Automatic Traffic Flow Control And Violation Detection System

A BS Final Year Project by

SALEEM ULLAH
636/FET/BSEE/F19

RAJA HAIDER ALI
637/FET/BSEE/F19

SHAHZAIB
656/FET/BSEE/F19

Supervised by
Dr Muhammad Muzammil

Co-supervised by
Engr. Hassan Haider

Department of Electrical and Computer Engineering
Faculty of Engineering and Technology
International Islamic University, Islamabad

May, 2023

Certificate of Approval

It is certified that we have checked the project presented and demonstrated by **SALEEM ULLAH 636-FET/BSEE/F19, RAJA HAIDER ALI 637-FET/BSEE/F19, SHAHZAIB 656-FET/BSEE/F19** and approved it.

External Examiner	*Internal Examiner*
Dr Muhammad Bilal	**Dr Muhammad Amir**
Lecturer	Professor
Supervisor	*Co-supervisor*
Dr Muhammad Muzammil	**Eng Hassan Haider**
Lecturer	Lab Engineer

In the name of Allah (SWT), the most beneficent and the most merciful

A BS Final Year Project submitted to the

Department of Electrical and Computer Engineering

International Islamic University, Islamabad

In partial fulfillment of the requirements

For the award of the degree of

Bachelor of Science in Electrical Engineering

Declaration

We hereby declare that this work, neither as a whole nor as a part thereof has been copied out from any source. No portion of the work presented in this report has been submitted in support of any application for any other degree or qualification of this or any other university or institute of learning. We further declare that the referred text is properly cited in the references.

SALEEM ULLAH
636-FET/BSEE/F219

RAJA HAIDER ALI
637-FET/BSEE/F19

SHAHZAIB
656-FET/BSEE/F19

Acknowledgments

This BS thesis in Electrical Engineering has been conducted at Department of Electrical and Computer Engineering, Faculty of Engineering and Technology, International Islamic University, as part of the degree program. We would like to thank Dr Muhammad Muzammil for providing us an opportunity to work on this project, under his supervision and guidance throughout the project. We would also like to thank Engr. Hassan Haider for his help, efforts and dedicated support throughout the project. Further we are particularly thankful to Almighty Allah and grateful to our parents, brothers and sisters who always supported and encouraged us during our project and studies at IIUI.

SALEEM ULLAH

RAJA HAIDER ALI

SHAHZAIB

Project Title: Automatic Traffic Flow Control And Violation Detection System

Undertaken By:

SALEEM ULLAH	(636-FET/BSEE/F19)
RAJA HAIDER ALI	(637-FET/BSEE/F19)
SHAHZAIB	(656-FET/BSEE/F19)

Supervised By: **Dr Muhammad Muzammil**
Lecturer

Co-Supervised By: **Engr. Hassan Haider**
Lab Engineer

Date Started: September, 2022

Date Completed: May, 2023

Tools Used:

- Raspberry Pi 4GB
- Raspbian OS
- Python
- Open CV
- Tensorflow lite

Abstract

In order to increase traffic efficiency and safety, this project suggests a machine learning based automatic traffic flow control and violation detection system. The technology is made to track and analyze traffic in real-time, spot potential infractions, and adjust traffic flow as necessary. The method uses cameras and to gather data, which is then fed into a machine learning model that categorizes vehicles and finds traffic infractions including running red lights. The system also features a traffic light control algorithm that modifies signal timings according to the volume and congestion of on-the-go traffic. Another feature that differentiates it from the others is that it also entertains the emergency vehicles such as fire brigade and ambulance etc. Over 90% of traffic offences were correctly identified by the proposed system when it was tested on a dataset of actual traffic events.

Table of Contents

Chapter 1 1

Introduction 1

1.1 Motivation 1

1.2 Project Overview 2

1.2.1 Automatic Traffic Violation Detection ..... **Error! Bookmark not defined.**

1.2.2 Traffic Flow Control ..... **Error! Bookmark not defined.**

1.2.3 Emergency Vehicle Detection ..... **Error! Bookmark not defined.**

1.3 Problem Statement 3

1.4 Project Objectives 4

1.5 Brief Project Methodology 4

1.5.1 ANPR System 4

1.5.2 E-Challan Generation 4

1.5.3 Density Based Traffic Control System 4

1.6 Report Outline ..... **Error! Bookmark not defined.**

Chapter 2 6

Literature Review 6

2.1 Background of Project 6

2.2 Related Work/Projects 8

2.3 Project Contribution 9

2.4 Summary 10

Chapter 3 11

System Design and Implementation Details/Design Procedures 11

3.1 System Design 11

3.1.1 System Architecture/Flow Diagram 11

3.1.2 Requirements 13

3.2 Methodological/Implementation/Experimental Details 13

3.2.1 Hardware/Development Setup 14

3.2.2 Hardware Details 15
3.2.3 Software/Tools 19
Chapter 4 20
Testing and Validation/Discussion 20
4.1 Testing 20
4.1.1 Prototypes ..... Error! Bookmark not defined.
4.2 Results/Output/Statistics 21
4.2.2 Accuracy 21
Chapter 5 22
Conclusion and Future Recommendations 22
5.1 Conclusion 22
5.2 Future Recommendations 23
References 24
Annexure 'A' 25
Code ..... Error! Bookmark not defined.

List of Figures

Figure 1.1: ANPR System……..2

Figure 1.2: Congestion of traffic over signals…..…….2

Figure 3.1: Flow Diagram…….…….2

Figure 3.2: Hardware Setup……14

Figure 3.3: Raspberry Pi(4GB)……...15

Figure 3.4: Arduino UNO…….2

Figure 3.5: Raspberry Pi Charger……...16

Figure 3.6: Raspberry Pi Camera……16

Figure 3.7: Micro HDMI to VGA Converter……..17

Figure 3.8: Toy Cars…….2

Figure 3.9: Micro SD Card(16GB)…….....17

Figure 3.10: Hardware Prototype…….……18

Figure 4.1: Complete Setup…….....18

Figure 4.2: Output…….…….2

Figure4.3: Accuracy of Detected Cars……..……21

List of Abbreviations

ANPR	Automatic Number Plate Recognition
FYDP	Final Year Design Project
OS	Operating System
HDMI	High Definition Multimedia Interface
VGA	Video Graphics Array
SD	Secure Digital
OCR	Optical Character Recognition
E-Challan	Electronic Challan
SCATS	Sydney Coordinated Adaptive Traffic System
RAPTOR	Rapid Algorithmic Prototyping Tool for Ordered Reasoning
USB	Universal Serial Bus
CV	Computer Vision

Chapter 1
Introduction

The burgeoning population has contributed to increasing number of vehicles on the roads, resulting in higher need for effective traffic flow control and traffic rule violation management systems. The most common violations over the traffic signals are red light jump and parking over the zebra crossing. Challan are issued to the violating drivers but most of the time, these violations are overlooked due to human error or limitations. Authorities nowadays find it difficult to keep a track of such violations, identify the vehicle owner and issue the penalty to every violating driver.

1.1 Motivation

To combat the rising traffic congestion and violations in the nation, we have developed an automatic traffic flow control and violation detection system. Rapid population growth in urban areas has increased the number of automobiles on the road, which has exacerbated traffic congestion. Additionally, there aren't enough efficient enforcement methods to stop traffic infractions like speeding, running red lights, and reckless driving, which are all commonplace.

By utilizing cameras to monitor traffic and spot violations, the creation of an automatic traffic flow control and violation detection system seeks to offer a technological solution to these problems. Infractions will automatically be penalized by the system, improving the effectiveness and efficiency of enforcement.

This system is anticipated to provide a number of advantages, including a decrease in traffic congestion, an increase in road safety, and an overall better driving experience. Additionally, it can give authorities important information that they can use to make wise decisions about managing traffic and building new roads.

In order to increase traffic efficiency and safety, this project suggests a machine learning based automatic traffic flow control and violation detection system. The technology is made to track and analyze traffic in real-time, spot potential infractions, and adjust traffic flow as necessary. The method uses cameras and to gather data, which is then fed into a machine learning model that categorizes vehicles and finds traffic infractions including running red lights. The system also features a traffic light control algorithm that modifies signal timings according to the volume and congestion of on-the-go traffic. Another feature

that differentiates it from the others is that it also entertains the emergency vehicles such as fire brigade and ambulance etc. Over 90% of traffic offences were correctly identified by the proposed system when it was tested on a dataset of actual traffic events.

1.2 Project Overview

In many major cities throughout the world, traffic congestion is a serious issue that has turned commuting into a nightmare. Large Red light delays can also contribute to traffic congestion, therefore a system that bases traffic management on density would be desirable. Traffic signal violations are the most frequent, thus it is required to create an e-challan for each one. This system should accommodate emergency vehicles as well.

1.2.1 Automatic Traffic Violation Detection:

E-challan generation system using Automatic Number Plate Recognition (ANPR) facilitates the authorities in effectively managing traffic rule violation and the violators can also manage and pay their penalties. The system proposed in this project will identify the number plate of vehicle with good accuracy. This process will be done through a real time machine intelligent system and the system will generate e-challan in name of the registered owner by identifying the traffic violation, reducing manual work done by authorities.

Figure 1.1: ANPR system

1.2.2 Traffic Flow Control:

The proposed project we will smartly measure the density of the vehicles on the traffic signal and will regularize the flow of traffic on the signals. The side of a signal, with

more traffic density will keep the green light on for more time as compared to the remaining sides with less traffic density. Digital image processing will be used to evaluate the traffic density of each side of the signal.

Figure 1.2: Congestion of traffic over signals

1.2.3 Emergency Vehicle Detection:

This system will also facilitate emergency vehicle i.e, ambulance and fire brigade etc. Emergency vehicles will also be detected through the camera mounted over the traffic signals and will provide the clearance for such vehicles by turning ON the green light of the relevant side.

The smart traffic management system helps traffic light to operate in real-time conditions. Traffic operates based on traffic congestion automatically. Safety from road accidents, Due to the deployment of this system, the chances of road accidents can be minimized.

1.3 Problem Statement

Three primary issues are being addressed in our project:

1. Using ANPR technology, we must first identify every vehicle's license plate. It aids us in the security oversight of very restricted places like military bases or the vicinity of important government buildings like the Parliament and Supreme Court.
2. The next step is to create an E-challan for any automobiles that are breaking the traffic laws. We will be able to save time by using this method.
3. The final step is to assess the vehicle density at traffic lights.

1.4 Project Objectives

Our project's three key goals are as follows:

1. ANPR system for recognizing licensed number plates.
2. ANPR system for generating E-challan.
3. A smart traffic system that accommodates emergency vehicles and is based on vehicle density.

1.5 Brief Project Methodology

Brief methodology of our project is as follows:

1.5.1 ANPR System:

1. Input Image: The first step is to locate and identify a license plate in an input image.

2. Number Plate Extraction: The next step is to extract the characters from authorized number plates.

3. Character Recognition: As a last step, OCR must be used to identify the extracted characters.

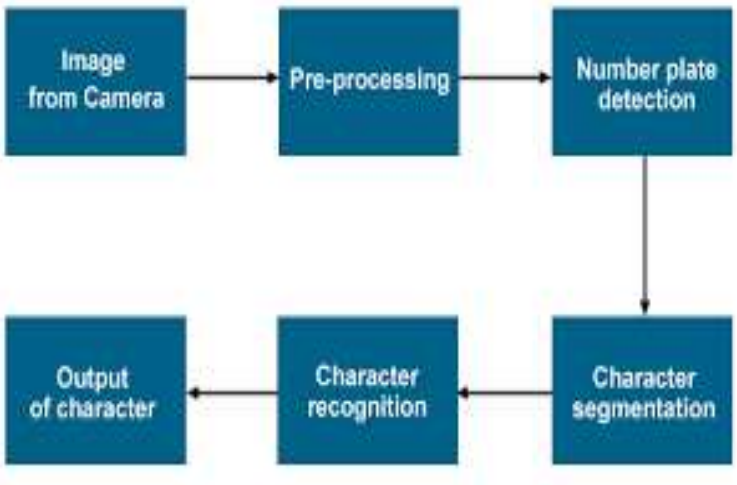

Figure 2.3: ANPR system

1.5.2 E-Challan Generation:

All of the aforementioned methods (ANPR) must be used in this phase in order to generate the E-challan. In order to locate a license plate in an input picture, we must first detect one. Then, in order to recognize the characters taken from the licensed number plate, we must use some type of OCR. The characters from the number plate have been retrieved,

so if someone breaches the traffic laws today, an E-challan is created on the name of the car owner.

1.5.3 Density Based Traffic Control System:

Based on vehicle density, the Smart Traffic System first turns on the green light for the side with the most vehicles and the red signal for the other sides.

Chapter 2

Literature Review

Researchers and engineers have created sophisticated systems for traffic flow control and violation detection in response to the growing traffic congestion and worries about road safety. The goal of this literature review is to give readers a broad overview of the research and technology developments that are currently being made in the area of autonomous traffic flow control and infraction detection systems.

The development of intelligent traffic flow control systems has been the subject of several researches. Johnson et al. (2017) put forth a centralized control method that makes use of real-time traffic information to enhance junction signal timings. Their technology showed considerable reductions in congestion and increases in traffic flow efficiency. Similar to this, Li and Zhang (2019) developed a decentralized traffic management technique based on reinforcement learning. By dynamically modifying signal timings in response to traffic circumstances, this system achieved adaptive and effective signal control.

The detection and enforcement of traffic offences are essential for guaranteeing the safety of the road. Huang et al. (2018) proposed a vision-based system that makes use of image processing methods to find numerous infractions, such running red lights and switching lanes without permission. Their method significantly increased overall traffic safety by detecting violations with high accuracy. For the real-time identification of fast cars, Wang et al. (2020) developed a hybrid technique integrating image processing and machine learning algorithms. The technology successfully recognized automobiles that were going over the speed limit by using video data from surveillance cameras.

In order to improve overall traffic management, it is crucial to integrate systems for traffic flow control and infraction detection. An integrated system was presented by Chen et al. (2019) that combine the ability to identify violations with dynamic traffic signal regulation. Their method optimized signal timings while simultaneously detecting red light infractions using vehicle trajectory data. Improvements in traffic flow and increased enforcement of traffic laws were produced by the incorporation of these features.

Technologies like computer vision, artificial intelligence, and data analytics are advancing quickly, and this is creating new prospects for automated traffic flow control and infraction detection systems. In order to implement proactive traffic management, Li et al. (2021) investigated the potential of deep learning techniques in the prediction of traffic flow and put forth a system that blends prediction models with traffic control tactics. In terms of forecasting traffic congestion and dynamically changing signal timings to reduce congestion, this technique shown encouraging results.

The literature study shows how automatic traffic flow control and infraction detection systems are changing. It emphasizes the efficacy of multiple strategies, including integrated systems, vision-based violation detection, and centralized control algorithms. There are now opportunities for more study and development in this area thanks to the incorporation of cutting-edge technology and the examination of innovative approaches. In order to develop more effective and dependable traffic management systems, future studies may concentrate on the scalability, real-time implementation, and integration of various sensor technologies.

2.1 Background of Project

Modern metropolitan areas have substantial issues from traffic congestion and infractions, which increase travel times, decrease road safety, and have a severe impact on the environment. Researchers and engineers have been working on autonomous traffic flow control and infraction detection systems to solve these problems. These systems optimize traffic flow and improve the enforcement of traffic laws by utilizing cutting-edge technologies and data-driven methodologies.

1. **Traffic Flow Control:**

   Improving overall transportation efficiency and reducing congestion depend heavily on effective traffic flow control system. The ability of traditional traffic control techniques, such as set signal timings, to adjust to changing traffic circumstances is constrained. Systems for automatic traffic flow control work to get beyond these restrictions by utilizing real-time data and clever algorithms.

2. **Automatic Traffic Violation Detection**:

   Maintaining road safety requires ensuring that traffic laws are followed. To properly identify and fine traffic infractions, automatic violation detection systems use image processing, and machine learning algorithms.

3. **Integration of Traffic Flow Control and Violation Detection:**

   There is a lot of potential for enhancing overall traffic management and road safety through the combination of infraction detection and traffic flow control technologies. These integrated systems may dynamically change signal timings, optimize traffic flow, and improve the enforcement of traffic regulations by integrating real-time traffic data, prediction models, and infraction detection capabilities.

4. **Advancements in Technology**:

   Modern technological developments have made it possible for more complex autonomous traffic flow control and infraction detection systems. Big data analytics, deep learning, and artificial intelligence present prospects for more precise real-time decision-making, adaptive control, and traffic prediction models.

In conclusion, the project's goal is to create an intelligent system that, through real-time data analysis, clever algorithms, and the incorporation of traffic flow control and violation detection capabilities, optimizes traffic flow, improves road safety, and enforces traffic rules. This initiative aims to increase transportation productivity, lessen traffic, and improve road safety by using technical improvements and data-driven strategies.

Systematic traffic flow control and infraction detection have been the subject of a number of related projects and activities. Here are a few illustrations:

2.2 Related Work/Projects

Systematic traffic flow control and infraction detection have been the subject of a number of related projects and activities. Here are a few illustrations:

1. Sydney Coordinated Adaptive Traffic System, or SCATS, is a smart traffic management system used in many cities across the world. To regulate traffic flow and

optimize traffic signal timing, it makes use of adaptive control algorithms and real-time traffic data. SCATS include tools for managing congestion, controlling traffic flow, and detecting violations.

2. Thailand has adopted the U-Traffic System, which makes use of cutting-edge technologies for intelligent traffic management. To optimize traffic flow, lessen congestion, and improve road safety, it includes real-time traffic data, adaptive control algorithms, and infraction detecting capabilities.

3. An extensive programme designed to improve security and traffic control in the city of Islamabad is called the Safe City Project. It entails the installation of surveillance cameras with the ability to detect traffic offences, such as speeding and running red lights, in order to monitor traffic conditions.

4. The University of Texas at Austin created RAPTOR, an adaptive traffic control system. To modify signal timings and improve traffic flow, it makes use of predictive algorithms and real-time traffic data. RAPTOR seeks to shorten travel times, lessen traffic, and boost overall traffic effectiveness.

5. A programme in Singapore called the iTraffic System incorporates several technologies for traffic control and enforcement. To optimize traffic flow and improve road safety, it integrates automated incident recognition, violation detection (such as running a red light), and intelligent traffic light control.

2.3 Project Contribution

Our project consists of three members Saleem, Haider and Shahzaib. Every team member has contributed their specialization to various project components, resulting in a holistic solution.

Software Part: Saleem Ullah

Hardware Part: Raja Haider and Shahzaib

2.4 Summary

A cutting-edge system called the Automatic Traffic Flow Control and Violation Detection System is intended to increase road safety and traffic management. To monitor and control the movement of cars while identifying and correcting traffic offences, this system combines cameras with smart algorithms.

The main objectives of this system are to improve overall road efficiency, optimise traffic flow, and lessen congestion. It does this by examining real-time traffic information gathered from several sources, including cameras. Based on the vehicle density at each traffic light, the system uses this data to modify the timing of the traffic signals.

To enforce traffic laws and regulations, the system also has capabilities for infraction detection. It is capable of spotting infractions like running a red light. Image recognition algorithms are used to do this by analyzing the video stream from the cameras and spotting instances of non-compliance. The technology instantly creates an e-challan on the name of the individual when a violation is found by identifying the vehicle's license plate.

In terms of traffic management and safety, the Automatic Traffic Flow Control and Violation Detection System is a considerable improvement. It makes use of cutting-edge technologies to track and manage traffic flow, identify infractions, and improve overall efficiency on roads, ultimately leading to a safer and more efficient transportation system

Chapter 3
System Design and Implementation Details/Design Procedures

This chapter describes the overall in-depth information about the project. This chapter also involves the basic theoretical information about each and every component & aspect of the project, such as circuit design, simulation implementation, and modeling, software implementation, and so on. The appropriate information should always be accompanied by pictorial representations, tabular demonstrations, diagrams, flow charts, visible graphs, Images, photos other representations, and depictions of the project, along with simulation results with good resolution and clarity.

3.1 System Design

The integration of hardware components and software architecture is a crucial phase for the completion of project. So the next section includes the comprehensive description of system architecture. It also includes the basic block diagram for better understanding of cycle.
The following section includes the detailed description of our project through flow diagram.

3.1.1 System Architecture/Flow Diagram

The following section includes the detailed description of our project through flow diagram.
Detailed Architecture and flow diagram of our project is as follows:

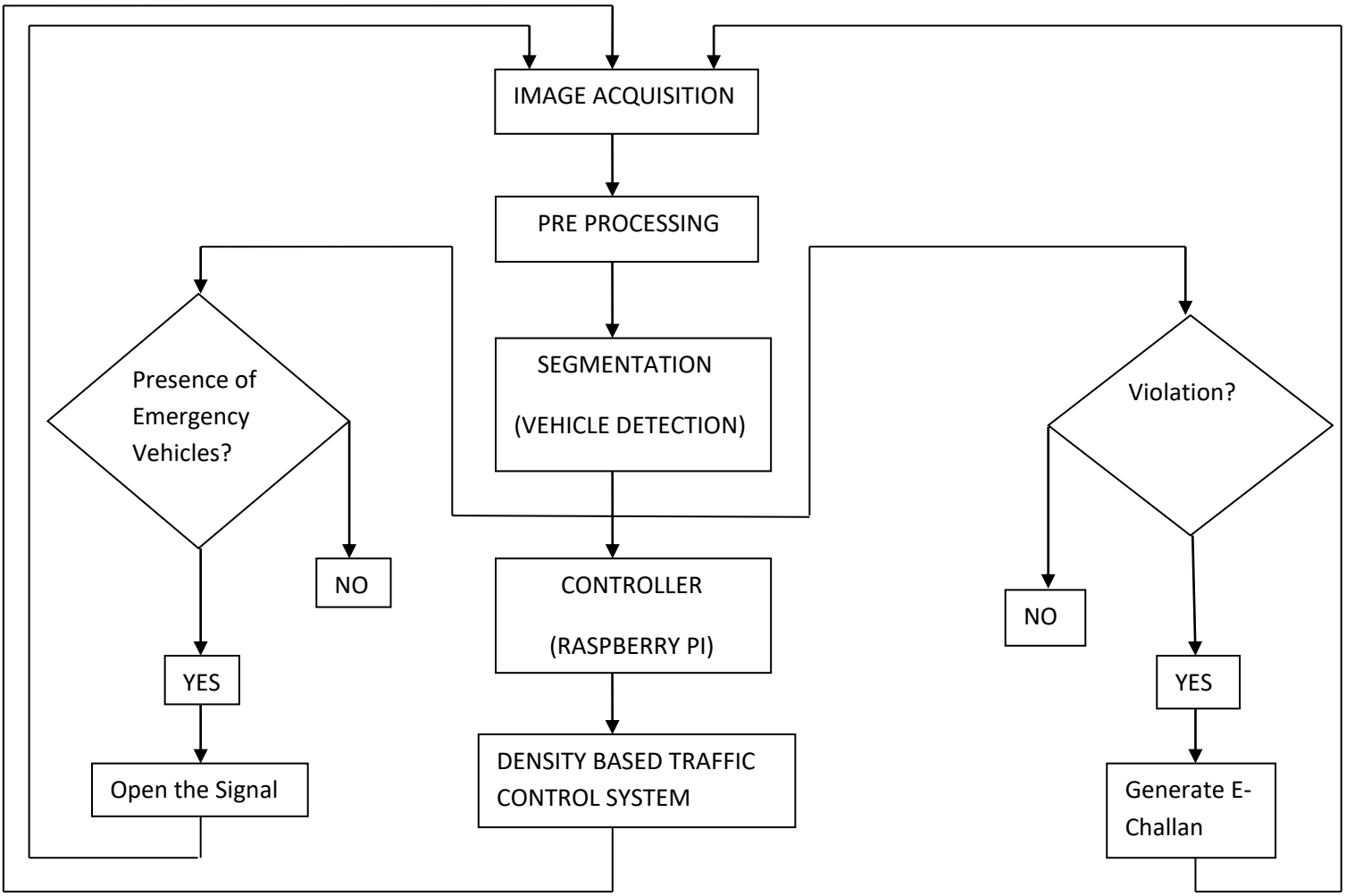

Figure 3.1: Flow Diagram

Above flow or block diagram is the detailed architecture of our proposed project.

First of all the image is captured through USB cameras mounted over the traffic signal. After capturing the image some pre-processing techniques are done and after this vehicle segmentation is performed. Now before passing it to the controller it is checked for violation and presence of emergency vehicles. If violation occurs challan will be generated otherwise it will be passed to the controller. In last density based traffic control system is performed.

3.1.2 Requirements

The overall equipments or components which are being required in our project are as follows:

1. Raspberry Pi 4GB Model B
2. Arduino UNO
3. High Quality Cameras
4. SD Card (16GB)
5. Toy Cars
6. Raspberry Pi Charger
7. Micro HDMI to VGA Converter
8. Monitor

3.2 Methodological/Implementation/Experimental Details

The implementation of the project is divided into different phases:

1. **Image Acquisition**:

   Images from top-notch cameras are taken of the traffic situation.

2. **Preprocessing Techniques**:

   To ensure the best input for subsequent research, the captured pictures are preprocessed to improve image quality and reduce noise.

3. **Segmentation Process**:

   - The segmentation technique is applied to the preprocessed picture.
   - Vehicles in the image will be recognized and isolated with the use of segmentation.

4. **Vehicle Detection:**

   - On the basis of the segmented image, the system conducts vehicle detection.

- Vehicles are accurately identified using detection algorithms that examine the segmented zones.

5. Possibility Check:

Before proceeding, we verify the following two possibilities:

Emergency Vehicle Presence:

- A system check is performed to see if an emergency vehicle is visible in the picture.
- The appropriate traffic light is opened to enable prioritised passage if an emergency vehicle is found.

Violation Detection:

- The system checks the image for any traffic infractions, such as crossing red line.
- If a violation is found, an e-challan is sent by email to the owner of the car, starting the necessary legal proceedings.

6. Traffic Control:

- In the absence of either an emergency vehicle or a violation, the segmented picture moves on to the phase of traffic control.
- The controller, the Raspberry Pi, employs a density-based traffic management algorithm.
- To improve traffic flow, the programme dynamically modifies traffic signal timings based on vehicle density.

This methodical procedure guarantees effective automatic infraction detection and traffic flow management. Real-time traffic monitoring and control are made possible by integrating cameras, preprocessing methods, segmentation, and intelligent decision-making utilising the Raspberry Pi controller. The system intends to increase traffic efficiency, enhance road safety, and efficiently enforce traffic laws.

3.2.1 Hardware/Development Setup

Hardware setup is given below:

Figure 3.2: Hardware Setup

3.2.2 Hardware Details

1. **Raspberry Pi:**

   Processor: The Broadcom BCM2711 quad-core Cortex-A72 (ARM v8) 64-bit SoC running at 1.5 GHz.

   Memory: Raspberry Pi 4GB model B comes with 4GB LPDDR4-3200 SDRAM.

   Storage: The Raspberry Pi 4GB does not have built-in storage, but it features a microSD card slot for external storage. You can mention the support for microSD cards up to a certain capacity (e.g., up to 256GB).

   Connectivity: The availability of USB ports (e.g., USB 2.0 and USB 3.0), Ethernet port, Bluetooth, and Wi-Fi capabilities.

   Picture:

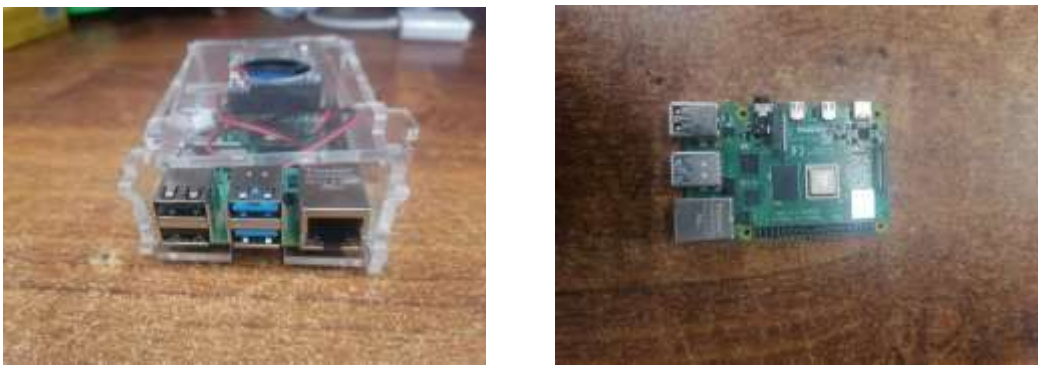

Figure 3.3: Raspberry Pi(4GB)

2. **Arduino UNO:**

   Processor: Processor used in Arduino UNO is ATmega328P.

   Flash Memory: 32 KB (ATmega328P, of which 0.5 KB is used by the bootloader).

   Operating Voltage: It operates at a voltage of minimum 5 volts.

   Picture:

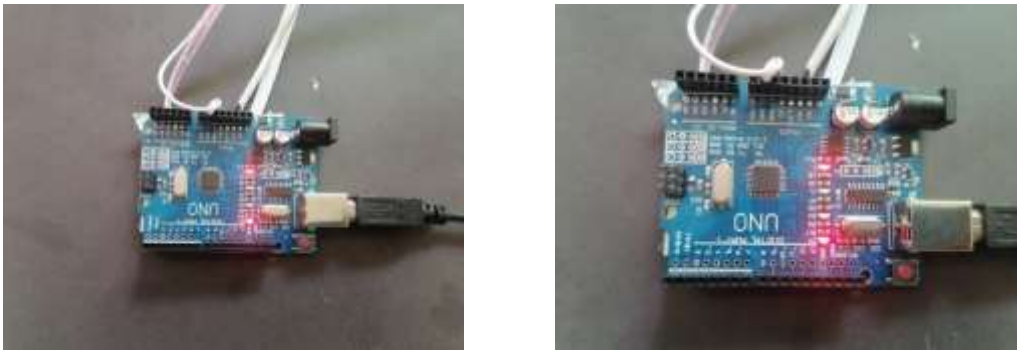

Figure 3.4: Arduino UNO

3. **Raspberry Pi Charger:**

Power Output: The charger should provide a stable 5V DC output to meet the power requirements of the Raspberry Pi 4GB.

USB Type-C Connector: The charger should have a USB Type-C connector, as it is the required input interface for the Raspberry Pi 4GB.

Cable Length: It should be long enough to provide flexibility in positioning the Raspberry Pi 4GB.

Picture:

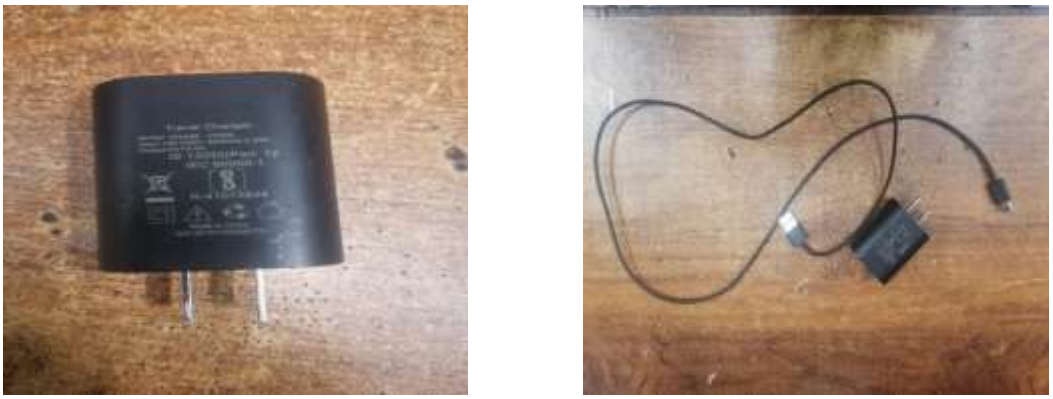

Figure 3.5: Raspberry Pi Charger

4. **Cameras:**

Camera type: We have used both types of cameras in our project, the one Raspberry Pi Camera Module which is designed specifically for the Raspberry Pi and similarly USB cameras connected to the USB ports of the Raspberry Pi 4GB.

Specifications: Specifications of the camera are as follows:

Image Resolution: Raspberry Pi Camera Module is 8MP where as the USB cameras are 5MP.

Connectivity: The Raspberry Pi High-Quality Camera uses a ribbon cable to connect to the Raspberry Pi where as the USB cameras are connected to the USB ports of the raspberry pi board.

Picture:

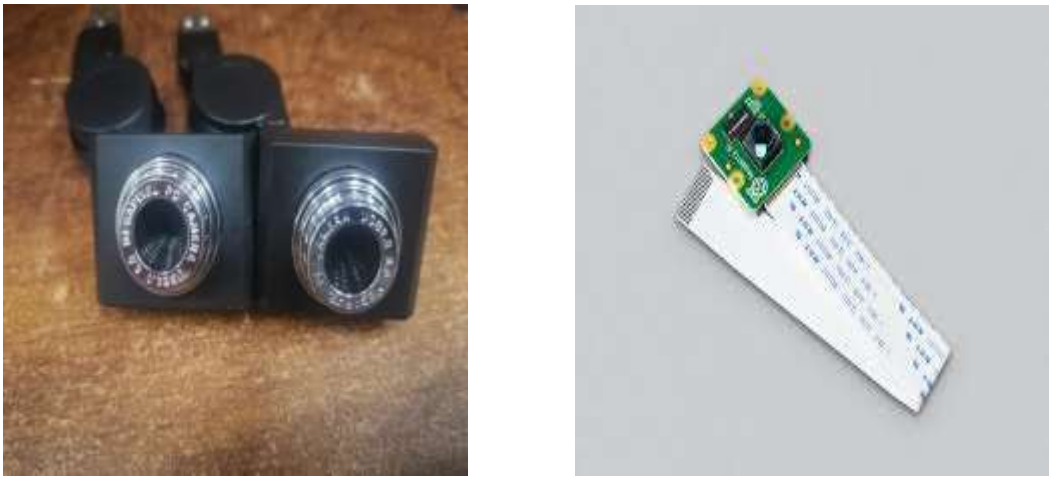

Figure 3.6: Raspberry Pi Camera

5. **Micro HDMI to VGA Converter:**

Connector type: The converter has a Micro HDMI input connector, which is the standard HDMI connector used on the Raspberry Pi 4GB.Also the converter provides a VGA output connector, allowing you to connect to VGA displays or projectors.

Functionality: The converter converts the digital HDMI signal from the Micro HDMI port to an analog VGA signal.

Picture:

Figure 3.7: Micro HDMI to VGA Converter

6. **Cars:**

Toys cars are being used in our project for density based traffic control system.

Figure 3.8: Toy Cars

7. SD Card:

In our project, the operating system needed to execute the Raspberry Pi software is stored on a 16GB SD card. The primary storage media is an SD card, which contains the data and software components required for the Raspberry Pi to operate.

Our hardware prototype consists of the following equipments:

Figure 3.9: Micro SD Card(16GB)

8. Hardware Prototype:

Our hardware prototype consists of the following equipments:

1. A wooden sheet of length (3x3) feet.
2. 4 pieces of black chart.
3. A pole at the corner of wooden sheet for holding cameras and traffic lights.
4. White tapes for pedestrian lane.

Picture:

Figure 3.10: Hardware Prototype

3.2.3 Software/Tools

Our system is being implemented and supported using a variety of software tools.

1. **Raspbian OS:**
   Raspberry Pi Imager is an official operating system for Raspberry Pi, based on Debian Linux. It offers a reliable and optimized environment for executing programmes.

2. **Python:**
   Python is being used in our project to support data processing and machine learning.
   Raspberry Pi Imager is an official operating system for Raspberry Pi, based on Debian Linux. It offers a reliable and optimized environment for executing programmes.

3. **Open CV:**
   Open CV is used in our project for image and video processing. It can be used for things like traffic flow analysis, license plate recognition, and vehicle identification.
   Python is being used in our project to support data processing and machine learning.

4. **Tensor Flow Lite:**
   A simplified variation of the Tensor Flow framework created especially for the Raspberry Pi, which has limited resources, is used to execute machine learning models. It is applicable to activities like object detection.

5. **SQ Lite:**
   A lightweight database management system that can be used for storing and managing traffic-related data on the Raspberry Pi. Smaller-scale applications can benefit from SQLite's ability to store and retrieve structured data.

Chapter 4
Testing and Validation/Discussion

In order to guarantee the dependability, correctness, and effectiveness of the established system, testing and validation are essential.

4.1 Testing

It is crucial to verify the veracity of the vehicle identification and infraction recognition algorithms. It entails contrasting the system's findings with real-world data, manually gathered information, or benchmark datasets that have already been created.

4.1.1 Prototypes

Prototype of our proposed project is as follows:

Figure 4.1: Complete Setup

4.2 Results/Output/Statistics

The "Automatic Traffic Flow Control and Violation Detection System's output and results give us important information about how well it functions.

Figure 4.2: Output

4.2.1 Accuracy

For a number of system functions, such as traffic flow control, infraction identification, and overall system performance, accurate vehicle detection is crucial. In this part, we talk about the vehicle detecting component's accuracy and give the test and assessment findings.

Figure4.3: Accuracy of Detected Cars

Chapter 5
Conclusion and Future Recommendations

The conclusion and recommendations part summarizes the whole report by highlighting all the chapters and their significance and the importance of the project and the achievements. The Recommendations are interlinked with the conclusion. The conclusion drawn from the project report can be further implemented in the recommendation section to overcome the constraints of the project.

5.1 Conclusion

In conclusion "Automatic Traffic Flow Control and Violation Detection System" offers a complete solution for effective traffic management and violation detection, to sum up. The system successfully handles the issues of traffic congestion and non-compliance with traffic laws by utilising high-quality cameras, cutting-edge image processing techniques, and sophisticated decision-making algorithms built on the Raspberry Pi controller.

The system assures accurate vehicle identification, emergency vehicle detection, and traffic law breaches through the sequential process of picture capture, preprocessing, segmentation, vehicle detection, and possibility checks. Traffic flow is optimised, congestion is reduced, and road efficiency is increased because to the system's capacity to dynamically change traffic signal timings based on vehicle density.

The system also includes means for enforcing violations by sending email-based e-challans to car owners, encouraging compliance and responsibility. The system's capabilities are further increased by the incorporation of several software tools and technologies, such as TensorFlow Lite for machine learning and OpenCV for image processing.

Overall, the "Automatic Traffic Flow Control and Violation Detection System" offers an intelligent and automated approach to traffic management, enhancing the enforcement of traffic laws while also improving road safety and traffic flow. The technology has tremendous promise for future deployment in real-world circumstances because of its ability to simplify traffic management and reduce human intervention, laying the groundwork for more dependable and efficient transportation networks.

5.2 Future Recommendations

There is always room for further enhancement and refinements. Future work recommendations as the extension of project are as follows:

- Including advanced sensor technologies will improve the system's ability to recognise and track moving objects.
- Introducing a Mobile Application for User Interaction.
- Introducing Cloud based Infrastructure.

References

[1] Li, C., Luo, H., Zhang, L., & Zhang, H. (2017). A real-time traffic flow estimation and control system based on computer vision. IEEE Access, 5, 2521-2531.

[2] Darwish, A., & Abdel-Rahman, E. (2017). Traffic light control using image processing. 2017 14th International Computer Engineering Conference (ICENCO). IEEE.

[3] Ukwandu, E., & Ogu, C. (2020). Smart traffic flow control system using Raspberry Pi and Internet of Things (IoT). International Journal of Electrical and Computer Engineering (IJECE), 10(3), 3177-3185.

[4] Mishra, A., et al. (2021). Raspberry Pi-based traffic flow control system using image processing. International Journal of Innovative Technology and Exploring Engineering (IJITEE), 10(4), 3392-3397.

[5] Singh, A., & Sharma, V. (2021). Traffic control system using Raspberry Pi. International Journal of Advanced Research in Engineering and Technology (IJARET), 12(3), 305-313.

[6] Patil, S., & Kharate, G. (2018). Automatic traffic density control using Raspberry Pi. International Journal of Innovative Research in Computer and Communication Engineering, 6(7), 5862-5866.

[7] Bappy, J. H., Rahman, M. A., & Nain, Z. (2019). Intelligent traffic control system using Raspberry Pi and image processing. 2019 4th International Conference on Electrical Information and Communication Technology (EICT). IEEE

[8] Gaur, S., & Singh, A. (2018). Traffic management system for smart city using Internet of Things. 2018 3rd International Conference on Computing, Communication, Control and Automation (ICCUBEA). IEEE.

[9] Al-Zubi, T. H., Al-Dubai, A. Y., Al-Raweshidy, H. S., & Al-Jarrah, O. Y. (2018). Raspberry Pi-based intelligent transportation system for real-time traffic monitoring and control. IET Intelligent Transport Systems, 12(4), 254-263.

[10] Mishra, S., & Kumar, S. (2017). Raspberry Pi-based automatic traffic light control system. 2017 7th International Conference on Cloud Computing, Data Science & Engineering-Confluence. IEEE.

[11] Kasim, S., Ali, S. A., & Zulkifli, A. A. (2020). Raspberry Pi-based smart traffic light control system for urban traffic management. Indonesian Journal of Electrical Engineering and Computer Science, 17(3), 1476-1484.

1.

Annexure ‘A’
Code

Python Code:

```
import os
import argparse
import cv2
import numpy as np
import sys
import glob
import importlib.util
import time
import serial
from serial.tools import list_ports
import os
import cv2
import time
import smtplib, ssl
from email.message import EmailMessage
import datetime
serial_en=False
email_flag=True

pre_vid=0
pre_vid_1=0
pre_vid_2=0
pre_vid_3=0

p_vid=0
p_vid_1=0
p_vid_2=0
p_vid_3=0

count_objects=0
vid = cv2.VideoCapture(0)
#vid_1 = cv2.VideoCapture(1)
vid_1=""
vid_2=""
vid_3=""
vid_4=""
##vid = cv2.VideoCapture(1)
##vid_1 = cv2.VideoCapture(2)
##vid_2 = cv2.VideoqCapture(3)
##vid_3 = cv2.VideoCapture(4)
dense_flag=False
frame_cnt=0
vid_en=True
vid_en_1=False
vid_en_2=False
vid_en_3=False
count_label=""
obj_count=0
```

```
obj_count_i=0
label_found=0
den_status=False
count_flag=False

den_status=True
obj_count_i=0
challan_flag=0
email_flag2=True
email_flag1=True



def send_email():
    port = 465  # For SSL
    smtp_server = "smtp.gmail.com"
    sender_email = "rajahaider99081@gmail.com"  # Enter your address
    receiver_email = "saleemqurashi32@gmail.com"  # Enter receiver address
    password = "yzdsgcqyciiydkhl"

    msg = EmailMessage()
    msg.set_content("Traffic Rule Violated \n pedestrian line crossing \n Vehicle NO:
ABC1234 \n Amount:1000 Pkr")
    msg['Subject'] = "E-Challan"
    msg['From'] = sender_email
    msg['To'] = receiver_email

    context = ssl.create_default_context()
    with smtplib.SMTP_SSL(smtp_server, port, context=context) as server:
        server.login(sender_email, password)
        server.send_message(msg, from_addr=sender_email, to_addrs=receiver_email)

ports = list(serial.tools.list_ports.comports())
for port in ports:
    port_c1=str(port)
    print(port_c1)
    if(port_c1.find("USB Serial")):
        port_c2=port_c1.split('-')[0]
        port_c3=port_c2[0:len(port_c2)-1]

        port_g = serial.Serial(port_c3,timeout=1, baudrate=115200)
        serial_en=True

        print(port_g)
        break


def write(x):
    port_g.write(bytes(x, 'utf-8'))
    time.sleep(0.05)
print(",,,,,,,,,,.....?????????")
print(serial_en)
print(",,,,,,,,,,,,,,,,")
```

```
serial_en=True
# Define and parse input arguments
parser = argparse.ArgumentParser()
parser.add_argument('--modeldir', help='Folder the .tflite file is located in',
                    required=True)
parser.add_argument('--graph', help='Name of the .tflite file, if different than detect.tflite',
                    default='detect.tflite')
parser.add_argument('--labels', help='Name of the labelmap file, if different than
labelmap.txt',
                    default='labelmap.txt')
parser.add_argument('--threshold', help='Minimum confidence threshold for displaying
detected objects',
                    default=0.50)
parser.add_argument('--image', help='Name of the single image to perform detection on. To
run detection on multiple images, use --imagedir',
                    default=None)
parser.add_argument('--imagedir', help='Name of the folder containing images to perform
detection on. Folder must contain only images.',
                    default=None)
parser.add_argument('--edgetpu', help='Use Coral Edge TPU Accelerator to speed up
detection',
                    action='store_true')

args = parser.parse_args()

MODEL_NAME = args.modeldir
GRAPH_NAME = args.graph
LABELMAP_NAME = args.labels
min_conf_threshold = float(args.threshold)
use_TPU = args.edgetpu

# Parse input image name and directory.
IM_NAME = args.image
IM_DIR = args.imagedir

# If both an image AND a folder are specified, throw an error
if (IM_NAME and IM_DIR):
    print('Error! Please only use the --image argument or the --imagedir argument, not both.
Issue "python TFLite_detection_image.py -h" for help.')
    sys.exit()

# If neither an image or a folder are specified, default to using 'test1.jpg' for image name
if (not IM_NAME and not IM_DIR):
    IM_NAME = '8.jpg'

# Import TensorFlow libraries
# If tflite_runtime is installed, import interpreter from tflite_runtime, else import from regular
tensorflow
# If using Coral Edge TPU, import the load_delegate library
pkg = importlib.util.find_spec('tflite_runtime')
if pkg:
    from tflite_runtime.interpreter import Interpreter
    if use_TPU:
```

```
        from tflite_runtime.interpreter import load_delegate
else:
    from tensorflow.lite.python.interpreter import Interpreter
    if use_TPU:
        from tensorflow.lite.python.interpreter import load_delegate

# If using Edge TPU, assign filename for Edge TPU model
if use_TPU:
    # If user has specified the name of the .tflite file, use that name, otherwise use default
'edgetpu.tflite'
    if (GRAPH_NAME == 'detect.tflite'):
        GRAPH_NAME = 'edgetpu.tflite'


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

# Define path to images and grab all image filenames
if IM_DIR:
    PATH_TO_IMAGES = os.path.join(CWD_PATH,IM_DIR)
    images = glob.glob(PATH_TO_IMAGES + '/*')

elif IM_NAME:
    PATH_TO_IMAGES = os.path.join(CWD_PATH,IM_NAME)
    images = glob.glob(PATH_TO_IMAGES)

# Path to .tflite file, which contains the model that is used for object detection
PATH_TO_CKPT = os.path.join(CWD_PATH,MODEL_NAME,GRAPH_NAME)

# Path to label map file
PATH_TO_LABELS = os.path.join(CWD_PATH,MODEL_NAME,LABELMAP_NAME)

# Load the label map
with open(PATH_TO_LABELS, 'r') as f:
    labels = [line.strip() for line in f.readlines()]

# Have to do a weird fix for label map if using the COCO "starter model" from
# https://www.tensorflow.org/lite/models/object_detection/overview
# First label is '???', which has to be removed.
if labels[0] == '???':
    del(labels[0])

# Load the Tensorflow Lite model.
# If using Edge TPU, use special load_delegate argument
if use_TPU:
    interpreter = Interpreter(model_path=PATH_TO_CKPT,
                              experimental_delegates=[load_delegate('libedgetpu.so.1.0')])
    print(PATH_TO_CKPT)
else:
    interpreter = Interpreter(model_path=PATH_TO_CKPT)

interpreter.allocate_tensors()
```

```
# Get model details
input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()
height = input_details[0]['shape'][1]
width = input_details[0]['shape'][2]

floating_model = (input_details[0]['dtype'] == np.float32)

input_mean = 127.5
input_std = 127.5

# Loop over every image and perform detection
while True:

    # Load image and resize to expected shape [1xHxWx3]
    #image = cv2.imread(image_path)
    try:
        frame_cnt+=1



        if(vid_en==True and frame_cnt>=50):
            frame_cnt=0
            vid_en=False
            vid_en_1=True
            vid_en_2=False
            vid_en_3=False
            den_status=False
            obj_count_i=0
            vid.release()
            time.sleep(.5)
            vid_1 = cv2.VideoCapture(2)

            time.sleep(.5)
        elif(vid_en==True and frame_cnt<50):

            ret, frame = vid.read()
            image=frame
            challan_flag=1
            print("cam_1")
        elif(vid_en_1==True and frame_cnt>=50):
            frame_cnt=0
            vid_en=True
            vid_en_1=False
            vid_en_2=False
            vid_en_3=False
            den_status=False
            obj_count_i=0
            vid_1.release()
            time.sleep(.5)
            vid = cv2.VideoCapture(0)

        elif(vid_en_1==True and frame_cnt<50):
```

```
            ret1, frame1 = vid_1.read()
            image=frame1
            challan_flag=2
            print("cam_2")

        # elif(vid_en_2==True and frame_cnt>=50):
            # frame_cnt=0
            # vid_en=True
            # vid_en_1=False
            # vid_en_2=False
            # vid_en_3=False
            # den_status=False
            # obj_count_i=0
            # vid_2.release()
            # time.sleep(.5)
            # vid = cv2.VideoCapture(1)

        # elif(vid_en_2==True and frame_cnt<50):

            # ret2, frame2 = vid_2.read()
            # image=frame2
            # print("cam_3")

        # elif(vid_en_3==True and frame_cnt>=50):
            # frame_cnt=0
            # vid_en=True
            # vid_en_1=False
            # vid_en_2=False
            # vid_en_3=False
            # den_status=False
            # obj_count_i=0
            # vid_3.release()
            # time.sleep(.5)
            # vid = cv2.VideoCapture(1)

        # elif(vid_en_3==True and frame_cnt<50):

            # ret3, frame3 = vid_3.read()
            # image=frame3
            # print("cam_4")

    except:
        pass

##    elif(vid_en_1==True and frame_cnt>=5):
##        frame_cnt=0
##        vid_en=False
##        vid_en_1=False
##        vid_en_2=True
##        vid_en_3=False
##        vid1.release()
##        time.sleep(2)
```

```
##        vid_2 = cv2.VideoCapture(3)
##    elif(vid_en_2==True and frame_cnt>=5):
##        frame_cnt=0
##        vid_en=False
##        vid_en_1=False
##        vid_en_2=False
##        vid_en_3=True
##        vid2.release()
##        time.sleep(2)
##        vid_3 = cv2.VideoCapture(4)
##    elif(vid_en_3==True and frame_cnt>=5):
##        frame_cnt=0
##        vid_en=True
##        vid_en_1=False
##        vid_en_2=False
##        vid_en_3=False
##        vid3.release()
##        time.sleep(2)
##        vid = cv2.VideoCapture(1)
##    print(vid_en)
##    print(vid_en_1)
##    print(vid_en_2)
##    print(vid_en_3)
##    print(".....")
##    if(vid_en==True):
##        ret, frame = vid.read()
##        image=frame
##    elif(vid_en_1==True):
##        ret_1, frame_1 = vid_1.read()
##        image=frame_1
##    elif(vid_en_2==True):
##        ret_2, frame_2 = vid_2.read()
##        image=frame_2
##    elif(vid_en_3==True):
##        ret_3, frame_3 = vid_3.read()
##        image=frame_3


    try:
        image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
        imH, imW, _ = image.shape
        image_resized = cv2.resize(image_rgb, (width, height))
        input_data = np.expand_dims(image_resized, axis=0)

        # Normalize pixel values if using a floating model (i.e. if model is non-quantized)
        if floating_model:
            input_data = (np.float32(input_data) - input_mean) / input_std

        # Perform the actual detection by running the model with the image as input
        interpreter.set_tensor(input_details[0]['index'],input_data)
        interpreter.invoke()

        # Retrieve detection results
```

```
        boxes = interpreter.get_tensor(output_details[0]['index'])[0] # Bounding box coordinates
of detected objects
        classes = interpreter.get_tensor(output_details[1]['index'])[0] # Class index of detected
objects
        scores = interpreter.get_tensor(output_details[2]['index'])[0] # Confidence of detected
objects
        #num = interpreter.get_tensor(output_details[3]['index'])[0]  # Total number of detected
objects (inaccurate and not needed)
    ##    print(type(boxes))
        # Loop over all detections and draw detection box if confidence is above minimum
threshold
        for i in range(len(scores)):
            if ((scores[i] > min_conf_threshold) and (scores[i] <= 1.0) and
labels[int(classes[i])]!='dining table'):
                count_objects+=1
                print(labels[int(classes[i])])
                ymin = int(max(1,(boxes[i][0] * imH)))
                xmin = int(max(1,(boxes[i][1] * imW)))
                ymax = int(min(imH,(boxes[i][2] * imH)))
                xmax = int(min(imW,(boxes[i][3] * imW)))

                cv2.rectangle(image, (xmin,ymin), (xmax,ymax), (10, 255, 0), 2)

                # Draw label
                object_name = labels[int(classes[i])] # Look up object name from "labels" array
using class index
                #object_name='car'

                label = '%s: %d%%' % (object_name, int(scores[i]*100)) # Example: 'person: 72%'
                labelSize, baseLine = cv2.getTextSize(label, cv2.FONT_HERSHEY_SIMPLEX,
0.7, 2) # Get font size
                label_ymin = max(ymin, labelSize[1] + 10) # Make sure not to draw label too close
to top of window
                cv2.rectangle(image, (xmin, label_ymin-labelSize[1]-10), (xmin+labelSize[0],
label_ymin+baseLine-10), (255, 255, 255), cv2.FILLED) # Draw white box to put label text
in
                cv2.putText(image, label, (xmin, label_ymin-7),
cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 0), 2) # Draw label text


                x_c1=(abs(xmax-xmin)/2)
                x_c2=xmin+x_c1
                y_c1=(abs(ymax-ymin)/2)
                y_c2=ymin+y_c1
                cv2.putText(image, str(x_c2)+' '+ str(y_c2), (100, 100),
cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255, 0), 2) # Draw label text
                print()
                if(challan_flag==1 and y_c2>200 and email_flag1==True):
                    print('challan 1 sending')
                    send_email();
                    email_flag1=False
                    print('challan  1 sending')
                elif(challan_flag==2 and y_c2<150 and email_flag2==True):
```

```
            print('challan 2 sending')
            send_email();
            email_flag2=False
            print('challan 2 sending')


        #print("("+str(xmin),str(xmax)+")")
        #print("("+str(ymin),str(ymax)+")")
        #print("("+str(x_c2)+","+str(y_c2)+")"+"  "+str(i))
# All the results have been drawn on the image, now display the image
if(count_objects>2):

    obj_count_i+=1
    print(obj_count_i)
    print(count_objects)
    print("*****************************************8")
elif(count_objects<=2):
    obj_count_i-=1

if(obj_count_i>10):
    dense_flag=True
    count_label="Status: "+"Dense"
    den_status=True
    obj_count_i=0
elif(obj_count_i<0):
    dense_flag=False
    count_label="Status: "+"Normal"
    obj_count_i=0
    den_status=False

if(dense_flag==True and vid_en==True):
    p_vid=1
    count_label=count_label+"   "+"Signal_1"
    if((serial_en==True) and (pre_vid!=p_vid)):
        #port_g.write(b"#1#\n")
        write(str(3))
        print("send_1")
        time.sleep(1)


elif(dense_flag==False and vid_en==True):
    p_vid=0
     #if((serial_en==True) and (pre_vid!=p_vid)):

        # port_g.write(b"#5#\n")
        # time.sleep(1)
        # print("send_5")


if(dense_flag==True and vid_en_1==True):
     p_vid_1=1
     count_label=count_label+"   "+"Signal_2"
     if((serial_en==True) and (pre_vid_1!=p_vid_1)):
```

```
            write(str(1))
            time.sleep(1)
            print("send_2")
        elif(dense_flag==False and vid_en_1==True):
            p_vid_1=0
            # if((serial_en==True) and (pre_vid_1!=p_vid_1)):
                # port_g.write(b"#6#\n")
                # time.sleep(1)
                # print("send_6")

        # if(dense_flag==True and vid_en_2==True):
            # p_vid_2=1
            # count_label=count_label+"   "+"Signal_3"
            # if((serial_en==True) and (pre_vid_2!=p_vid_2)):
                # port_g.write(b"#3#\n")
                # print("send_3")
                # time.sleep(1)
        #elif(dense_flag==False and vid_en_2==True):
        #    p_vid_2=0
         #   if((serial_en==True) and (pre_vid_2!=p_vid_2)):
          #      port_g.write(b"#7#\n")
           #     print("send_7")
            #    time.sleep(1)

##        if(dense_flag==True and vid_en_3==True):
##            count_label=count_label+"   "+"Signal_4"
##            if(serial_en==True):
####                port_g.write(b"#4#\n")
##              time.sleep(.2)
##        elif(dense_flag==False and vid_en_3==True):
##            if(serial_en==True):
####                port_g.write(b"#8#\n")
##              time.sleep(.2)

        pre_vid=p_vid
        pre_vid_1=p_vid_1
        #pre_vid_2=p_vid_2

        cv2.putText(image, count_label, (0, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0,
255, 0), 2) # Draw label text

        cv2.imshow('Object detector', image)
        count_objects=0
        cv2.imwrite('output\8.jpg',image)
        print(serial_en)
        print('flag_enable')
        #time.sleep(1)
    # Press any key to continue to next image, or press 'q' to quit

    except:
        pass
    if cv2.waitKey(1) == ord('q'):
        vid.release()
```

```
        vid_1.release()
        #vid_2.release()
        break

# Clean up
cv2.destroyAllWindows()
```