25
26       """This Function stores a given file into Google Firebase Storage in
27       a specific location categorised in the order of year and month
28       Parameters
29
30       path : String
31           It is the Path of the saved image
32       image_name : String
33           It is the name of the image
34
35       Returns
36       None
37
38   """
39
40   firebase = pyrebase.initialize_app(config)
41   storage = firebase.storage()
42   path_on_cloud = "{Car Recorded}/{0}/{1}".format(path,image_name)
43   path_local = "{0}/{1}/{2}/".format(path,image_name)
44   storage.child(path_on_cloud).put(path_local)
45
46
47
48   def firebase_realtime_db(text) :
49
50
51       """This function pushes a given string into Realtime Database in Google Firebase
52       Parameters
53
54       text : String
55           Its a string, the license plate number
56
57       Returns
58
59   """
60
61
62   result = firebase.post('/Detected', text)
63   print(result)
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
79
80
81
82
83
84
85
86
87
88
89
89
90
91
92
93
94
95
96
97
98
99
99
100
101
102
103
104
105
106
107
108
109
109
110
111
112
113
114
115
116
117
118
119
119
120
121
122
123
124
125
126
127
128
129
129
130
131
132
133
134
135
136
137
138
139
139
140
141
142
143
144
145
146
147
148
149
149
150
151
152
153
154
155
156
157
158
159
159
160
161
162
163
164
165
166
167
168
169
169
170
171
172
173
174
175
176
177
178
179
179
180
181
182
183
184
185
186
187
188
189
189
190
191
192
193
194
195
196
197
198
199
199
200
201
202
203
204
205
206
207
208
209
209
210
211
212
213
214
215
216
217
218
219
219
220
221
222
223
224
225
226
227
228
229
229
230
231
232
233
234
235
236
237
238
239
239
240
241
242
243
244
245
246
247
248
249
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
288
289
289
290
291
292
293
294
295
296
297
298
299
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
318
319
319
320
321
322
323
324
325
326
327
328
329
329
330
331
332
333
334
335
336
337
338
339
339
340
341
342
343
344
345
346
347
348
349
349
350
351
352
353
354
355
356
357
358
359
359
360
361
362
363
364
365
366
367
368
369
369
370
371
372
373
374
375
376
377
378
379
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
418
419
419
420
421
422
423
424
425
426
427
428
429
429
430
431
432
433
434
435
436
437
438
439
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
487
488
489
489
490
491
492
493
494
495
496
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
787
788
789
789
790
791
792
793
794
795
796
797
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
887
888
889
889
890
891
892
893
894
895
896
897
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
979
979
980
981
982
983
984
985
986
987
987
988
989
989
990
991
992
993
994
995
996
997
997
998
999
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1087
1088
1089
1089
1090
1091
1092
1093
1094
1095
1096
1096
1097
1098
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1179
1180
1181
1182
1183
1184
1185
1186
1187
1187
1188
1189
1189
1190
1191
1192
1193
1194
1195
1195
1196
1197
1197
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1279
1280
1281
1282
1283
1284
1285
1286
1287
1287
1288
1289
1289
1290
1291
1292
1293
1294
1295
1295
1296
1297
1297
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1379
1380
1381
1382
1383
1384
1385
1386
1387
1387
1388
1389
1389
1390
1391
1392
1393
1394
1395
1395
1396
1397
1397
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1479
1480
1481
1482
1483
1484
1485
1486
1487
1487
1488
1489
1489
1490
1491
1492
1493
1494
1495
1495
1496
1497
1497
1498
1499
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1579
1580
1581
1582
1583
1584
1585
1586
1587
1587
1588
1589
1589
1590
1591
1592
1593
1594
1595
1595
1596
1597
1597
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1679
1680
1681
1682
1683
1684
1685
1686
1687
1687
1688
1689
1689
1690
1691
1692
1693
1694
1695
1695
1696
1697
1697
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1779
1780
1781
1782
1783
1784
1785
1786
1787
1787
1788
1789
1789
1790
1791
1792
1793
1794
1795
1795
1796
1797
1797
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1879
1880
1881
1882
1883
1884
1885
1886
1887
1887
1888
1889
1889
1890
1891
1892
1893
1894
1895
1895
1896
1897
1897
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1979
1980
1981
1982
1983
1984
1985
1986
1987
1987
1988
1989
1989
1990
1991
1992
1993
1994
1994
1995
1996
1996
1997
1997
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2089
2090
2091
2092
2093
2094
2095
2095
2096
2097
2097
2098
2099
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2189
2190
2191
2192
2193
2194
2195
2195
2196
2197
2197
2198
2199
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2289
2290
2291
2292
2293
2294
2295
2295
2296
2297
2297
2298
2299
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
236
```

Figure 5.18: Firebase Real-Time Database

5.3.4 Parking Lot Occupancy management using IoT

The second part of this system is to keep track of the status of the parking lots in the area such as keeping a track of empty lot and no of occupied lots. For achieving this task we have implemented Internet of Things in our project using the simple modules such as NodeMCU development board, a microcontroller board with ESP8366 WiFi chip embedded in it and the Ultrasonic Distancing Meter (UDM) for measuring distance. UDM is a sonar based distance measurement device. Both of the components are connected with wires and here NodeMCU is the driving element of the system and UDM is the driven element. NodeMCU is connected to a Local Wifi Network, by achieving this, the system is connected to the internet and can be controlled from any part of the world.

The UDM is fitted in front of the parking space perpendicular to the length if the parking space. When it is unoccupied the distance measured by UDM is very high as there is no surface for the sound waves to rebound from. When a car is parked in the UDM senses decrease in distance as the bonnet of the car acts as a surface to reflect the sound waves back. The NodeMCU is programmed such that when the distance measured falls below a certain threshold, it declares the parking lot as occupied. The updated status of the parking lot is pushed to the Firebase RTDB. One NodeMCU can keep track of 6 parking lot, so multiple such systems can be installed for a large number of parking lots. When the parking space is occupied the system uploads ‘1’ to firebase, this indicates that the space is occupied and if the car moves out it is reset to ‘0’ indicating it to be vacant and allow other cars to park in that area.





```
BE_final
#include <ESP8266WiFi.h>
#include <FirebaseArduino.h>

// Initialise Firebase database ID, Token Secrets
// SSID, password of the local WIFI Network
#define FIREBASE_HOST "nodemcu-led-b3637.firebaseio.com"
#define FIREBASE_AUTH "bkIScvsVBNfLitAt4pZO3EqEtM6Sfc2cLn7MJAcB"
#define WIFI_SSID "Anonymous"
#define WIFI_PASSWORD "batman123"

//Firebase RTDB Path
const String path = "/Parking";
const String Floor = "/Floor 1"; //initialise the number according to the floor
```

Figure 5.19: Firebase and WiFi credentials

```

// defining pins for UDM connection
const int trigPin1 = D0;
const int echoPin1 = D1;
const int trigPin2 = D2;
const int echoPin2 = D3;
const int trigPin3 = D4;
const int echoPin3 = D5;

// defines variables to store the values received from UDMs
long duration;
int distance;
long Sensor1,Sensor2,Sensor3;

// Function for calculation of Distance based on UDM readings
void SonarSensor(int trigPin,int echoPin)
{
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);
    duration = pulseIn(echoPin, HIGH);
    distance = (duration/2) / 29.1;
}

```

Figure 5.20: UDM pins and Distance Calculating Function

```

void setup() {
    // This Code runs only once and hence used to connect to wifi
    // and initialises pins
    // Start Serial Monitor for Reading Outputs
    Serial.begin(115200);

    // connect to wifi.
    WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
    Serial.print("Connecting to WiFi");
    while (WiFi.status() != WL_CONNECTED) {
        Serial.print(".");
        delay(500);
    }
    Serial.println();
    Serial.print("connected: ");
    Serial.println(WiFi.localIP());

    // Firebase Initialised
    Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);

    // Initialising the pins as Input or Output
    pinMode(trigPin1, OUTPUT); // Sets the trigPin as an Output
    pinMode(echoPin1, INPUT); // Sets the echoPin as an Input
    pinMode(trigPin2, OUTPUT); // Sets the trigPin as an Output
    pinMode(echoPin2, INPUT); // Sets the echoPin as an Input
    pinMode(trigPin3, OUTPUT); // Sets the trigPin as an Output
    pinMode(echoPin3, INPUT); // Sets the echoPin as an Input
}

```

Figure 5.21: One time run Loop for Setting up the system and connecting to Wifi

```

void loop()
{
    // this code runs repeatedly till the system is shut off

    SonarSensor(trigPin1, echoPin1); //Function Call for Parking 1
    Sensor1 = distance;
    if(Sensor1<10) // Loop to check distance
    {
        Firebase.setInt(Floor + "/Parking 1", 1); // Parking space Occupied
    }
    else
    {
        Firebase.setInt(Floor + "/Parking 1", 0); // Parking space Vacant
    }

    SonarSensor(trigPin2, echoPin2); //Function Call for Parking 2
    Sensor2 = distance;
    if(Sensor2<10) // Loop to check distance
    {
        Firebase.setInt(Floor + "/Parking 2", 1); // Parking space Occupied
    }
    else
    {
        Firebase.setInt(Floor + "/Parking 2", 0); // Parking space Vacant
    }

    SonarSensor(trigPin3, echoPin3); //Function Call for Parking 3
    Sensor3 = distance;
    if(Sensor3<10) // Loop to check distance
    {
        Firebase.setInt(Floor + "/Parking 3", 1); // Parking space Occupied
    }
    else
    {
        Firebase.setInt(Floor + "/Parking 3", 0); // Parking space Vacant
    }

    // Print the reading in Serial Monitor
    Serial.print(Sensor1);
    Serial.print(" \t ");
    Serial.print(Sensor2);
    Serial.print(" \t ");
    Serial.print(Sensor3);
    Serial.println();
    delay(10);
}

```

Figure 5.22: Main code

Chapter 6

Results

6.1 Object Detection Output



Figure 6.1: Detected License plate and Confidence score

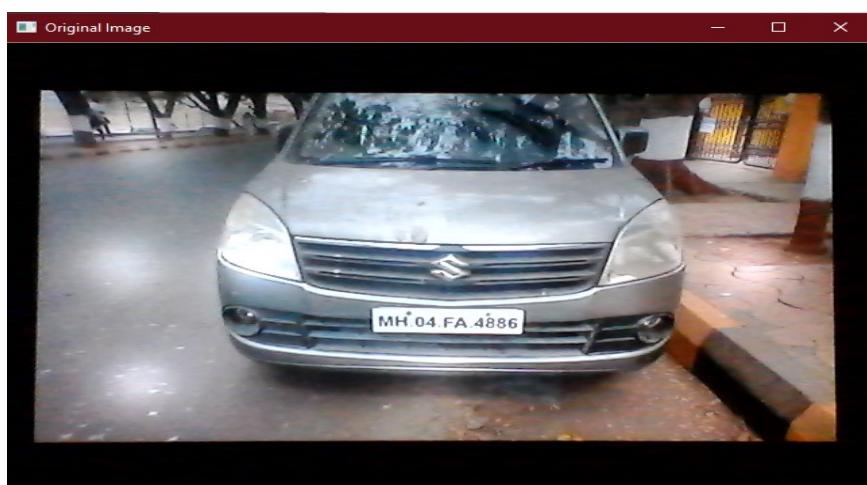


Figure 6.2: Original image Captured



Figure 6.3: License Plate Recognised and Displayed

6.2 Database Output – Machine Learning

A screenshot of a web browser displaying the Firebase Storage interface for a project named "Python Test". The left sidebar shows various Firebase services like Authentication, Database, Storage, Hosting, Functions, ML Kit, Analytics, Predictions, A/B Testing, Cloud Messaging, In-App Messaging, Remote Config, Dynamic Links, AdMob, and Extensions. The main area is titled "Storage" and shows a list of files under "Car Recorded > April". The list includes 14 image files, each with a thumbnail, name, size, type, and last modified date. The files are named CAP_19Apr_03-57AM.png through CAP_20Apr_03-36AM.png, all of which are image/png files ranging from 166.68 KB to 394.65 KB in size, and were last modified on April 20, 2020.

Figure 6.4: Firebase image store

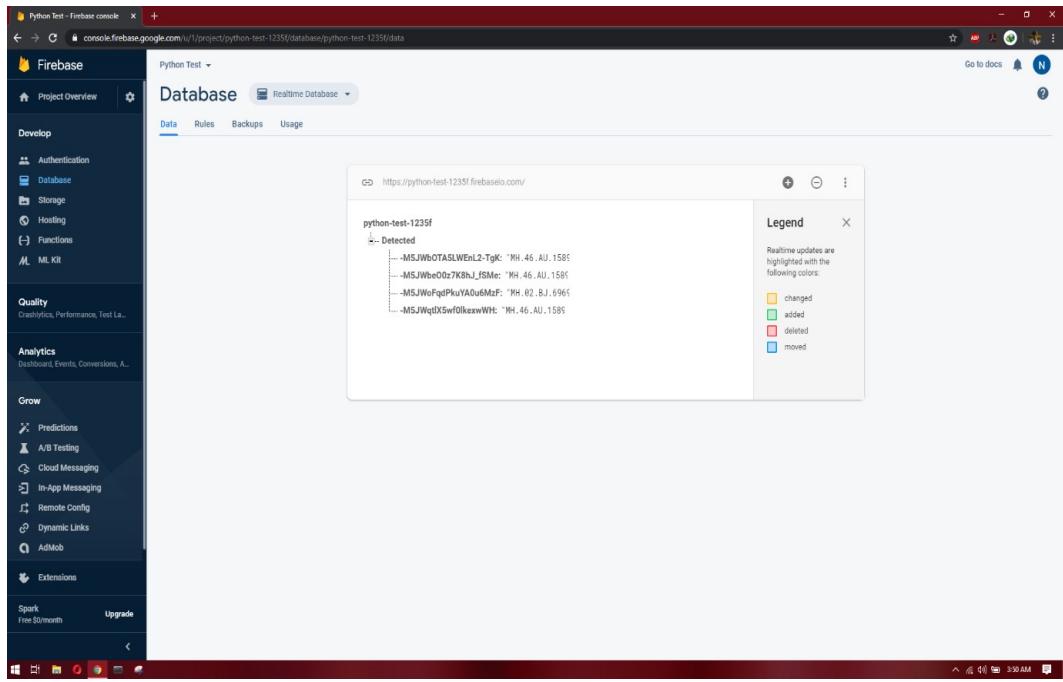


Figure 6.5: Realtime database

6.3 Database Output – IoT

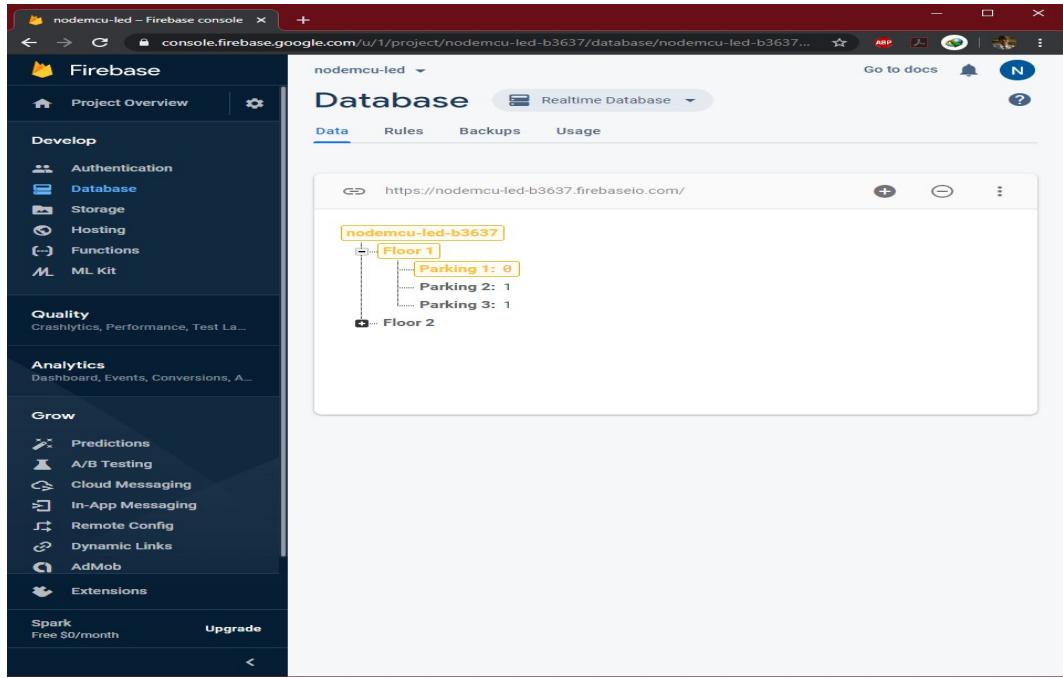


Figure 6.6: When Parking Lot 1 is unoccupied

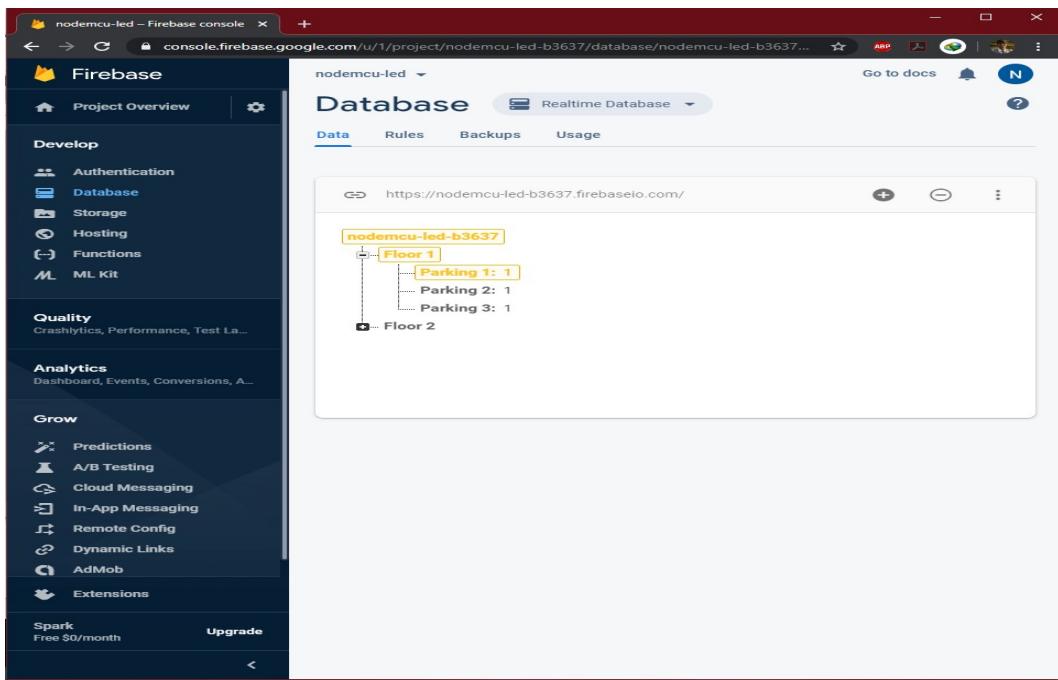


Figure 6.7: When Parking Lot 1 is occupied

6.4 Webpage Output

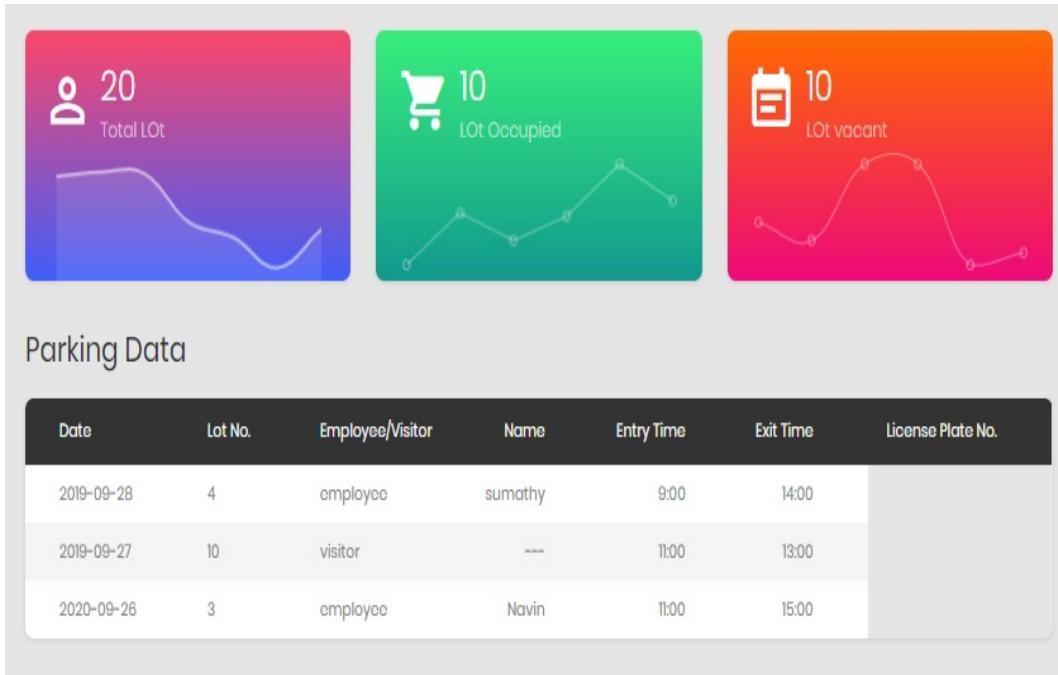


Figure 6.8: web page front end

Chapter 7

Conclusion and Future Scope

7.1 Conclusion

In this work, a first approach of using existing surveillance cameras for detecting license plates is presented. An open source platform is used and improved to evaluate only objects that are moving. The algorithm run on a distributed platform that operate multiple cameras and sensors, to support comprehensive security management. The first results are reasonable in performance, reaching a near real-time LPR detection, even for high resolution images. The final output of the license plates was found to be very close to the actual value on the plates. However, some errors creep in depending upon the quality of the original image, the lighting, the shakiness and general condition of the license plate.

7.2 Future Scope

In future the system can be extended which is not only specific to a private parking like malls, company parking, etc. but can also be implemented on public parking which has extending features such as reduction of traffic as search time for parking is reduced ,enhanced user experience as user can easily find the vacant parking lot, real-time data and trend insight which is achieved by studying the arrival and leaving time of a particular car using ML, decreased management costs as this reduces the labour cost and can be more efficient. Since data stored is real-time safety can be ensured which helps prevent parking violations and suspicious activity.

Chapter 8

References

R. Lookmuang, K. Nambut and S. Usanavasin, "Smart parking using IoT technology," 2018 5th International Conference on Business and Industrial Research (ICBIR) , Bangkok, 2018, pp. 1-6. URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=arnumber=8391155isnumber=839>

G. Amato, F. Carrara, F . Falchi, C. Gennaro and C. Vairo, "Car parking occupancy detection using smart camera networks a nd Deep Learning," 2016 IEEE Symposium on Computers and Communication (ISCC) , Messina, 2016, pp. 1212-1217. URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=arnumber=7543901isnumber=7543699>

L. Dominguez, J. P. D 'Amato, A. Perez, A. Rubiales and R. Barbuza, "Running license plate recognition (LPR) algorithms on smart surveillance cameras. A f easibility analysis," 2018 13th Iberian Conference on Information Systems and Technologies (CISTI) , Caceres, 2018, pp. 1-5. URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=arnumber=8399194isnum>

T. N. Pham, M. Tsai, D . B. Nguyen, C. Dow and D. D eng, "A Cloud-Based Smart-Parking System Based on Internet-of-Things Technologies," in IEEE Access , vol. 3, pp. 1581-1591, 2015. URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=arnumber=7247632isnumber=7247632>

OpenCV, open source i mage processing library for python URL:<https://opencv.org/about/>

[6]Optical character r ecognition or optical character reader (OCR) is the electronic or mechanical conversion of images of t yped, handwritten or printed text into machine-encoded text, whether from a scanned document, a photo of a document, a s cene-photo URL:https://en.wikipedia.org/wiki/Optical_character_recognition

Tesseract is an optical character recognition engine for various operating systems. It is free software, released under the Apache License, Version 2.0 and development has been sponsored by Google since 2006. URL:[https://en.wikipedia.org/wiki/Tesseract_\(software\)](https://en.wikipedia.org/wiki/Tesseract_(software))

The library for OCR t ool in Tesseract posted in github URL: <https://github.com/tesseract-ocr/tesseract>

[9] The Raspberry Pi is a low cost, credit-card sized computer that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. It is a capable little device that enables people of all ages to explore computing, and to learn how to program in languages like Scratch and Python. URL:<https://www.raspberrypi.org/about/>

[10]Mike Liao, March 1, 2018 Detecting Objects in (almost) Real-time: FasterRCNN Explained with Code. <https://towardsdatascience.com/fasterrcnn-explained-part-1-with-code-599c16568cff>

[11]Subrata Goswami , July 11, 2018. A deeper look at how Faster-RCNN works <https://medium.com/@wdeeper-look-at-how-faster-rcnn-works-84081284e1cd>