" 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 (Internal)	Head of Department (Electronics and Telecommunication)	Principal (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 Ganesha 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

List of Figures

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

Chapter7 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

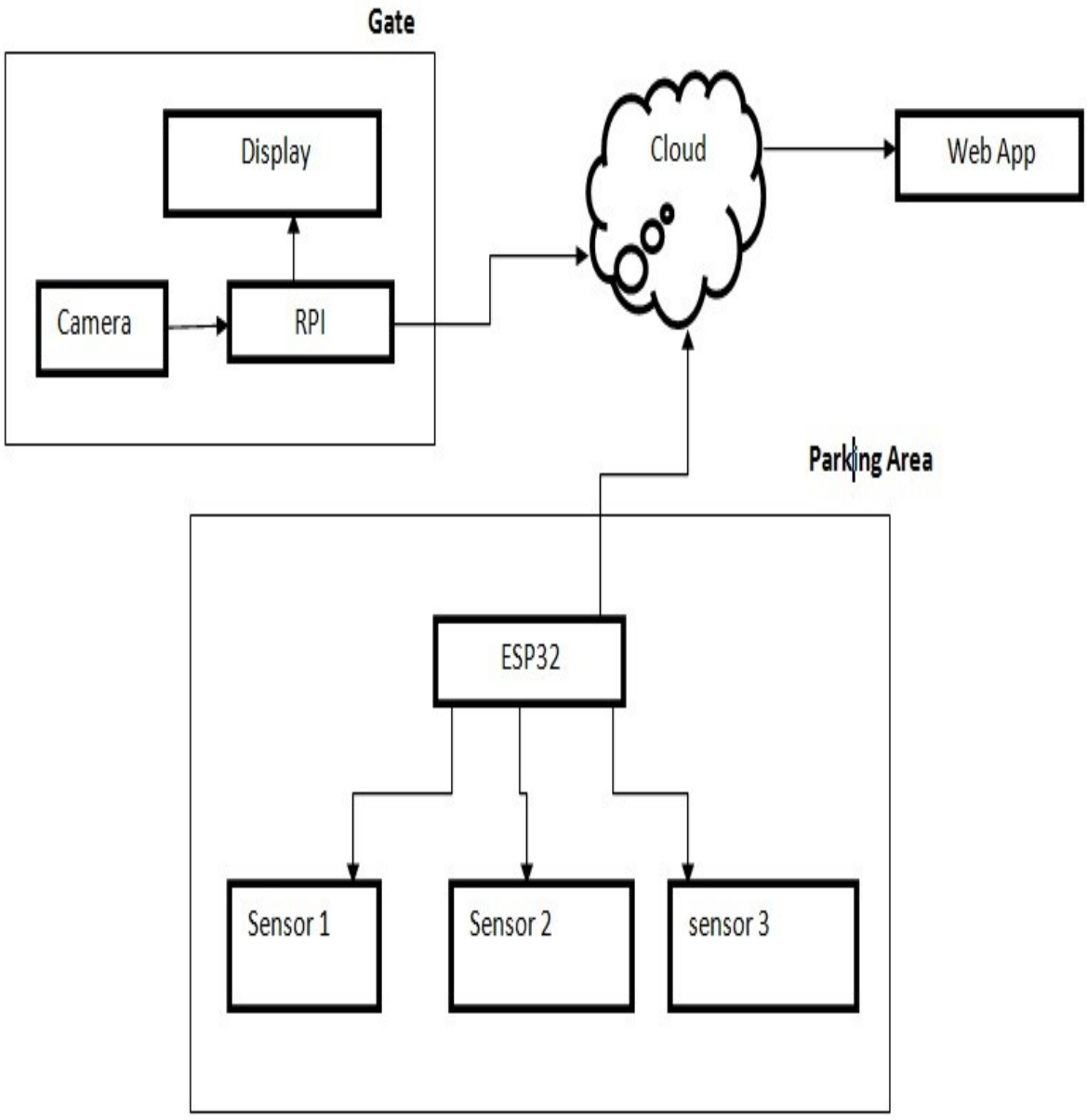


Figure 3.1: Flow Diagram

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

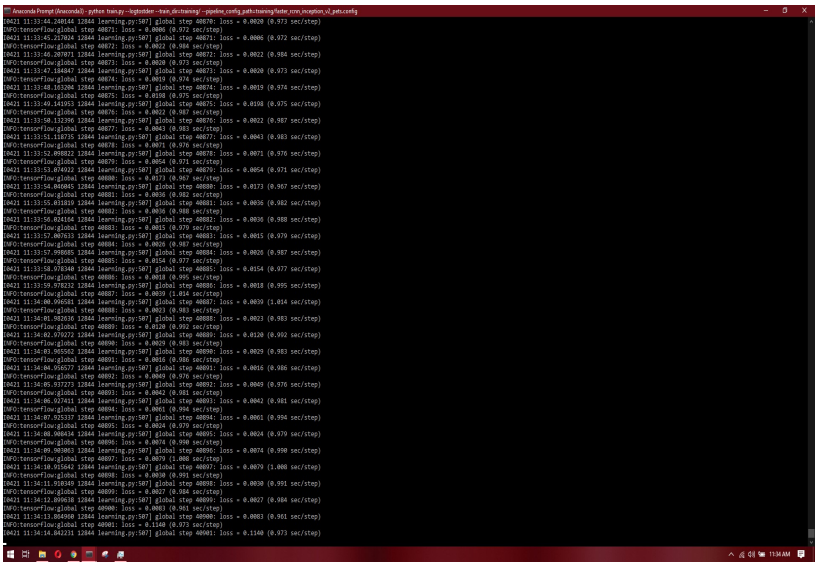
Our main purpose is to build a parking system to save time and to increase security. Our parking system has an organised way to allot resident/employee a permanent parking space. People waste time in finding parking spots to park their car so we can created a system to create an efficient system to overcome 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 increase the need for more hardware storage. To overcome this, we have used Machine Learining. 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 generate good features from the image. RPN is 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 itll 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



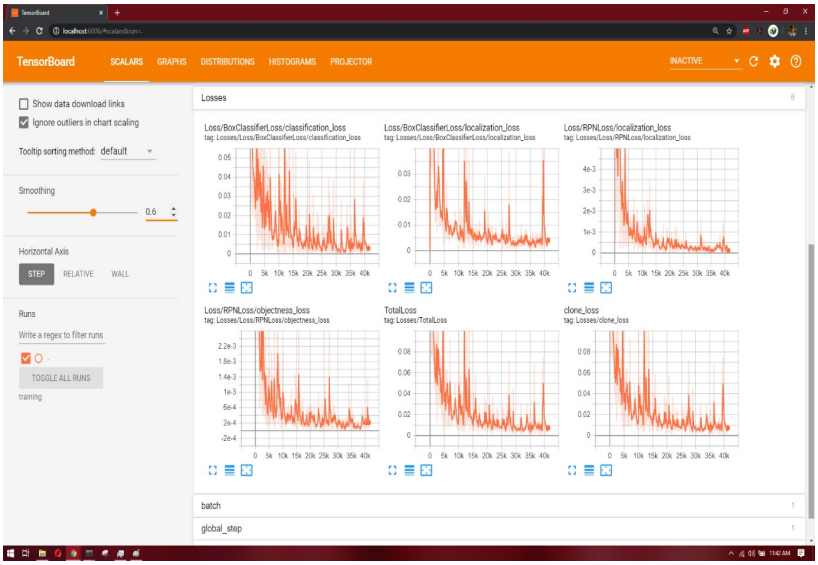
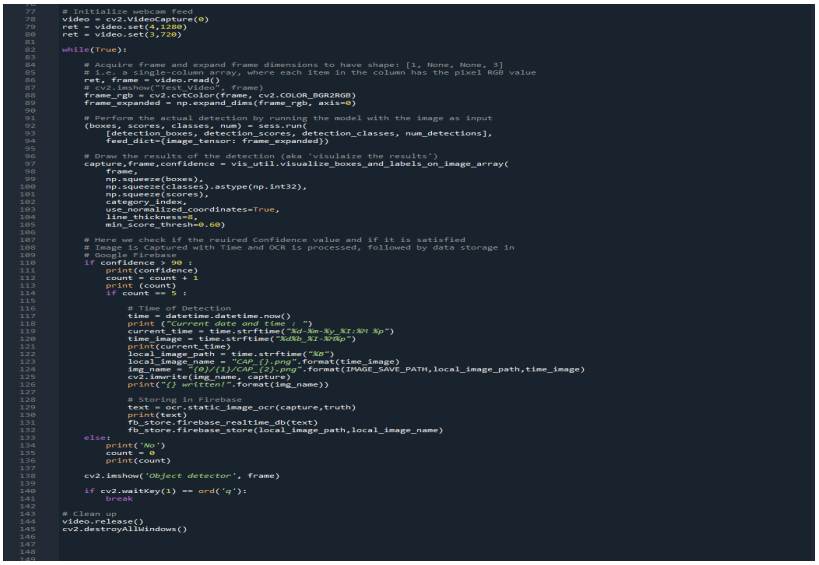


Figure 5.1: Losses

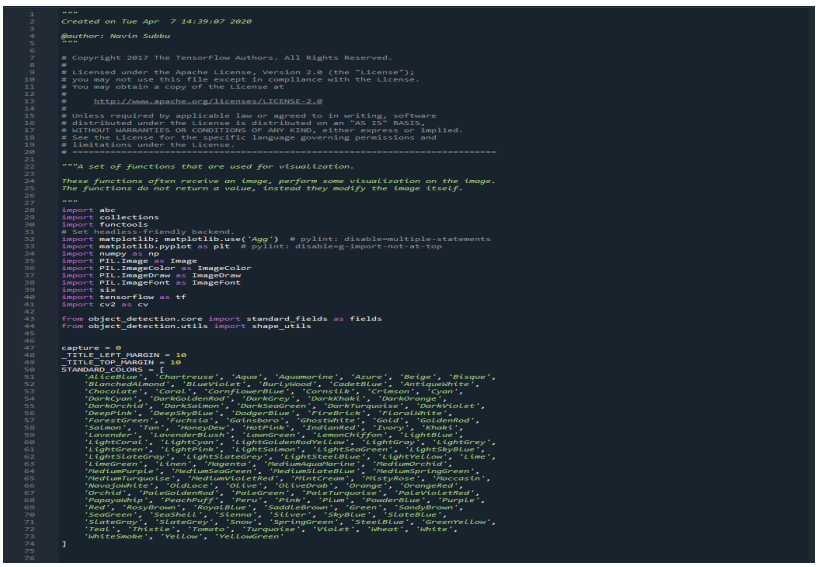
```python
# Import packages
import os
import cv2
import numpy as np
import tensorflow as tf
import sys
import datetime

# This is needed since the notebook is stored in the object_detection folder.
sys.path.append("..")

# Import utilities and modules
from utils import label_map_util
from utils import custom_visualization_utils as vis_util
import ocr_extraction as ocr
import firebase_storage as fb_store

# Name of the directory containing the object detection module we're using
MODEL_NAME = 'inference_graph'

# Grab path to current working directory
CWD_PATH = os.getcwd()

# Path to frozen detection graph .pb file, which contains the model that is used
# for object detection.
PATH_TO_CKPT = os.path.join(CWD_PATH,MODEL_NAME,'frozen_inference_graph.pb')

# Path to label map file
PATH_TO_LABELS = os.path.join(CWD_PATH,'training','labelmap.pbtxt')

#Saving images when Object is detected
IMAGE_SAVE_PATH = os.path.join(CWD_PATH,'detected')
truth = False

# Number of classes the object detector can identify
NUM_CLASSES = 1

## Load the label map.
# Label maps map indices to category names, so that when our convolution
# network predicts `5`, we know that this corresponds to `king`.
# Here we use internal utility functions, but anything that returns a
# dictionary mapping integers to appropriate string labels would be fine
label_map = label_map_util.load_labelmap(PATH_TO_LABELS)
categories = label_map_util.convert_label_map_to_categories(label_map,
                                                            max_num_classes=NUM_CLASSES,
                                                            use_display_name=True)

category_index = label_map_util.create_category_index(categories)

# Load the Tensorflow model into memory.
detection_graph = tf.Graph()
with detection_graph.as_default():
    od_graph_def = tf.GraphDef()
    with tf.gfile.GFile(PATH_TO_CKPT, 'rb') as fid:
        serialized_graph = fid.read()
        od_graph_def.ParseFromString(serialized_graph)
        tf.import_graph_def(od_graph_def, name='')

    sess = tf.Session(graph=detection_graph)

# Define input and output tensors (i.e. data) for the object detection classifier
# Input tensor is the image
image_tensor = detection_graph.get_tensor_by_name('image_tensor:0')

# Output tensors are the detection boxes, scores, and classes
# Each box represents a part of the image where a particular object was detected
detection_boxes = detection_graph.get_tensor_by_name('detection_boxes:0')

# Each score represents level of confidence for each of the objects.
# The score is shown on the result image, together with the class label.
detection_scores = detection_graph.get_tensor_by_name('detection_scores:0')
detection_classes = detection_graph.get_tensor_by_name('detection_classes:0')

# Number of objects detected
num_detections = detection_graph.get_tensor_by_name('num_detections:0')

# Initialize webcam feed
video = cv2.VideoCapture(0)
ret = video.set(4,1280)
ret = video.set(3,720)
```

Figure 5.2: Tensorflow Initialization



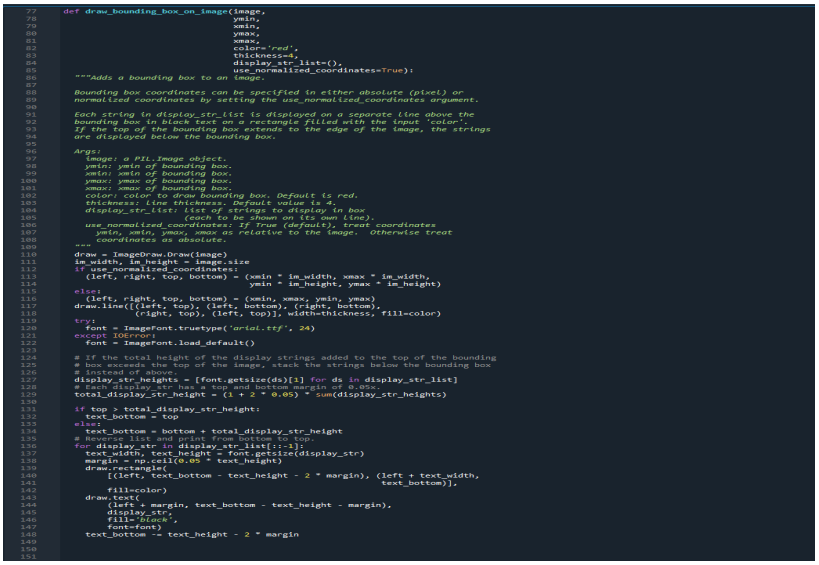
```
# Initialize webcam feed
video = cv2.VideoCapture(0)
ret = video.set(4,1280)
ret = video.set(3,720)

while(True):

    # Acquire frame and expand frame dimensions to have shape: [1, None, None, 3]
    # i.e. a single-column array, where each item in the column has the pixel RGB value
    ret, frame = video.read()
    # cv2.imshow('test video', frame)
    frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    frame_expanded = np.expand_dims(frame_rgb, axis=0)

    # Perform the actual detection by running the model with the image as input
    (boxes, scores, classes, num) = sess.run(
        [detection_boxes, detection_scores, detection_classes, num_detections],
        feed_dict={image_tensor: frame_expanded})

    # Draw the results of the detection (aka 'visualize the results')
    capture,frame,confidence = vis_util.visualize_boxes_and_labels_on_image_array(
        frame,
        np.squeeze(boxes),
        np.squeeze(classes).astype(np.int32),
        np.squeeze(scores),
        category_index,
        use_normalized_coordinates=True,
        line_thickness=8,
        min_score_thresh=0.60)

    # Here we check if the reuired Confidence value and if it is satisfied
    # Image is Captured with Time and OCR is processed, followed by data storage in
    # Google Firebase
    if confidence > 90 :
        print(confidence)
        count = count + 1
        print (count)
        if count == 5 :

            # Time of Detection
            time = datetime.datetime.now()
            print ("current date and time : ")
            current_time = time.strftime("%d-%m-%y_%I:%M %p")
            time_image = time.strftime("%d%b_%I-%M%p")
            print(current_time)
            local_image_path = time.strftime("%m")
            local_image_name = "CAP_{}.png".format(time_image)
            img_name = "{0}/{1}/CAP_{2}.png".format(IMAGE_SAVE_PATH,local_image_path,time_image)
            cv2.imwrite(img_name, capture)
            print("{} written!".format(img_name))

            # Storing in Firebase
            text = ocr.static_image_ocr(capture,truth)
            print(text)
            fb_store.firebase_realtime_db(text)
            fb_store.firebase_store(local_image_path,local_image_name)
    else:
        print('No')
        count = 0
        print(count)

    cv2.imshow('Object detector', frame)

    if cv2.waitKey(1) == ord('q'):
        break

# clean up
video.release()
cv2.destroyAllWindows()
```

Figure 5.3: Initialize Webcam and Object detection



```
"""
Created on Tue Apr  7 14:39:07 2020

@author: Navin Subbu
"""

# Copyright 2017 The TensorFlow Authors. All Rights Reserved.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
# ==============================================================================

"""A set of functions that are used for visualization.

These functions often receive an image, perform some visualization on the image.
The functions do not return a value, instead they modify the image itself.

"""
import abc
import collections
import functools
# Set headless-friendly backend.
import matplotlib; matplotlib.use('Agg')  # pylint: disable=multiple-statements
import matplotlib.pyplot as plt  # pylint: disable=g-import-not-at-top
import numpy as np
import PIL.Image as Image
import PIL.ImageColor as ImageColor
import PIL.ImageDraw as ImageDraw
import PIL.ImageFont as ImageFont
import six
import tensorflow as tf
import cv2 as cv

from object_detection.core import standard_fields as fields
from object_detection.utils import shape_utils


capture = 0
_TITLE_LEFT_MARGIN = 10
_TITLE_TOP_MARGIN = 10
STANDARD_COLORS = [
    'AliceBlue', 'Chartreuse', 'Aqua', 'Aquamarine', 'Azure', 'Beige', 'Bisque',
    'BlanchedAlmond', 'BlueViolet', 'BurlyWood', 'CadetBlue', 'AntiqueWhite',
    'Chocolate', 'Coral', 'CornflowerBlue', 'Cornsilk', 'Crimson', 'Cyan',
    'DarkCyan', 'DarkGoldenRod', 'DarkGrey', 'DarkKhaki', 'DarkOrange',
    'DarkOrchid', 'DarkSalmon', 'DarkSeaGreen', 'DarkTurquoise', 'DarkViolet',
    'DeepPink', 'DeepSkyBlue', 'DodgerBlue', 'FireBrick', 'FloralWhite',
    'ForestGreen', 'Fuchsia', 'Gainsboro', 'GhostWhite', 'Gold', 'GoldenRod',
    'Salmon', 'Tan', 'HoneyDew', 'HotPink', 'IndianRed', 'Ivory', 'Khaki',
    'Lavender', 'LavenderBlush', 'LawnGreen', 'LemonChiffon', 'LightBlue',
    'LightCoral', 'LightCyan', 'LightGoldenRodYellow', 'LightGray', 'LightGrey',
    'LightGreen', 'LightPink', 'LightSalmon', 'LightSeaGreen', 'LightSkyBlue',
    'LightSlateGray', 'LightSlateGrey', 'LightSteelBlue', 'LightYellow', 'Lime',
    'LimeGreen', 'Linen', 'Magenta', 'MediumAquaMarine', 'MediumOrchid',
    'MediumPurple', 'MediumSeaGreen', 'MediumSlateBlue', 'MediumSpringGreen',
    'MediumTurquoise', 'MediumVioletRed', 'MintCream', 'MistyRose', 'Moccasin',
    'NavajoWhite', 'OldLace', 'Olive', 'OliveDrab', 'Orange', 'OrangeRed',
    'Orchid', 'PaleGoldenRod', 'PaleGreen', 'PaleTurquoise', 'PaleVioletRed',
    'PapayaWhip', 'PeachPuff', 'Peru', 'Pink', 'Plum', 'PowderBlue', 'Purple',
    'Red', 'RosyBrown', 'RoyalBlue', 'SaddleBrown', 'Green', 'SandyBrown',
    'SeaGreen', 'SeaShell', 'Sienna', 'Silver', 'SkyBlue', 'SlateBlue',
    'SlateGray', 'SlateGrey', 'Snow', 'SpringGreen', 'SteelBlue', 'GreenYellow',
    'Teal', 'Thistle', 'Tomato', 'Turquoise', 'Violet', 'Wheat', 'White',
    'WhiteSmoke', 'Yellow', 'YellowGreen'
]
```

Figure 5.4: Visualizations libraries



```python
def draw_bounding_box_on_image(image,
                               ymin,
                               xmin,
                               ymax,
                               xmax,
                               color='red',
                               thickness=4,
                               display_str_list=(),
                               use_normalized_coordinates=True):
  """Adds a bounding box to an image.

  Bounding box coordinates can be specified in either absolute (pixel) or
  normalized coordinates by setting the use_normalized_coordinates argument.

  Each string in display_str_list is displayed on a separate line above the
  bounding box in black text on a rectangle filled with the input 'color'.
  If the top of the bounding box extends to the edge of the image, the strings
  are displayed below the bounding box.

  Args:
    image: a PIL.Image object.
    ymin: ymin of bounding box.
    xmin: xmin of bounding box.
    ymax: ymax of bounding box.
    xmax: xmax of bounding box.
    color: color to draw bounding box. Default is red.
    thickness: line thickness. Default value is 4.
    display_str_list: list of strings to display in box
                      (each to be shown on its own line).
    use_normalized_coordinates: If True (default), treat coordinates
      ymin, xmin, ymax, xmax as relative to the image.  Otherwise treat
      coordinates as absolute.
  """
  draw = ImageDraw.Draw(image)
  im_width, im_height = image.size
  if use_normalized_coordinates:
    (left, right, top, bottom) = (xmin * im_width, xmax * im_width,
                                  ymin * im_height, ymax * im_height)
  else:
    (left, right, top, bottom) = (xmin, xmax, ymin, ymax)
  draw.line([(left, top), (left, bottom), (right, bottom),
             (right, top), (left, top)], width=thickness, fill=color)
  try:
    font = ImageFont.truetype('arial.ttf', 24)
  except IOError:
    font = ImageFont.load_default()

  # If the total height of the display strings added to the top of the bounding
  # box exceeds the top of the image, stack the strings below the bounding box
  # instead of above.
  display_str_heights = [font.getsize(ds)[1] for ds in display_str_list]
  # Each display_str has a top and bottom margin of 0.05x.
  total_display_str_height = (1 + 2 * 0.05) * sum(display_str_heights)

  if top > total_display_str_height:
    text_bottom = top
  else:
    text_bottom = bottom + total_display_str_height
  # Reverse list and print from bottom to top.
  for display_str in display_str_list[::-1]:
    text_width, text_height = font.getsize(display_str)
    margin = np.ceil(0.05 * text_height)
    draw.rectangle(
        [(left, text_bottom - text_height - 2 * margin), (left + text_width,
                                                          text_bottom)],
        fill=color)
    draw.text(
        (left + margin, text_bottom - text_height - margin),
        display_str,
        fill='black',
        font=font)
    text_bottom -= text_height - 2 * margin
```

```python
def draw_bounding_box_on_image_array(image,
                                     ymin,
                                     xmin,
                                     ymax,
                                     xmax,
                                     color='red',
                                     thickness=4,
                                     display_str_list=(),
                                     use_normalized_coordinates=True):
  """Adds a bounding box to an image (numpy array).

  Bounding box coordinates can be specified in either absolute (pixel) or
  normalized coordinates by setting the use_normalized_coordinates argument.

  Args:
    image: a numpy array with shape [height, width, 3].
    ymin: ymin of bounding box.
    xmin: xmin of bounding box.
    ymax: ymax of bounding box.
    xmax: xmax of bounding box.
    color: color to draw bounding box. Default is red.
    thickness: line thickness. Default value is 4.
    display_str_list: list of strings to display in box
                      (each to be shown on its own line).
    use_normalized_coordinates: If True (default), treat coordinates
      ymin, xmin, ymax, xmax as relative to the image.  Otherwise treat
      coordinates as absolute.
  """
  image_pil = Image.fromarray(np.uint8(image)).convert('RGB')
  draw_bounding_box_on_image(image_pil, ymin, xmin, ymax, xmax, color,
                             thickness, display_str_list,
                             use_normalized_coordinates)
  np.copyto(image, np.array(image_pil))

def visualize_boxes_and_labels_on_image_array(
    image,
    boxes,
    classes,
    scores,
    category_index,
    instance_masks=None,
    instance_boundaries=None,
    keypoints=None,
    use_normalized_coordinates=False,
    max_boxes_to_draw=20,
    min_score_thresh=.5,
    agnostic_mode=False,
    line_thickness=4,
    groundtruth_box_visualization_color='black',
    skip_scores=False,
    skip_labels=False):
  """Overlay labeled boxes on an image with formatted scores and label names.

  This function groups boxes that correspond to the same location
  and creates a display string for each detection and overlays these
  on the image. Note that this function modifies the image in place, and returns
  that same image.

  Args:
    image: uint8 numpy array with shape (img_height, img_width, 3)
    boxes: a numpy array of shape [N, 4]
    classes: a numpy array of shape [N]. Note that class indices are 1-based,
      and match the keys in the label map.
    scores: a numpy array of shape [N] or None.  If scores=None, then
      this function assumes that the boxes to be plotted are groundtruth
      boxes and plot all boxes as black with no classes or scores.
    category_index: a dict containing category dictionaries (each holding
      category index `id` and category name `name`) keyed by category indices.
    instance_masks: a numpy array of shape [N, image_height, image_width] with
      values ranging between 0 and 1, can be None.
    instance_boundaries: a numpy array of shape [N, image_height, image_width]
      with values ranging between 0 and 1, can be None.
    keypoints: a numpy array of shape [N, num_keypoints, 2], can
      be None
    use_normalized_coordinates: whether boxes is to be interpreted as
      normalized coordinates or not.
    max_boxes_to_draw: maximum number of boxes to visualize.  If None, draw
      all boxes.
    min_score_thresh: minimum score threshold for a box to be visualized
    agnostic_mode: boolean (default: False) controlling whether to evaluate in
      class-agnostic mode or not.  This mode will display scores but ignore
      classes.
    line_thickness: integer (default: 4) controlling line width of the boxes.
    groundtruth_box_visualization_color: box color for visualizing groundtruth
      boxes
    skip_scores: whether to skip score when drawing a single detection
    skip_labels: whether to skip label when drawing a single detection

  Returns:
    uint8 numpy array with shape (img_height, img_width, 3) with overlaid boxes.
  """
```

Figure 5.5: Object Detection and draw bounding boxes

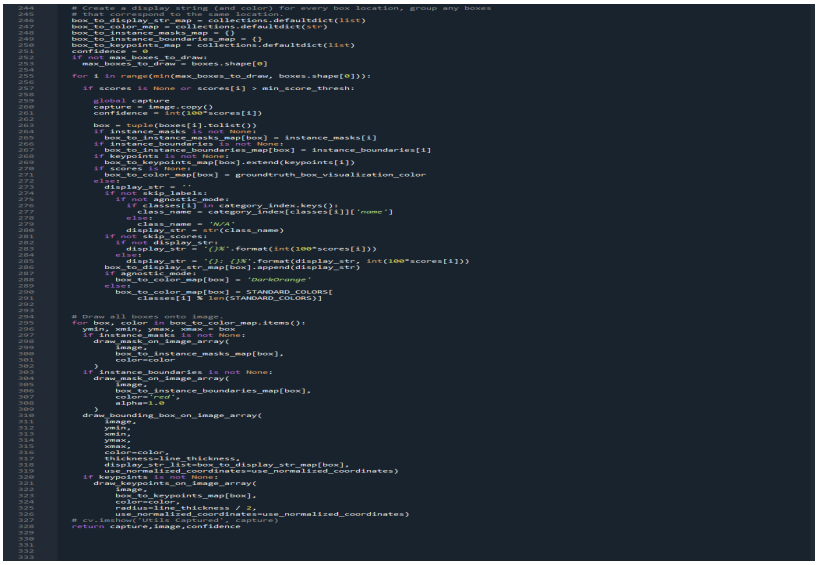


Figure 5.6: Confidence calculations and capture Image

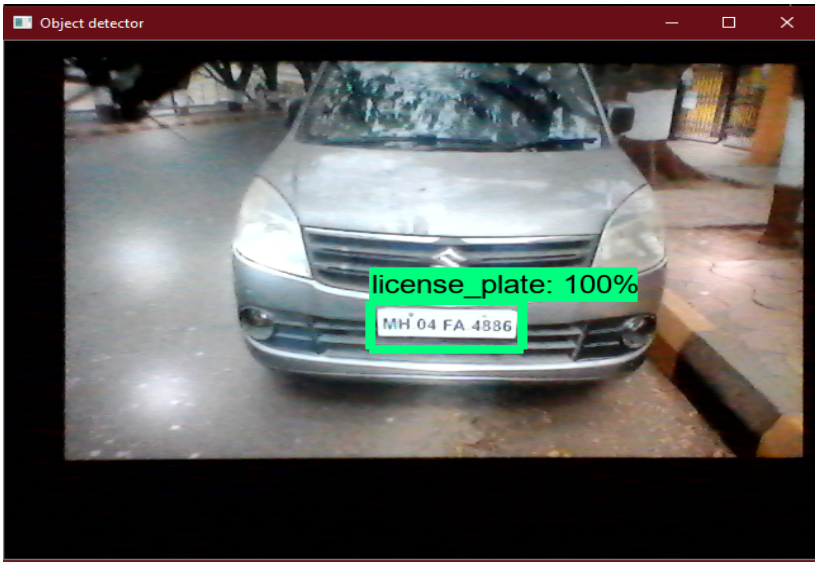


Figure 5.7: Output of Object Detection

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 the 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 whote 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 recognised 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.

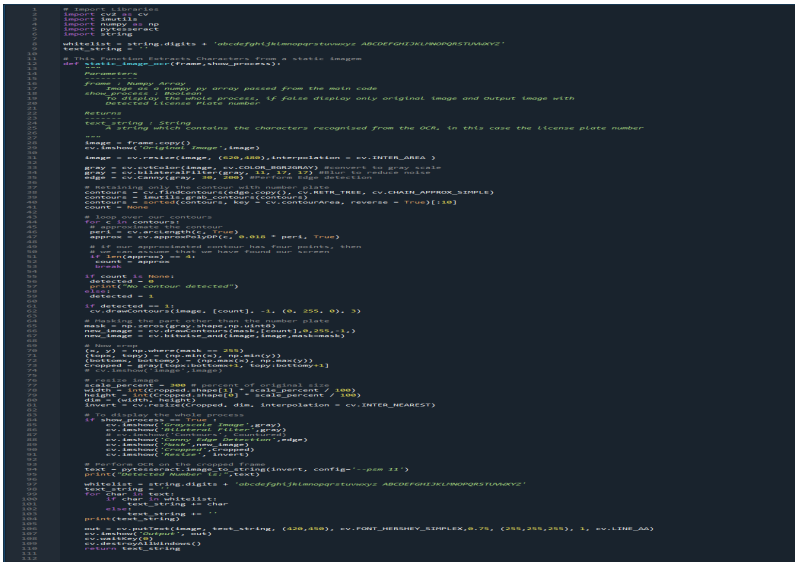


Figure 5.8: Image Segmentation

Image Processing and Optical character recognition:

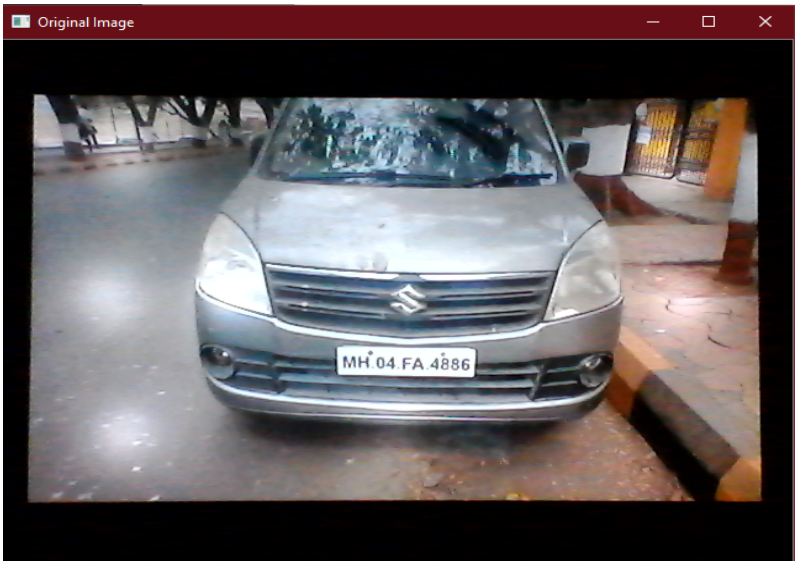


Figure 5.9: Original Image

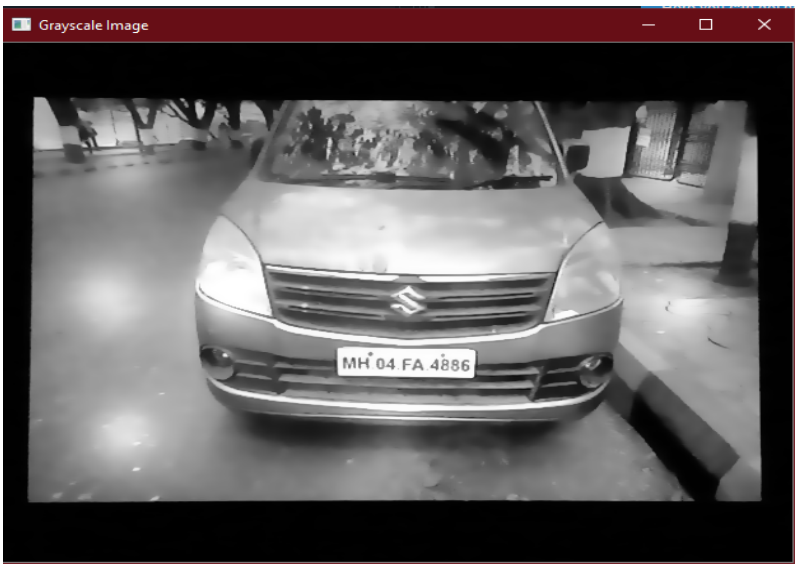


Figure 5.10: Grayscale Image

Figure 5.11: Filtered Image

Figure 5.12: Canny Edge Detection

Figure 5.13: Detected Contours

Figure 5.14: Contour Containing License Plate

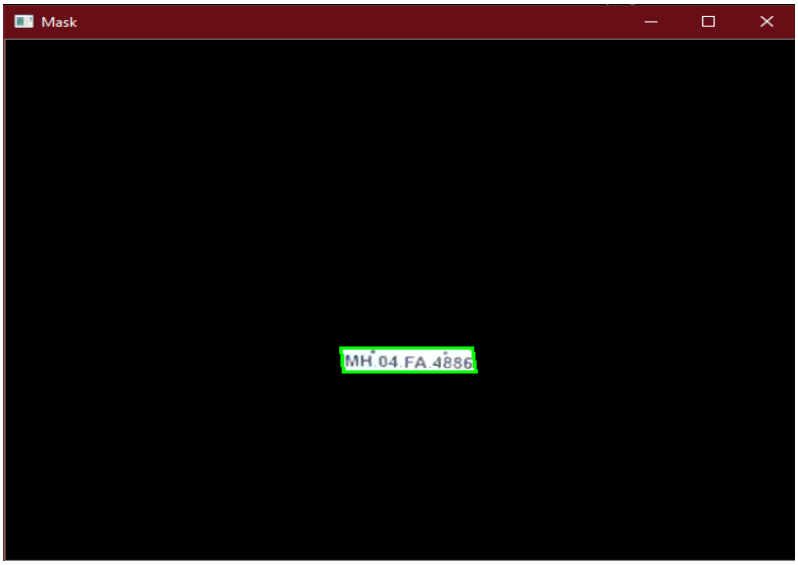


Figure 5.15: 7xx Masked Image

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.

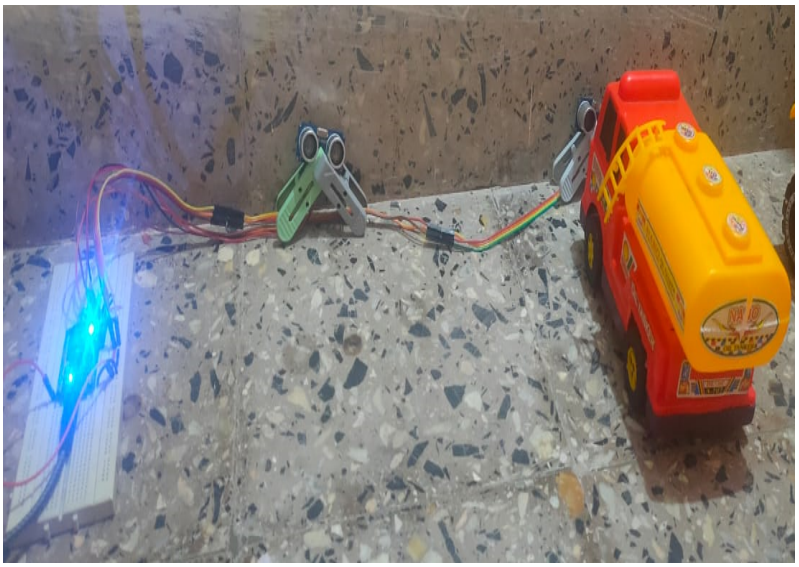
```
"""
Created on Fri Apr 17 18:08:19 2020

@author: Navin Subbu

"""
from firebase import firebase
import pyrebase

# Firebase Initialization wiht App secrets
firebase =  firebase.FirebaseApplication("https://python-test-1235f.firebaseio.com/",None)
config = {
    "apiKey" : "AIzaSyAx3rxrUmCOE5oCdj3NP1x_ITyeC_YYjno",
    "authDomain" : "python-test-1235f.firebaseapp.com",
    "databaseURL" : "https://python-test-1235f.firebaseio.com",
    "projectId" : "python-test-1235f",
    "storageBucket" : "python-test-1235f.appspot.com",
    "messagingSenderId" : "707011126206",
    "appId" : "1:707011126206:web:93df56c96e28b114b48fef",
    "measurementId" : "G-8YHDDWVLP7"
    }


def firebase_store(path,image_name) :
    """
    This Fucntion stores a given file into Google Firebase Storage in
    a Specifed Location categorised in the order of Year and Month
    Parameters
    ----------
    path : String
        It is the Path of the saved image
    image_name : String
        It is the name of the Image

    Returns
    -------
    None.

    """
    firebase = pyrebase.initialize_app(config)
    storage = firebase.storage()
    path_on_cloud = "Car Recorded/{0}/{1}".format(path,image_name)
    path_local = "detected/{0}/{1}.".format(path,image_name)
    storage.child(path_on_cloud).put(path_local)


def firebase_realtime_db(text) :

    """
    This function pushes a given string into Realtime Database in Google Firebase

    Parameters
    ----------
    text : String
        Its a string, the license plate number
    Returns
    -------

    None.

    """
    result = firebase.post('/Detected', text)
    print(result)
```

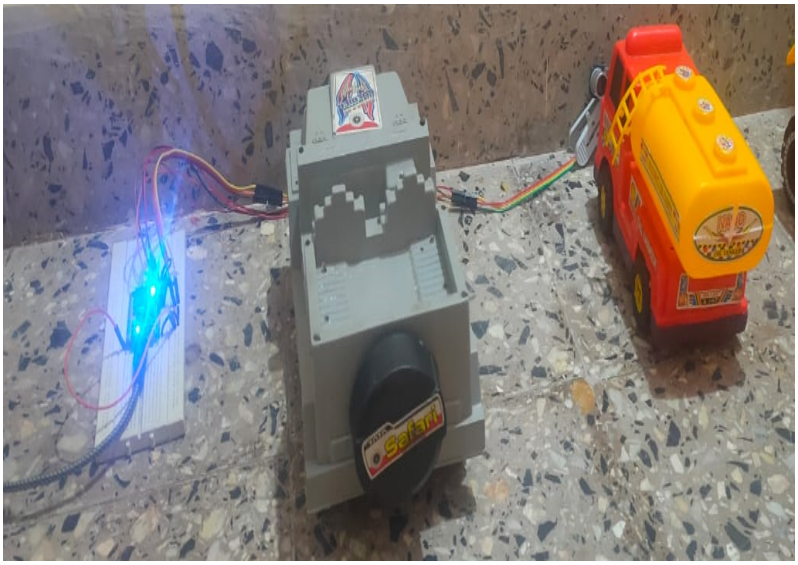
Figure 5.18: Firebase Real-Time Database

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 "bk1ScvsVBNfLitAt4pZO3EqEtM6SfC2cLn7MJAcb"
#define WIFI_SSID "Anonymous"
#define WIFI_PASSWORD "batman123"

//Firebase RTDB Path
const String path = "/Parking";
const String Floor = "/Floor 1"; //initalise 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 initials 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", 0); // Parking space Occupied
    }
    else
    {
      Firebase.setInt(Floor + "/Parking 2", 1); // 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

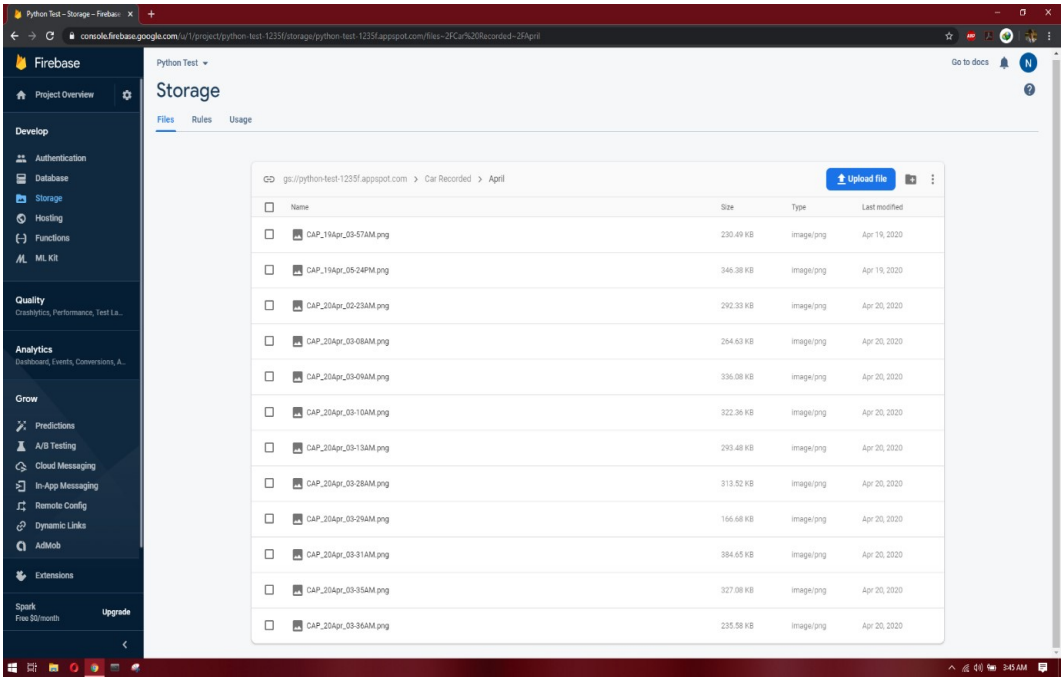


Figure 6.4: Firebase image store

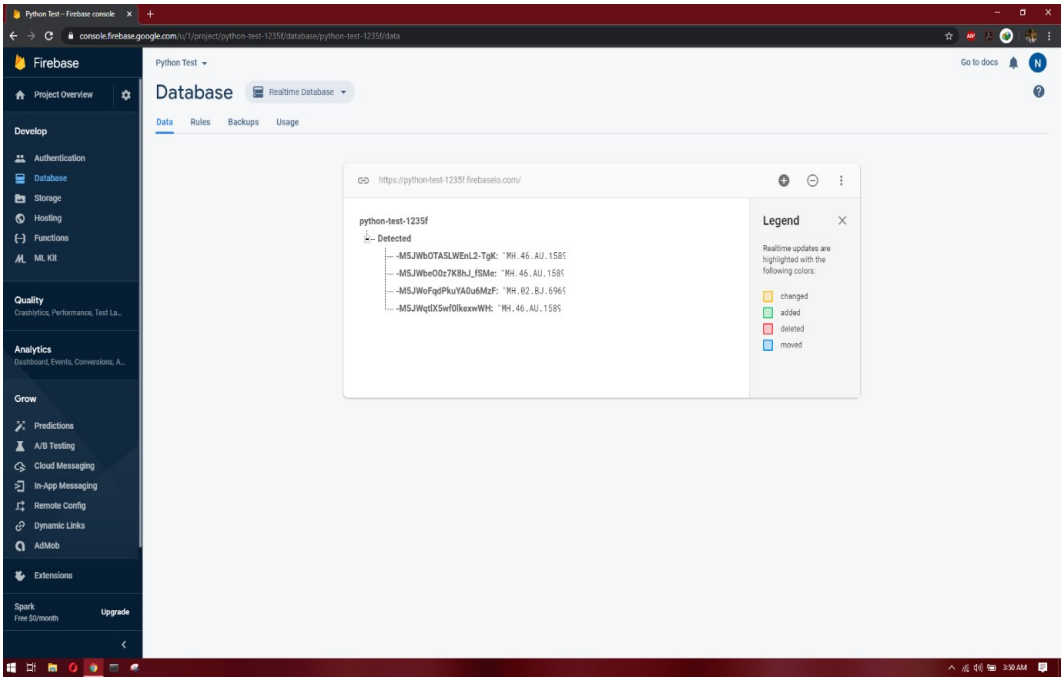


Figure 6.5: Realtime database

6.3 Database Output – IoT

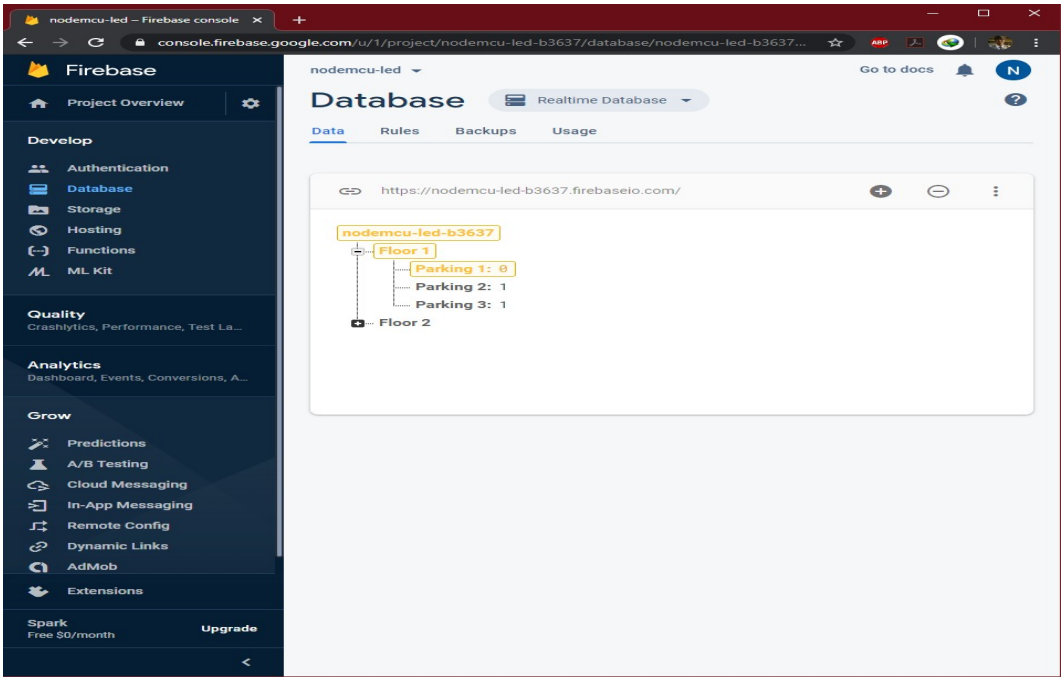


Figure 6.6: When Parking Lot 1 is unoccupied

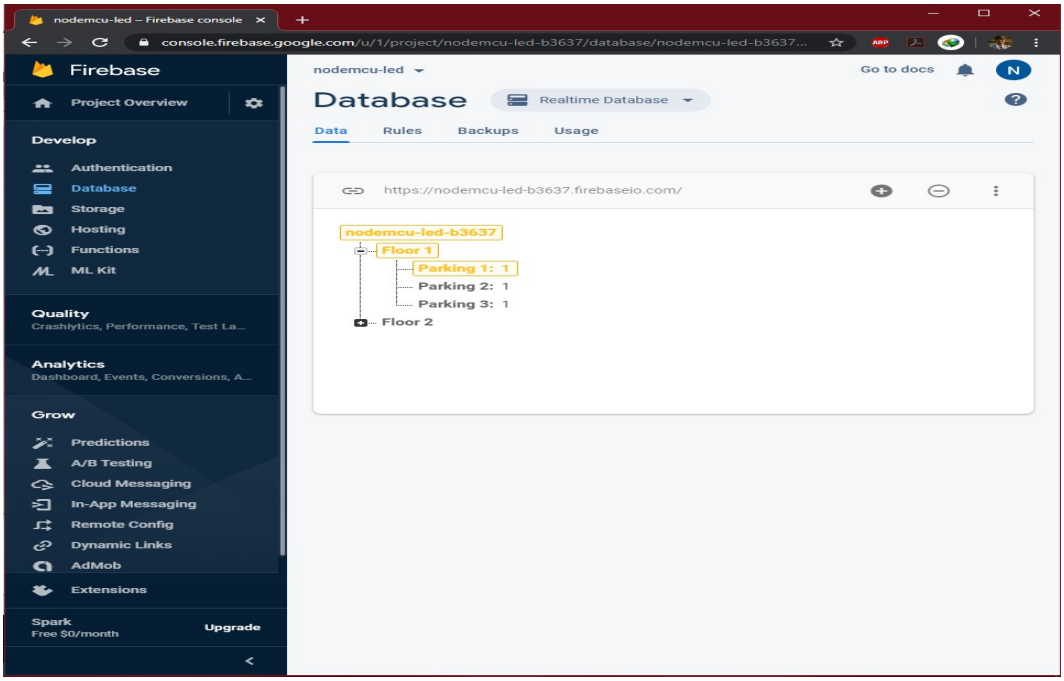


Figure 6.7: When Parking Lot 1 is occupied

6.4 Webpage Output

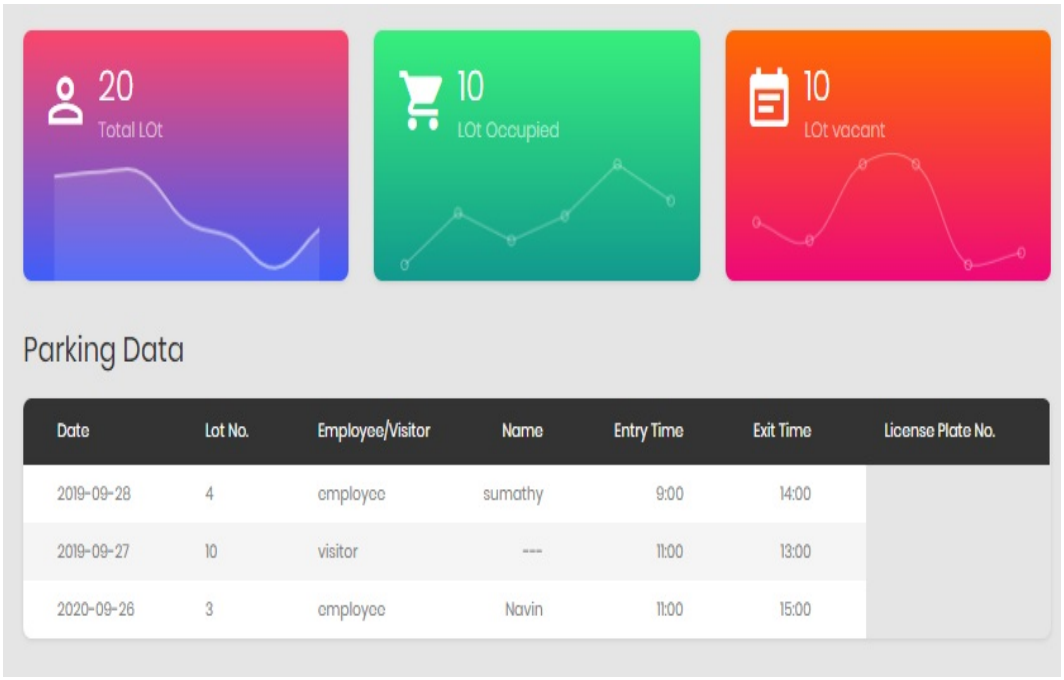


Figure 6.8: web page front end

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. Barbuzza, "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=7

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$_{c}haracter_{r}ecognition$

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$_{(}software$)

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

[9] The Raspberry P i is a low cost, credit-card sized computer that plugs into a computer monitor or TV, and u ses 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 1,2108 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/@v
deeper-look-at-how-faster-rcnn-works-84081284e1cd