VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"Jnana Sangama", Belagavi-590018, Karnataka

A Project Report on

"NUMBER PLATE RECOGNITION (NPR) USING IMAGE PROCESSING TECHNOLOGY"

Submitted in fulfilment for the requirements of VIII semester degree of

BACHELOR OF ENGINEERING

IN

INFORMATION SCIENCE AND ENGINEERING

By

Abhishek Chaudhary (1DB19IS003)

Akhilesh (1DB19IS004)

Hasan Raihan (1DB19IS030)

Neel Mitesh Patel (1DB19IS054)

Under the Guidance of

Mrs. Roopashree M S

Assistant Professor

Dept. of ISE, DBIT

Department of Information Science and Engineering

DON BOSCO INSTITUTE OF TECHNOLOGY

Kumbalagodu, Mysuru Road, Bengaluru-560074

2022-2023

DON BOSCO INSTITUTE OF TECHNOLOGY

Kumbalagodu, Mysuru Road, Bengaluru-560074

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

CERTIFICATE

It is Certified that the Project on the topic **“NUMBER PLATE RECOGNITION (NPR) USING IMAGE PROCESSING TECHNOLOGY”** has been successfully presented at Don Bosco Institute of Technology by **Abhishek Chaudhary(1DB19IS003), Akhilesh (1DB19SI004), Hasan Raihan (1DB19IS030)** and **Neel Mitesh Patel (1DB19IS054)**, in partial fulfillment of the requirements for the VIII semester degree of **Bachelor of Engineering in Information Science and Engineering** of Visvesvaraya Technological University, Belagavi during academic year 2022-2023. It is certified that all corrections and suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The Project report has been approved as it satisfies the academic requirements in respect of Project work for the said degree.

Signature of Guide	**Signature of HoD**	**Signature of Principal**
……………………….	……………………….	………………………..
Mrs. Roopashree M S	**Dr. B. K Raghavendra**	**Dr. Naghabhushana B S**
Assistant professor,	Professor & Head,	Principal,
Dept. of ISE,	Dept. of ISE,	DBIT, Bengaluru.
DBIT, Bengaluru.	DBIT, Bengaluru.	

Name of Externals	**Signature with Date**
1……………………….	……………………….
2……………………….	……………………….

DON BOSCO INSTITUTE OF TECHNOLOGY

Kumbalagodu, Mysuru Road, Bengaluru-560074

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

DECLARATION

We, **Abhishek Chaudhary, Akhilesh, Hasan Raihan and Neel Mitesh Patel** students of eighth semester B.E, Department of Information Science and Engineering, Don Bosco Institute of Technology, Kumbalagodu, Bengaluru, declare that the project work entitled **"NUMBER PLATE RECOGNITION (NPR) USING IMAGE PROCESSING"** has been carried out by us and submitted in partial fulfillment of the course requirements for the award of degree in **Bachelor of Engineering** in **Information Science and Engineering** of **Visvesvaraya Technological University, Belagavi** during the academic year **2022-2023**. The matter embodied in this report has not been submitted to any other university or institution for the award of any other degree or diploma.

Place: Bengaluru

Date:

Abhishek Chaudhary (1DB19IS003)

Akhilesh (1DB19IS004)

Hasan Raihan (1DB19IS030)

Neel Mitesh Patel (1DB19IS054)

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crowned the efforts with success.

We would like to profoundly thank **Management** of **Don Bosco Institute of Technology** for providing such a healthy environment for the successful completion of Project work.

We would like to express our thanks to the Principal **Dr. Naghabhushana B S** for his encouragement that motivated us for the successful completion of Project work.

It gives us immense pleasure to thank **Dr. B K Raghavendra**, Professor & Head of Department for his constant support and encouragement.

Also, we would like to express our deepest sense of gratitude to our guide **Mrs. Roopashree M S**, Assistant Professor, Department of Information Science & Engineering for her constant support and guidance throughout the Project work.

We would like to thank our parents, the Laboratory System Administrators in the Department of Information Science & Engineering and all other teaching and non-teaching staff of Information Science Department who have directly or indirectly helped me in the completion of our Project work.

Abhishek Chaudhary(1DB19IS003)

Akhilesh (1DB19IS004)

Hasan Raihan(1DB19IS030)

Neel Mitesh Patel(1DB19IS054)

ABSTRACT

In a metropolis full of traffic recognizing license plates has become a very laborious task. The proposed project deals with a Concept capable of automatically tracking license plates and verifying their legitimacy. This project will use the latest YOLO object detection algorithm and deep learning model. A dedicated camera will be installed at a junction to record real-time images of passing vehicles. This will be given as input to the object detection algorithm, extracting the bounding box from the license plate and raw attributes such as vehicle type and color the optical character recognition system filters the numbers and the characters that appear on the license plate. This data is inserted into a local database along with other metadata such as where the photo was taken and the timestamp. This database is migrated to the cloud, where the Map Reduce function runs, comparing it to a local database created elsewhere to identify discrepancies to be found, and to identify temporary discrepancies specific to duplicate license plates. It also does type-checking, cross-validation, and other things. Once these vehicles are located, they are tracked by road-mounted traffic cameras and sent to local authorities who can take further appropriate action related to the violation.

LIST OF FIGURES

CONTENT

CHAPTER 1

INTRODUCTION

1.1 Introduction

The social scenario in India is significantly different due to problems such as poverty, unemployment as well as a considerably lower respect for rules. This makes it unfeasible to go for a completely automatic toll booth. The industry requires an automatic vehicle classification system in India not to reduce or eliminate human intervention or labour, but to ensure that human intervention does not cause any financial malpractices. The industry requires a system that runs in the background and merely keeps a cross-check on the manual.

The traditional OCR based approach for number plate recognition does not work for the variations in painting style of the number plates. In this paper authors have presented an image retrieval-based method to recognize the car number plate captured using a smart phone to facilitate the Car management system of a Smart office premise. In the proposed method a smart phone is used to capture the images and extract features of the car number plate. These features are matched against predefined set of same car number plate images in the database. The character images are matched in an efficient manner to make it a real time solution.

As already stated, the system using fibre optics inherently possesses a large number of problems apart from the main concerns of high cost and maintenance. Although an IR curtain system reduces the cost significantly, it is still quite expensive and cheaper alternatives are desired. As almost all the toll booths employ cameras for security purposes, it was felt that the feasibility of a system using IP cameras should be tested.

With respect to vehicle safety, India meets only two out of the seven vehicle safety standards by the World Health Organization (WHO). The main cause of these fatalities is people riding two wheelers under the influence of alcohol results and violation of traffic rules which later on results in serious accidents.

1.2 Problem Background

1.2.1 What is the Problem?

The challenge at hand is to develop an automated vehicle classification system in India that can curb financial malpractices. The current manual system is rife with corruption and revenue loss, necessitating an overhaul. The conventional OCR-based approach falls short in India due to the myriad styles of number plate painting. Therefore, a system that can operate in stealth mode and keep a cross-check on the manual process is essential. Additionally, vehicle safety is a pressing issue in India, with non-compliance with global safety standards and traffic regulations causing fatalities.

1.2.2 Why is it a Problem?

The challenge of financial malpractices at toll booths is a significant issue in India. The manual system is plagued with corruption, with toll booth employees colluding with drivers to evade charges. This has resulted in massive revenue loss, undermining infrastructure growth and maintenance. An automated vehicle classification system is necessary to ensure that human intervention doesn't lead to financial malpractices. The system will operate in stealth mode, providing a manual process check to ensure correct toll charges for all vehicles passing through.

The conventional OCR-based approach for number plate recognition falls flat in India due to the variety of number plate painting styles. This leads to misidentification and inaccurate toll charges. The suggested image retrieval-based method that employs a smartphone to capture images and extract number plate features is a potential solution. However, the system's efficacy and efficiency must be thoroughly tested before widespread implementation.

Vehicle safety is a significant concern in India, as the country meets only two out of seven vehicle safety standards set by the WHO. This is primarily due to the lack of compliance with global safety standards and traffic rules, resulting in fatalities. People riding two-wheelers under the influence of alcohol and flouting traffic rules are the leading cause of these fatalities. The lack of traffic rule enforcement exacerbates the issue, making road safety for all a daunting challenge.

1.2.3 Where is the Problem Now?

The problem of financial malpractices at toll booths continues to plague India, with the manual system still in use. This makes it susceptible to corruption and revenue loss. Although the need for an automated vehicle classification system has been identified, there is a lack of large-scale implementation. The proposed image retrieval-based method, using a smartphone to capture number plate features, is a potential solution. However, thorough testing is required to gauge its efficiency and efficacy.

Regarding vehicle safety, India's compliance with WHO safety standards is severely lacking, leading to fatalities. The lack of compliance with global safety standards and traffic regulations exacerbates the issue, making it a challenge to ensure road safety for all. A comprehensive approach that includes traffic rule enforcement, compliance with global safety standards, and greater public awareness of road safety is necessary to address the problem.

1.3 Aim and Objectives

1.3.1 Aim

The aim of this project is to develop a system for automatic number plate detection and recognition that can accurately identify and track vehicles in real-time.

1.3.2 Objectives

- To design and develop an image processing algorithm that can accurately detect and extract number plates from images captured by surveillance cameras.
- To train a machine learning model to recognize characters on the number plates and convert them into text format.
- To integrate the number plate recognition system with a database of registered vehicles and generate alerts for any violations such as expired registration or stolen vehicles.
- To implement additional features such as bike and people detection, emission tests, and fake license plate detection to improve the system's accuracy and effectiveness.

CHAPTER 2

LITERATURE SURVEY

2.1 Existing System

The high-performance fiber optic sensors are used for detection of moving vehicles. A typical installation consists of an interface device with transmitter (LED), receiver (photo detector), and light guide connection cable (feeder) and fiber optic sensor. As the vehicle passes over the sensors there is a change in the signal levels obtained from the sensors. The output signals from the fiber optic sensors are fed into a signal processing and data evaluation unit which comprises of the algorithm, which computes axle count, axle spacing, vehicle lengths and vehicle classes based on time, distance formula, and amount of micro bending. An IR curtain basically consists of an infra-red transmitter and receiver. These curtains create a clear profile of the vehicle as it passes through it. However the entire profile of the vehicle cannot be obtained by using just one strip of IR curtain due to varying speed of the vehicle that passes the gate. Thus it is important to know the speed of the vehicle. Using the distance between the curtains and the time, we calculate the speed of the vehicle. With the speed of the vehicle known and the frequency of pulses known we can determine the correct profile of the vehicle.

2.2 Proposed System

This section presents the proposed approach for real-time detection of bike-riders without helmet which works in two phases. In the first phase, we detect a bike-rider in the video frame. In the second phase, we locate the head of the bike-rider and detect the number plate and also detect whether the rider is using a helmet or not. In order to reduce false predictions, we consolidate the results from consecutive frames for final prediction. The block diagram shows the various steps of proposed framework such as background subtraction, feature extraction, object classification using sample frames. As helmet is relevant only in case of moving bike-riders, so processing full frame becomes computational overhead which does not add any value to detection rate. In order to proceed further, we apply background subtraction on gray-scale frames, with an intention to distinguish between moving and static objects. Next, we present steps involved in background modeling.

2.3 Detailed Description

[1] Traffic Rules Violation Detection using Machine Learning Techniques
Authors: P. Srinivas Reddy, T. Nishwa, R. Shiva Kiran Reddy, Ch Sadviq , K. Rithvik
Published in: IEEE 2021

An automatic number plate recognition (ANPR) system is a key aspect in traffic congestion. This will help to minimize the different kind of violations on the road. Advanced systems for tracking and identifying stolen, unauthorized vehicles are based on automated number plate recognition technology. This paper's main objectives are to review other methods and propose our own algorithm. A short review is performed on the various methods of number plate recognition algorithms. Further explanations of the proposed algorithm are illustrated in graphical forms to show how the algorithm works. This paper concluded with tests and evaluation results.

[2] An Efficient FPGA Implementation of Optical Character Recognition for License Plate Recognition
Authors: Yuan Jing, Bahar Youssefi, Mitra Mirhassani
Published in: 2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE)

Optical Character Recognition system (OCR) can be used in intelligent transportation systems for license plate detection. However, most times the systems are unable to work with noisy and imperfect images. In this work, a robust FPGA based OCR system has been designed and tested with imperfect and noisy license plate images. The OCR system is based on a feedforward neural networks, which uses an efficient and precise neuron. The neuron transfer function is based on an approximation of the Hyperbolic Tangent Activation Function.

[3] Autonomous Vehicle And Real Time Road Lanes Detection And Tracking
Authors: Farid Bounini, Denis Gingras, Vincent Lapointe, Herve Pollart
Published in: 2015 IEEE

Advanced Driving Assistant Systems, intelligent and autonomous vehicles are promising solutions to enhance road safety, traffic issues and passengers' comfort. Such applications require advanced computer vision algorithms that demandpowerful computers with high-speed processing capabilities. Keeping intelligent vehicles on the road until its destination, in some cases, remains a great challenge, particularly when driving at high speeds. The first principle task is robust navigation, which is often based on system vision to acquire RGB images of the road for more advanced processing. The second task is the vehicle's dynamic controller according to its position, speed and direction. This paper presents an accurate and efficient road boundaries and painted lines' detection algorithm for intelligent and autonomous vehicle. It combines Hough Transform to initialize the algorithm at each time needed, and Canny edges' detector, least-square method and Kalman filter to minimize the adaptive region of interest, redict the future road boundaries' location and lines parameters. The scenarios are simulated on the Pro-SiVIC simulator provided by Civitec, which is a realistic simulator of vehicles' dynamics, road infrastructures, and sensors behaviors, and OPAL-RT product dedicated for real time processing and parallel computing.

[4] Vehicle Number Plate Detection and Recognition using Bounding Box Method

Authors: Mahesh Babu K, M V Raghunadh

Published in: 2016 International Conference on Advanced Communication Control and Computing Technologies (ICACCCT)

The use of vehicles in our life is increasing exponentially day by day and as increasing vehicles are violating the traffic rules, theft of vehicles, entering in restricted areas, high number of accidents lead to increase in the crime rates linearly. For any vehicle to be recognized, vehicle license plate detection will play a major significant role in this active world. For finding vehicles commonly used in field of security and safety system, LPDR plays a significant role and we need to recognize vehicles registration number at a certain distance.

This paper has four major steps as follows: Preprocessing of captured image, Extracting license number plate region, Segmentation and Character Recognition of license plate. In pre-processing the desired vehicle image is taken through the digital camera, brightness of image is adjusted, noise removal using filters and image is converted to gray scale. Exactions of license plate region consist of finding the edges in the image where exact location of licenses plate is located and crop it into rectangular frame. Segmentation plays a vital role in vehicle licenses plate recognition; the legibility of character recognition completely relies on the segmentation done. The approach which we have used is simple but appropriate. First we segmented all characters in the image (LP) using Bounding box method. Finally, recognition of each character is done. The template matching method is used for recognition each character in the vehicle license plate.

[5] Dynamic Traffic Rule Violation Monitoring System Using Automatic Number Plate Recognition with SMS Feedback

Authors: R Shreyas, Pradeep Kumar B V, Adithya H B, Padmaja B

Published in: 2017 2nd International Conference on Telecommunication and Networks (TEL-NET 2017)

In the last couple of decades, number of vehicles has been increased drastically. Hence it has become very difficult to keep track of each and every vehicle for the purpose of traffic management and the law enforcement. Use of Automation number Plate Recognition is increasingly now days for maintaining traffic activities and as similar to the method of automatic electronic toll collection. In the past, from the survey many techniques and algorithms have been proposed for number plate detection and recognition, each technique having its own advantages and disadvantages. The fundamental step in number plate detection is localization of a number plate. The proposed Automatic Number Plate Recognition (ANPR) System is based on an image processing technology. The proposed system can be mainly used to monitor road traffic activities such as the identification of vehicle during traffic violations such as speed of vehicle and to detect at the street traffic signals lane violation. And thereby can be traced every vehicle for traffic rule violation and can provide the information to the concern authority to take further effective action, so we can have smooth traffic flow and also we can avoid accidents occurring on the traffic junction.

This system can also be used to assist the authorities in identifying for any stolen vehicle. The proposed system first detects for any vehicle which violates traffic rule and then captures the vehicle image. From the captured image using image segmentation technique the vehicle number plate region will be extracted. And the technique used for the character recognition on number plate is Optical character recognition. The system is implemented and simulated using Matlab. Next we have feedback system i.e., the vehicle number which is extracted using the proposed ANPR algorithm from PC which is then given to GSM modem for further SMS feedback system to the user and concern authority. The system design also involves the design and development of GUI using Matlab, to ease the user in step by step recognizing the characters and numbers from the vehicle license plate and displaying on the desktop GUI screen.

[6] Image Recognition for Automatic Number Plate Surveillance

Authors: P.Meghana, S. SagarImambi, P. Sivateja, K. Sairam

Published in: International Journal of Innovative Technology and Exploring Engineering (IJITEE)

ISSN: 2278-3075, Volume-8 Issue-4, February 2019

Automatic number plate recognition is a well known proposal in todays world due to the rapid growth of cars, bikes and other vehicles. This automatic number plate recognition system uses image processing technology for identification of the vehicles. This system can be used in highly populated areas and higly restricted areas to easily identify traffic rule violated vehicles and owners name, address and other information can be retrieved using this system. This system can be automated and it is used to recognize vehicles without authorization, vehicles that violated rules at populated areas like malls, universities, hospitals and other car parking lots. This can also be used in the case of car usage in terrorist activities, smuggling, invalid number plates, stolen cars and other illegal activities. It can also be used in highway electronic toll collection. Image of the car number plate is captured and detection is done by image processing, character segmentation which locate the alpha numeric characters on a number plate. Then the segmented characters are translated into text entries using optical character recognition (OCR). ANPR systems are already available but efficiency is not gained thoroughly. These systems are developed using different methodologies but some factors like vehicle speed, different font styles, font sizes, language of vehicle number and light conditions are required to be explored. These can affect a lot in the overall recognition rate. ANPR systems use (OCR) optical character recognition to scan the vehicle number plates, and it can be retrieved whenever required. The other details of the owners of the vehicles like address and mobile number can be manipulated whenever necessary by contacting the system administrative. The purpose of this paper is to recognize a car number plate using ANN, image segmentation. We intended to develop a system in mat lab which can perform detection as well as recognition of a car number plate.

CHAPTER 3

REQUIREMENT SPECIFICATION

3.1 Purpose

The project aims to provide total safety for bike riders. Recently helmets have been made compulsory, but still people drive without helmets. Bengaluru has approximately 50 lakh two-wheeler riders, which includes 500-600 accidents every year out of which 300-400 are fatal. Bengaluru ranks first in the city when it comes to two wheelers riders. In the last few years, there has been rapid increase in number of road accidents. Due to rise in road accidents, it has now become necessary to generate a system to limit accidental deaths. To keep a tab on the operators some tollbooths employ a system using fibre optic sensors to automatically classify a vehicle in the background and tally the results with the manual entries. However this system is expensive complicated and requires high maintenance. We aim to study the various systems that can be used to replace such a system with a cheaper and efficient alternative.

3.2 Scope

License Plate recognition is one of the techniques used for vehicle identification purposes. The sole intention of this project is to find the most efficient way to recognize the registration information from the digital image (obtained from the camera). This process usually comprises of three steps. First step is the license plate localization, regardless of the license-plate size and orientation. The second step is the segmentation of the characters and last step is the recognition of the characters from the license plate. Thus, this project uncovers the fundamental idea of various algorithms required to accomplish character recognition from the license plate during Template Matching. This system can also be used in highly populated areas and highly restricted areas to easily identify traffic rule violated vehicles and owner’s name, address and other information can be retrieved using this system. This system can be automated and it is used to recognize vehicles without authorization, vehicles that violated rules at populated areas like malls, universities, hospitals and other car parking lots. This can also be used in the case of car usage in terrorist activites, smuggling, invalid number plates, stolen cars and other illegal activities. Because of this wearing helmet is mandatory as per traffic rules, violation of which attract hefty fines.

Inspite, a large number of motorcyclists do not obey the rule. Presently, all major cities already deployed large video surveillance network to keep a vigil on a wide variety of threats. Thus using such already existing system will be a cost efficient solution, however these systems involve a large number of humans whose performance is not sustainable for long periods of time. Recent studies have shown that human surveillance proves ineffective, as the duration of monitoring of videos increases, the errors made by humans also increases.

3.3 General Description

In the case of license plate extraction, Hough transform was used to extract the license plate by using storing the horizontal and vertical edge information. But the disadvantage is that, this method requires huge memory and is computationally expensive. Various segmentation techniques were presented in the segmentation stage. Then the literature for recognition of characters using various approaches was also discussed. Lastly, some of the number plate recognition systems which have been developed commercially were presented.

Vehicle license plate recognition is one form of automatic vehicle identification system. LPR systems are of considerable interest, because of their potential applications to areas such as highway electronic toll collection, automatic parking attendant, petrol station forecourt surveillance, speed limit enforcement, security, customer identification enabling personalized services, etc.

Real time LPR plays a major role in automatic monitoring of traffic rules and maintaining law enforcement on public roads. This area is challenging because it requires an integration of many computer vision problem solvers, which include Object Detection and Character Recognition. The automatic identification of vehicles by the contents of their license plates is important in private transport applications. Number plate recognition basically consists of the following steps:

1. Number Plate Extraction
2. Detection of Helmet & Triple Riding
3. Detection of over speeding
4. Character Segmentation
5. Template Matching

Figure 3.3 shows the flow diagram for number plate extraction.

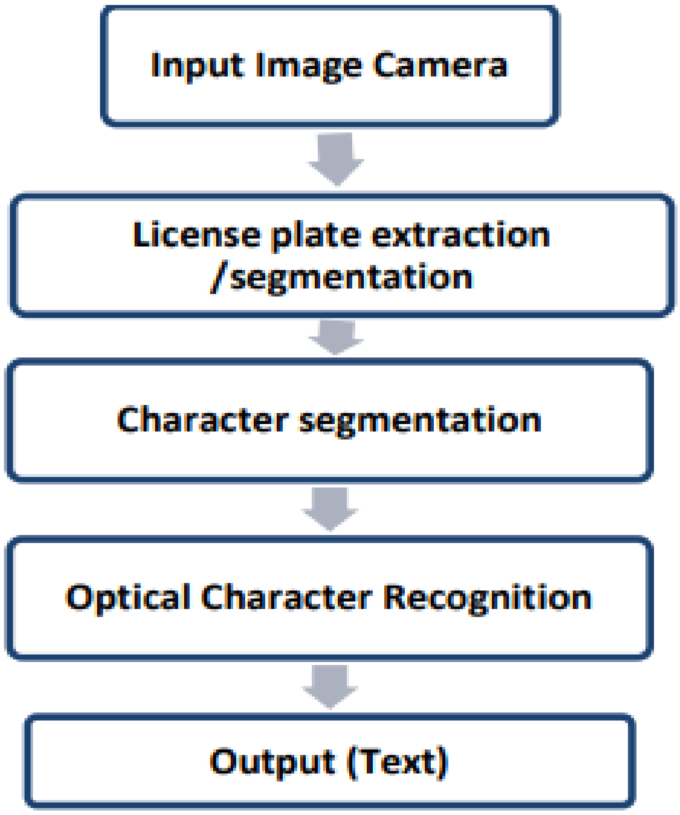

Input Image Camera
License plate extraction /segmentation
Character segmentation
Optical Character Recognition
Output (Text)

Figure 3.3 Flow diagram for Number Plate Extraction

3.4 Functional Requirements

Background Modeling and Moving Object Detection:

First, we apply background subtraction method to separate moving objects such as motorcycle, humans, cars from traffic videos using improved adaptive Gaussian mixture model in which is robust to certain challenges like illumination variance over the day, shadows, shaking tree branches and other sudden changes. We use variable number of Gaussian models for each pixel because single Gaussian is not sufficient to completely model these variations in complex and variable situations.

Convolutional Neural Network for Object Classification:

A convolutional neural network (CNN) is a variant of feed forward neural networks using back propagation algorithm. It learns high-level features from the spatial data like image. The recent widespread success of convolutional neural networks is in it’s ability to extract inter-dependent information from the images i.e localization of the pixels which are highly sensitive to other pixels. The convolutional neural network training consist of convolution layers, rely layers max pooling layers, fully connected layers and a loss function (e.g. SVM/SoftMax) on the last (fully-connected) layer. In the primary layers we get the edge information of the images similar to some of the handcrafted algorithms but, In the final layers, we start getting texture and ridge information which helps us in getting sensitive information useful for classification.

Recognition of Motorcyclists from Moving Objects:

To find bounding boxes of different objects, we used Gaussian background subtraction which uses a method to model each background pixel by a mixture of K Gaussian distributions (K = 3 to 5). The probable background colours are the ones which stay longer and are more static. On these varying pixels, we draw a rectangular bounding box. After obtaining all the objects of motorcyclists and non-motorcyclists, a CNN model is built using these images to separate the motorcyclists from other moving objects. Fig. 2 show the feature maps of the sample motorcycles. These feature maps illustrate that the CNN learns the common hidden structures among the motorcyclist in the training set and thus able to distinguish between a motorcyclist and other objects.

Recognition of Motorcyclists without Helmet:

To recognize motorcyclists without helmet, from the images of motorcyclists, we cropped only the top one fourth part of the image as that was the region where the motorcyclist’s head is located most of the time.

3.5 Modules and Frameworks

Object Detection:

Object Detection is the process of finding real-world object instances like car, bike, TV, flowers, and humans in still images or Videos. It allows for the recognition, localization, and detection of multiple objects within an image which provides us with a much better understanding of an image as a whole. It is commonly used in applications such as image retrieval, security, surveillance, and advanced driver assistance systems (ADAS).

Object Detection can be done via multiple ways:

- Feature-Based Object Detection
- Viola Jones Object Detection
- SVM Classifications with HOG Features
- Deep Learning Object Detection

In this Object Detection Tutorial, we'll focus on **Deep Learning Object Detection** as Tensorflow uses Deep Learning for computation.

OpenCV:

OpenCV is a computer vision library originally developed by Intel and now supported by Willow Garage. It is free for use under the open source BSD license. The library is cross-platform. It focuses mainly on real-time image processing. If the library finds Intel's Integrated Performance Primitives on the system, it will use these commercial optimized routines to accelerate it. OpenCV is NOT a piece of software that you run and process images. You need to write code. It is one superb IDE. You need to download the Visual C++ 2010 Express. Also, OpenCV is not some executable file that you double click and it'll start working. When you write your own code, you "link" to these library files to access the OpenCV functions. The Figure 3.5.1 shows the logo of the OpenCV library. The Figure 3.5.1 shows the logo of OpenCV Library.

Why OpenCV?

There are a couple of why to prefer OpenCV over Matlab. Specific OpenCV was made for image processing. Each function and data structure was designed with the Image Processing coder in mind. Matlab, on the other hand, is quite generic. You get almost anything in the world in the form of toolboxes. All the way from financial tool boxes to highly specialized DNA tool boxes.

Speedy:

Matlab is just way too slow. Matlab itself is built upon Java. And Java is built upon C. So when you run a Matlab program, your computer is busy trying to interpret all that Matlab code. Then it turns it into Java, and then finally executes the code. If you use C/C++ you don't waste all that time. You directly provide machine language code to the computer, and it gets executed. So ultimately you get more image processing, and not more interpreting.

Sure you pay the price for speed – a more cryptic language to deal with, but it's definitely worth it. You can do a lot more. You could do some really complex mathematics on images with C and still get away with good enough speeds for your application.

Efficient:

Matlab uses just way too much system resources. With OpenCV, you can get away with as little as 10mb RAM for a realtime application. But with today's computers, the RAM factor isn't a big thing to be worried about. You do need to take care about memory leaks, but it isn't that difficult. You can read the article about various Memory Management present in OpenCV Library. Figure 3.5.1 shows the logo of OpenCV Library.

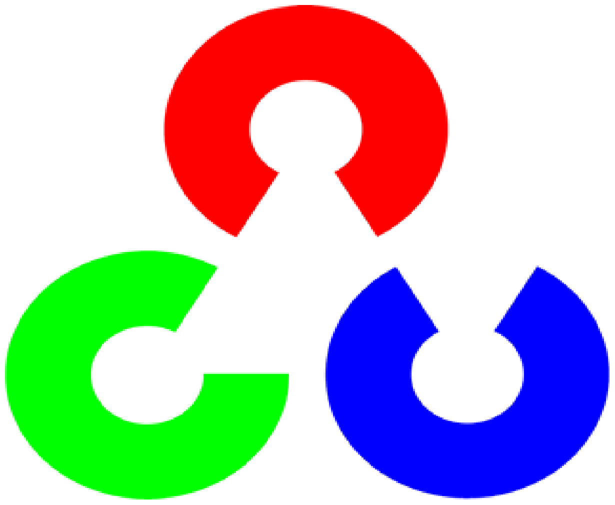


Figure 3.5.1 OpenCV Library

Tensorflow:

Tensorflow is Google's Open Source Machine Learning Framework for dataflow programming across a range of tasks. Nodes in the graph represent mathematical operations, while the graph edges represent the multi-dimensional data arrays (**tensors**) communicated between them. Tensors are just multidimensional arrays, an extension of 2-dimensional tables to data with a higher dimension. There are many features of Tensorflow which makes it appropriate for Deep Learning. So, without wasting any time, let's see how we can implement Object Detection using Tensorflow. The Figure 3.5.2 shows the logo of Tensorflow Library.

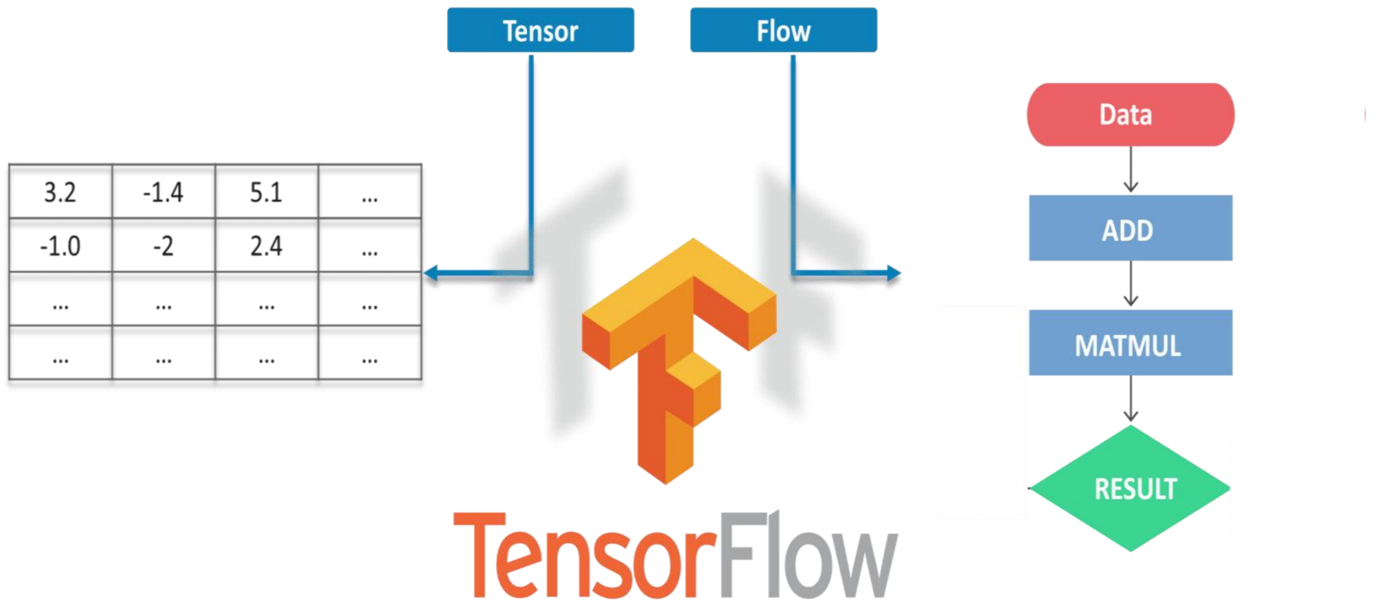


Figure 3.5.2 Tensorflow Library

YOLO:

The R-CNN family of techniques we saw in Part 1 primarily use regions to localize the objects within the image. The network does not look at the entire image, only at the parts of the images which have a higher chance of containing an object. The YOLO framework (You Only Look Once) on the other hand, deals with object detection in a different way. It takes the entire image in a single instance and predicts the bounding box coordinates and class probabilities for these boxes. The biggest advantage of using YOLO is its superb speed – it's incredibly fast and can process 45 frames per second. YOLO also understands generalized object representation. This is one of the best algorithms for object detection and has shown a comparatively similar performance to the R-CNN algorithms. In the upcoming sections, we will learn about different techniques used in YOLO algorithm. The following explanations are inspired by Andrew NG's course on Object Detection which helped me a lot in understanding the working of YOLO. The Figure 3.5.3 shows object detection using YOLO.

Figure 3.5.3 Shows the YOLO Object Detection

Figure 3.5.3 YOLO Object Detection

3.6 Design Requirements

Hardware Requirements:

1. Processor: Intel Core i3 2.30GHz
2. Hard Disk: 250 GB
3. Monitor: 15 VGA Color
4. RAM: 8 GB
5. Camera

Software Requirements:

1. Operating System: Microsoft Windows 7/8/10 (32 or 64 bit)
2. Shell / Visual Studio Code
3. Python
4. OpenCV Library

CHAPTER 4

SYSTEM DESIGN

4.1 Design Description

System design is the process of the defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. System design could be seen as the application of systems theory to product development. Object- oriented analysis and methods are becoming the most widely used methods for computer systems design. Systems design is therefore the process of defining and developing systems to satisfy specified requirements of the user. The UML has become the standard language in object-oriented analysis and design.

4.2 Data Flow Diagram

A DFD is often used as a preliminary step to create an overview of the system without going into great detail, which can later be elaborated. DFDs can also be used for the visualization of data processing. A DFD shows what kind of information will be input to and output from the system, how the data will advance through the system. The Figure 4.2 shows the data flow diagram of the system design.

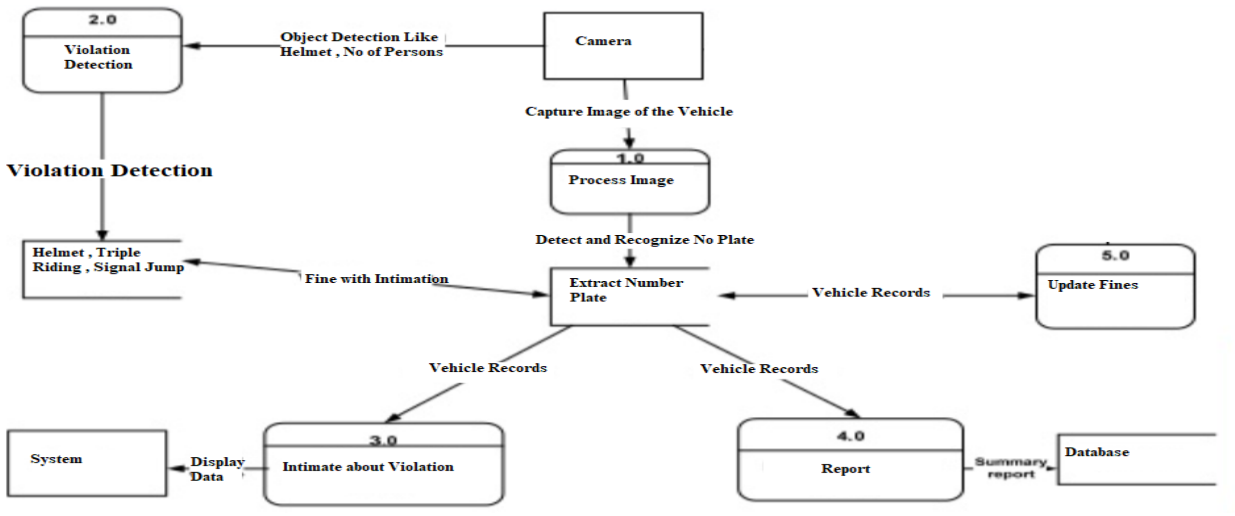


Figure 4.2 Data Flow Diagram

4.3 Use Case, Activity Diagram and Sequence Diagram

Use Case:

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well. While a use case itself might drill into a lot of detail about every possibility, a use case diagram can help provide a higher-level view of the system. It has been said before that "Use case diagrams are the blueprints for your system". They provide the simplified and graphical representation of what the system must actually do. The Figure 4.3.1 shows the use case of the system design.

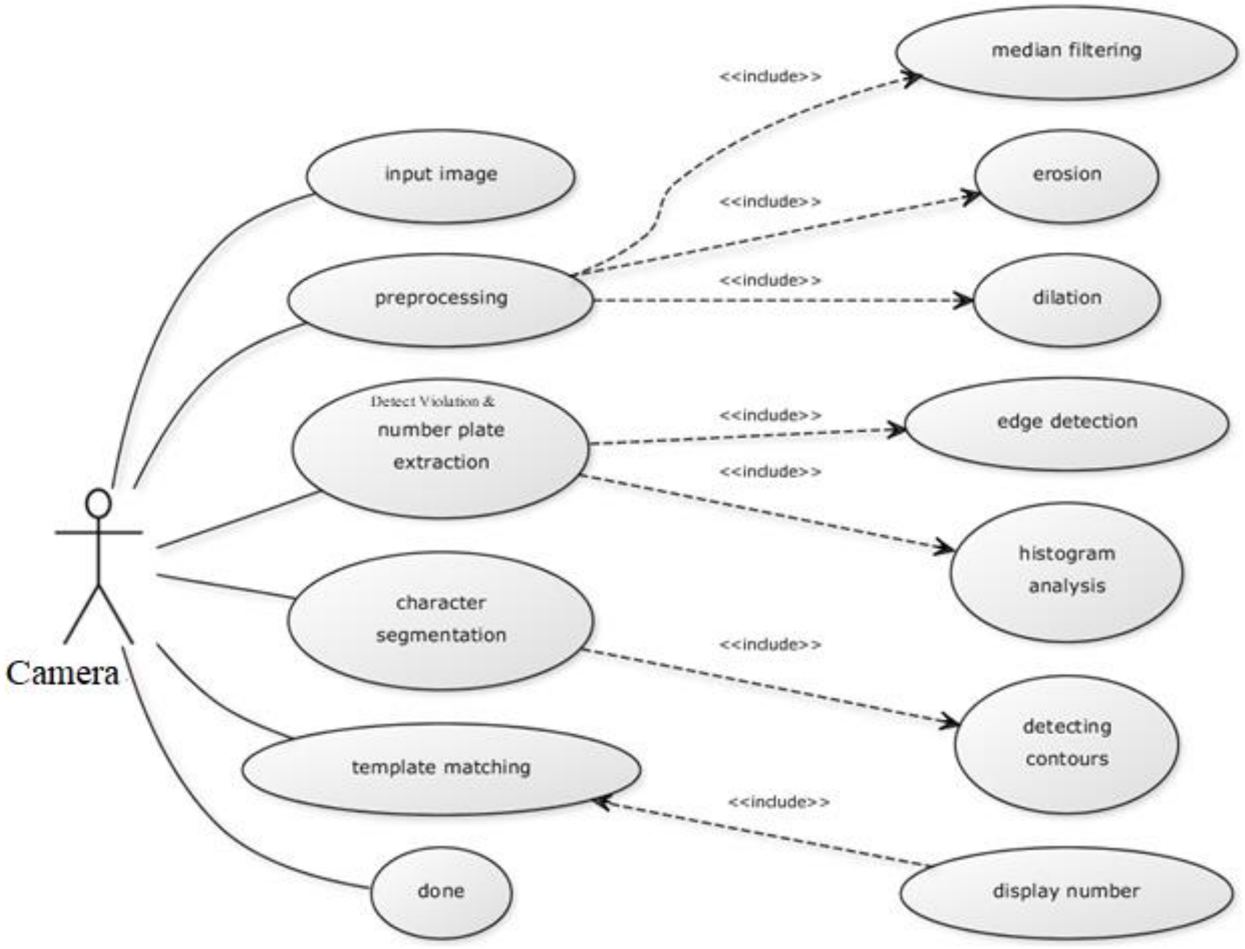


Figure 4.3.1 Use Case Diagram

Activity Diagram:

An activity diagram visually presents a series of actions or flow of control in a system similar to a flowchart or a data flow diagram. Activity diagrams are often used in business process modeling. They can also describe the steps in a use case diagram. Activities modeled can be sequential and concurrent. In both cases an activity diagram will have a beginning (an initial state) and an end (a final state). The Figure 4.3.2 shows the activity diagram of the system design.

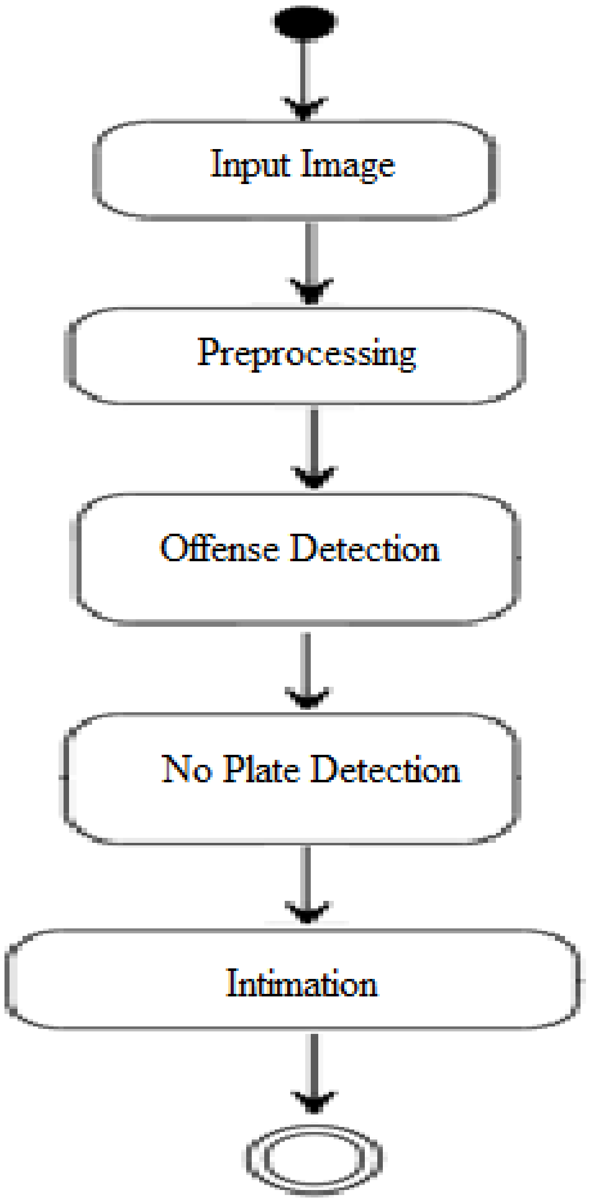


Figure 4.3.2 Activity Diagram

Sequence Diagram:

A sequence diagram is a Unified Modeling Language diagram that illustrates the sequence of messages between objects in an interaction. A sequence diagram consists of a group of objects that are represented by lifelines, and the messages that they exchange over time during the interaction. The Figure 4.3.3 shows the sequence diagram of the system design.

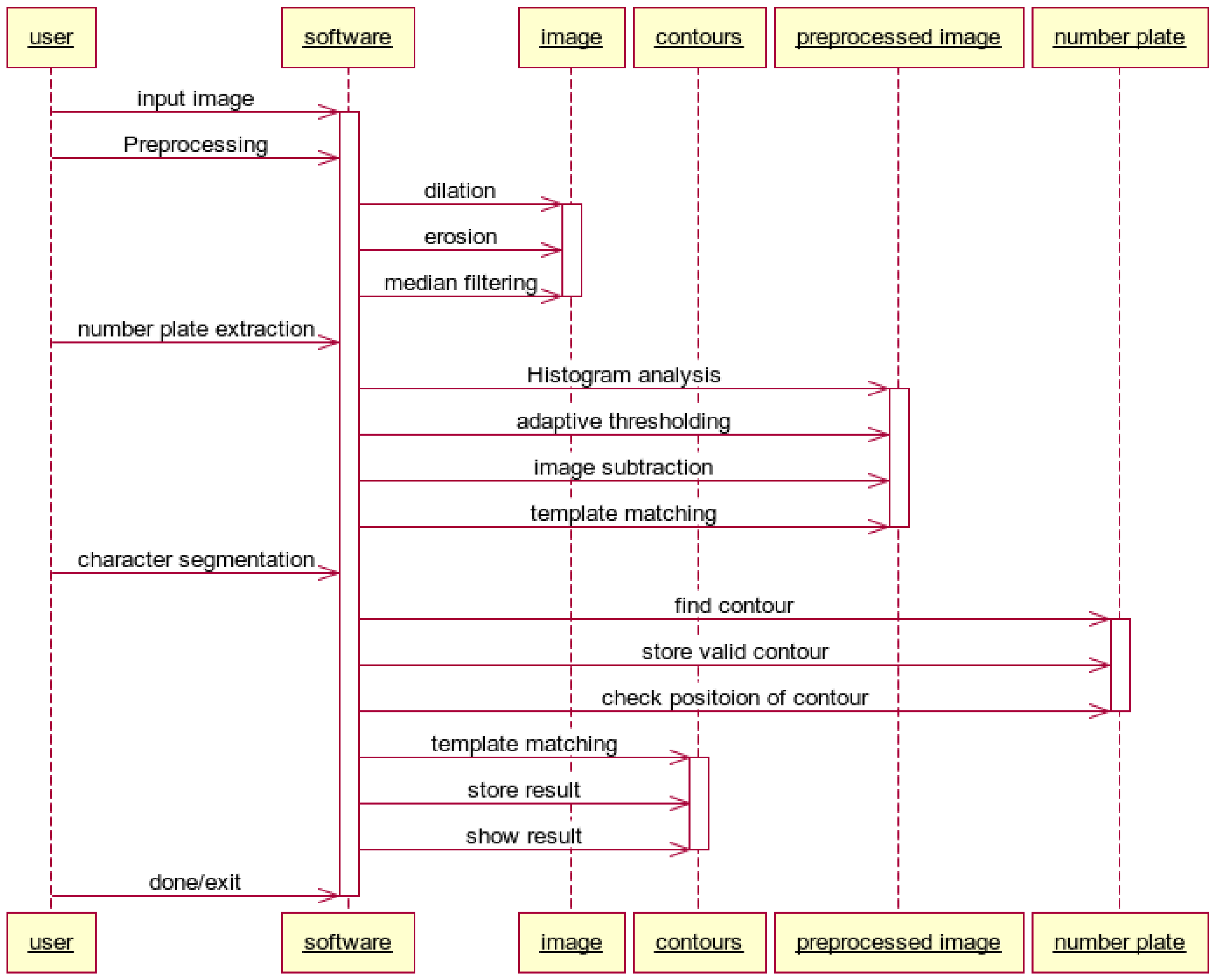


Figure 4.3.3 Sequence Diagram

CHAPTER 5

DETAILED DESIGN

5.1 System Architecture

System architecture is a conceptual model that defines the structure and behavior of the system. It comprises of the system components and the relationship describing how they work together to implement the overall system. The Figure 5.1 shows the system architecture of the system.

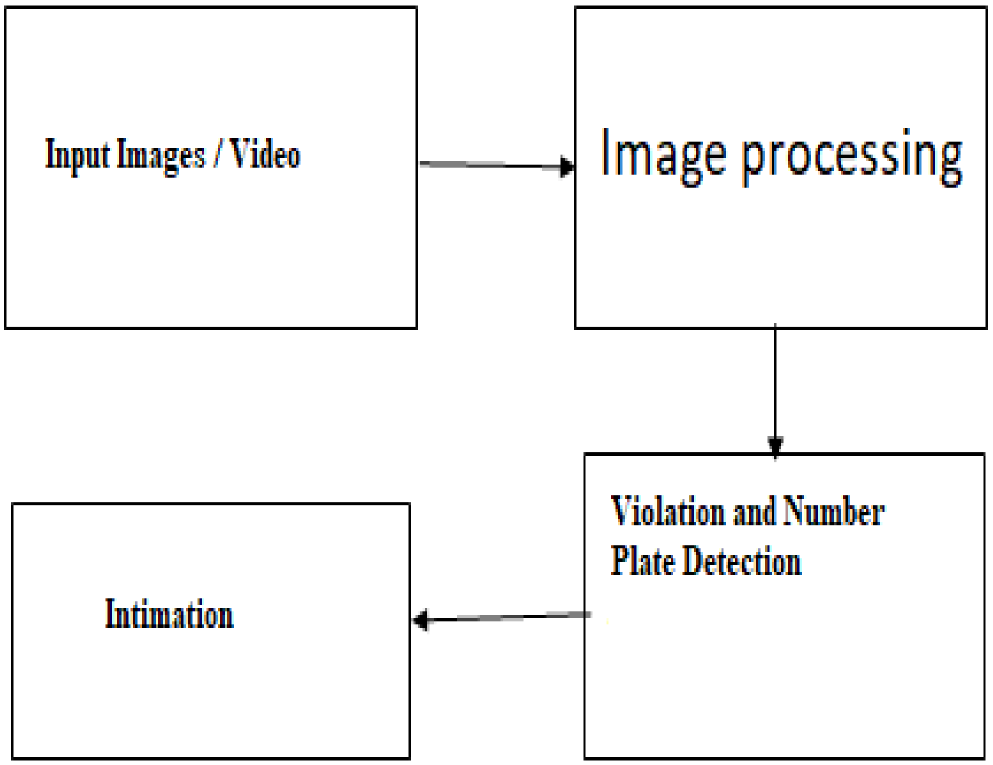


Figure 5.1 System Architecture

5.2 Class Diagram

Class diagrams are the blueprints of your system or subsystem. You can use class diagrams to model the objects that make up the system, to display the relationships between the objects, and to describe what those objects do and the services that they provide. Class diagrams are useful in many stages of system design. In the analysis stage, a class diagram can help you to understand the requirements of your problem domain and to identify its components. The Figure 5.2 shows the class diagram of the system design.

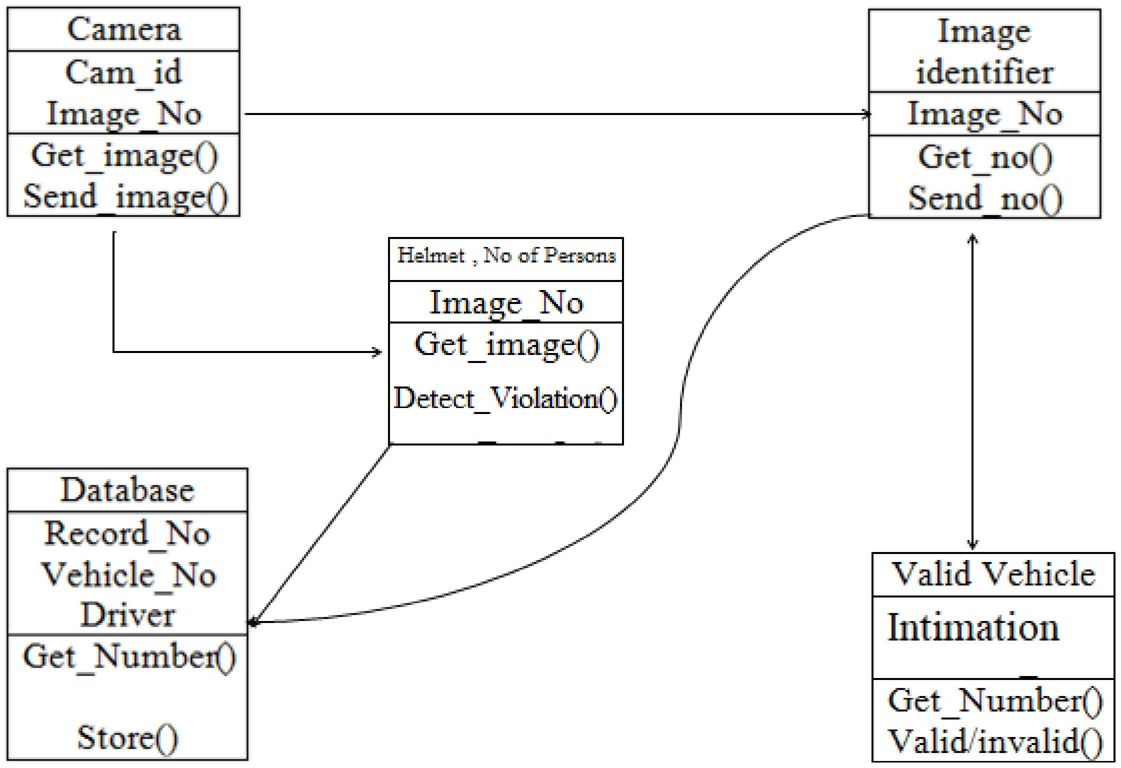


Figure 5.2 Class Diagram

5.3 Sample Code

```
import argparse
import os
import sys
from pathlib import Path

import cv2
import torch
import torch.backends.cudnn as cudnn

FILE = Path(__file__).resolve()
ROOT = FILE.parents[0]  # YOLOv5 root directory
if str(ROOT) not in sys.path:
    sys.path.append(str(ROOT))  # add ROOT to PATH
ROOT = Path(os.path.relpath(ROOT, Path.cwd()))  # relative

from models.common import DetectMultiBackend
from utils.datasets import IMG_FORMATS, VID_FORMATS, LoadImages, LoadStreams
from utils.general import (LOGGER, check_file, check_img_size, check_imshow,
check_requirements, colorstr,
                           increment_path, non_max_suppression, print_args, scale_coords,
strip_optimizer, xyxy2xywh)
from utils.plots import Annotator, colors, save_one_box
from utils.torch_utils import select_device, time_sync
from datetime import datetime
import time

@torch.no_grad()
def run(weights=ROOT / 'best.pt',  # model.pt path(s)
        source='0',  # file/dir/URL/glob, 0 for webcam
        data=ROOT / 'data/coco128.yaml',  # dataset.yaml path
        imgsz=(640, 640),  # inference size (height, width)
        conf_thres=0.25,  # confidence threshold
        iou_thres=0.45,  # NMS IOU threshold
        max_det=1000,  # maximum detections per image
        device='',  # cuda device, i.e. 0 or 0,1,2,3 or cpu
        view_img=False,  # show results
        save_txt=False,  # save results to *.txt
        save_conf=False,  # save confidences in --save-txt labels
        save_crop=False,  # save cropped prediction boxes
        nosave=False,  # do not save images/videos
        classes=None,  # filter by class: --class 0, or --class 0 2 3
        agnostic_nms=False,  # class-agnostic NMS
        augment=False,  # augmented inference
        visualize=False,  # visualize features
        update=False,  # update all models
        project=ROOT / 'runs/detect',  # save results to project/name
        name='exp',  # save results to project/name
        exist_ok=False,  # existing project/name ok, do not increment
        line_thickness=3,  # bounding box thickness (pixels)
        hide_labels=False,  # hide labels
        hide_conf=False,  # hide confidences
        half=False,  # use FP16 half-precision inference
        dnn=False,  # use OpenCV DNN for ONNX inference
        ):
    source = str(source)
    save_img = not nosave and not source.endswith('.txt')  # save inference images
    is_file = Path(source).suffix[1:] in (IMG_FORMATS + VID_FORMATS)
    is_url = source.lower().startswith(('rtsp://', 'rtmp://', 'http://', 'https://'))
    webcam = source.isnumeric() or source.endswith('.txt') or (is_url and not is_file)
    if is_url and is_file:
        source = check_file(source)  # download

    # Directories
    save_dir = increment_path(Path(project) / name, exist_ok=exist_ok)  # increment run
    (save_dir / 'labels' if save_txt else save_dir).mkdir(parents=True, exist_ok=True)  #
make dir
```

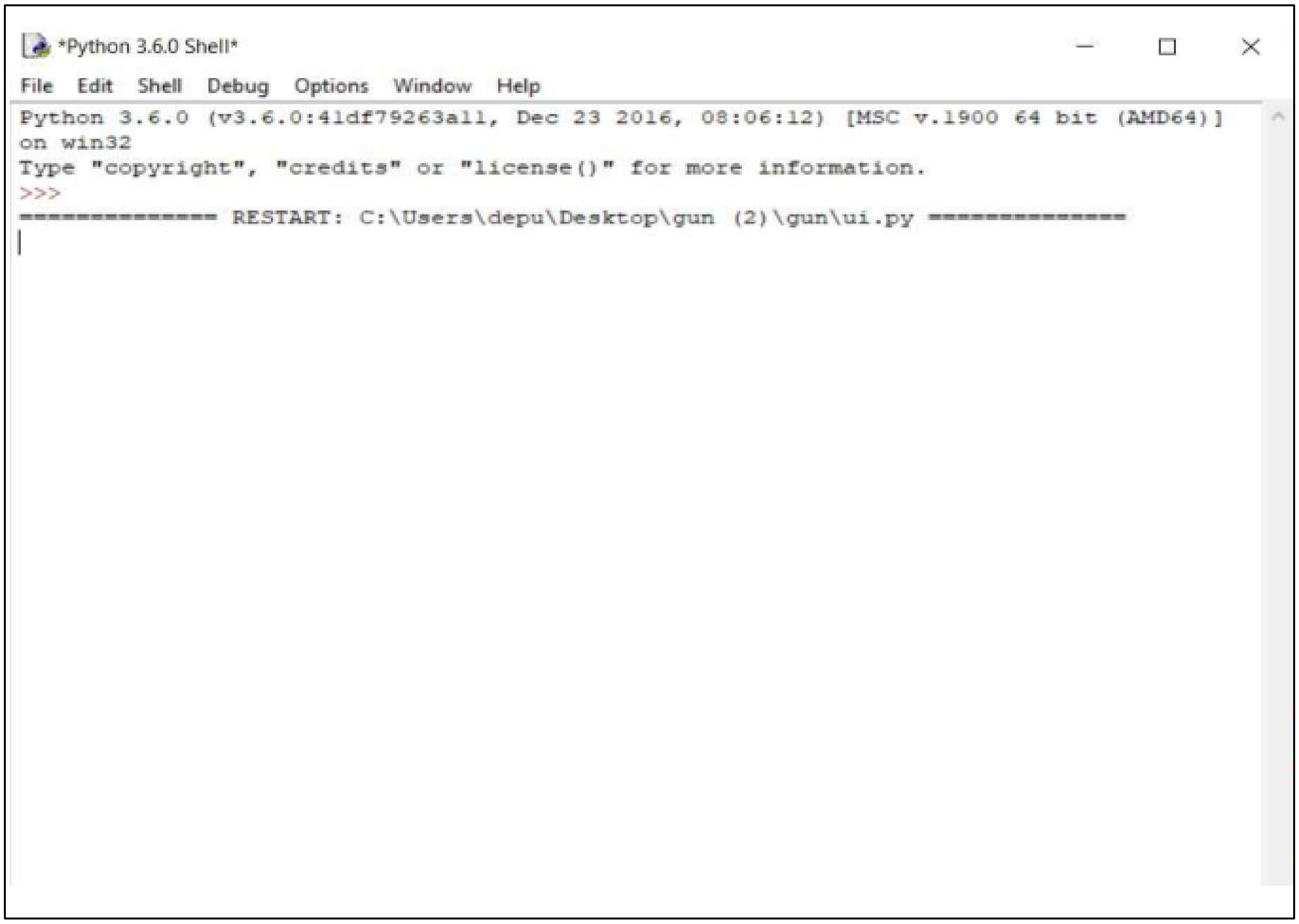
```
    # Load model
    device = select_device(device)
    model = DetectMultiBackend(weights, device=device, dnn=dnn, data=data)
    stride, names, pt, jit, onnx, engine = model.stride, model.names, model.pt, model.jit, 
model.onnx, model.engine
    imgsz = check_img_size(imgsz, s=stride)  # check image size

    # Half
    half &= (pt or jit or onnx or engine) and device.type != 'cpu'  # FP16 supported on 
limited backends with CUDA
    if pt or jit:
        model.model.half() if half else model.model.float()
    elif engine and model.trt_fp16_input != half:
        LOGGER.info('model ' + (
            'requires' if model.trt_fp16_input else 'incompatible with') + ' --half. 
Adjusting automatically.')
        half = model.trt_fp16_input

    # Dataloader
    if webcam:
        view_img = check_imshow()
        cudnn.benchmark = True  # set True to speed up constant image size inference
        dataset = LoadStreams(source, img_size=imgsz, stride=stride, auto=pt)
        bs = len(dataset)  # batch_size
    else:
        dataset = LoadImages(source, img_size=imgsz, stride=stride, auto=pt)
        bs = 1  # batch_size
    vid_path, vid_writer = [None] * bs, [None] * bs

    # Run inference
    model.warmup(imgsz=(1 if pt else bs, 3, *imgsz), half=half)  # warmup
    dt, seen = [0.0, 0.0, 0.0], 0
    for path, im, im0s, vid_cap, s in dataset:
        t1 = time_sync()
        im = torch.from_numpy(im).to(device)
        im = im.half() if half else im.float()  # uint8 to fp16/32
        im /= 255  # 0 - 255 to 0.0 - 1.0
        if len(im.shape) == 3:
            im = im[None]  # expand for batch dim
        t2 = time_sync()
        dt[0] += t2 - t1

        # Inference
        visualize = increment_path(save_dir / Path(path).stem, mkdir=True) if visualize else
False
        pred = model(im, augment=augment, visualize=visualize)
        t3 = time_sync()
        dt[1] += t3 - t2

        # NMS
        pred = non_max_suppression(pred, conf_thres, iou_thres, classes, agnostic_nms,
max_det=max_det)
        dt[2] += time_sync() - t3


        # Process predictions
        for i, det in enumerate(pred):  # per image
            seen += 1
            if webcam:  # batch_size >= 1
                p, im0, frame = path[i], im0s[i].copy(), dataset.count
                s += f'{i}: '
            else:
                p, im0, frame = path, im0s.copy(), getattr(dataset, 'frame', 0)

            p = Path(p)  # to Path
            save_path = str(save_dir / p.name)  # im.jpg
            txt_path = str(save_dir / 'labels' / p.stem) + ('' if dataset.mode == 'image'
else f'_{frame}')  # im.txt
            s += '%gx%g ' % im.shape[2:]  # print string
            gn = torch.tensor(im0.shape)[[1, 0, 1, 0]]  # normalization gain whwh
            imc = im0.copy() if save_crop else im0  # for save_crop
            annotator = Annotator(im0, line_width=line_thickness, example=str(names))

            if len(det):
                # Rescale boxes from img_size to im0 size
                det[:, :4] = scale_coords(im.shape[2:], det[:, :4], im0.shape).round()

                # Print results
                for c in det[:, -1].unique():
                    n = (det[:, -1] == c).sum()  # detections per class
                    s += f"{n} {names[int(c)]}{'s' * (n > 1)}, "  # add to string

                # Write results
                for *xyxy, conf, cls in reversed(det):
                    if save_txt:  # Write to file
                        xywh = (xyxy2xywh(torch.tensor(xyxy).view(1, 4)) / gn).view(-
1).tolist()  # normalized xywh
                        line = (cls, *xywh, conf) if save_conf else (cls, *xywh)  # label
format
                        with open(txt_path + '.txt', 'a') as f:
                            f.write(('%g ' * len(line)).rstrip() % line + '\n')

                    if save_img or save_crop or view_img:  # Add bbox to image
                        c = int(cls)  # integer class
                        label = None if hide_labels else (names[c] if hide_conf else
f'{names[c]} {conf:.2f}')
                        annotator.box_label(xyxy, label, color=colors(c, True))

                        if save_crop:
                            save_one_box(xyxy, imc, file=save_dir / 'crops' / names[c] /
f'{p.stem}.jpg', BGR=True)
```

```
            # Stream results
            im0 = annotator.result()
            #if view_img:
            cv2.imshow(str(p), im0)
            cv2.waitKey(1)  # 1 millisecond

            # Save results (image with detections)
            if save_img:
                if dataset.mode == 'image':
                    cv2.imwrite(save_path, im0)
                else:  # 'video' or 'stream'
                    if vid_path[i] != save_path:  # new video
                        vid_path[i] = save_path
                        if isinstance(vid_writer[i], cv2.VideoWriter):
                            vid_writer[i].release()  # release previous video writer
                        if vid_cap:  # video
                            fps = vid_cap.get(cv2.CAP_PROP_FPS)
                            w = int(vid_cap.get(cv2.CAP_PROP_FRAME_WIDTH))
                            h = int(vid_cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
                        else:  # stream
                            fps, w, h = 30, im0.shape[1], im0.shape[0]
                        save_path = str(Path(save_path).with_suffix('.mp4'))  # force *.mp4
suffix on results videos
                        vid_writer[i] = cv2.VideoWriter(save_path,
cv2.VideoWriter_fourcc(*'mp4v'), fps, (w, h))
                    vid_writer[i].write(im0)

        # Print time (inference-only)
        #LOGGER.info(f'{s}Done. ({t3 - t2:.3f}s)')

    # Print results
    t = tuple(x / seen * 1E3 for x in dt)  # speeds per image
    LOGGER.info(f'Speed: %.1fms pre-process, %.1fms inference, %.1fms NMS per image at shape
{(1, 3, *imgsz)}' % t)
    if save_txt or save_img:
        s = f"\n{len(list(save_dir.glob('labels/*.txt')))} labels saved to {save_dir / 
'labels'}" if save_txt else ''
        LOGGER.info(f"Results saved to {colorstr('bold', save_dir)}{s}")
    if update:
        strip_optimizer(weights)  # update model (to fix SourceChangeWarning)


def parse_opt():
    parser = argparse.ArgumentParser()
    parser.add_argument('--weights', nargs='+', type=str, default=ROOT / 'best.pt', 
help='model path(s)')
    parser.add_argument('--source', type=str, default='C:\\Users\\91889\\Downloads\\₹ 12000 
fine !!! Almost Under My Car _ Triple Riding _ IDIOTS of BANGALORE _ DASHCAM Video _.mp4', 
help='file/dir/URL/glob, 0 for webcam')
    parser.add_argument('--data', type=str, default=ROOT / 'data/coco128.yaml', 
help='(optional) dataset.yaml path')
    parser.add_argument('--imgsz', '--img', '--img-size', nargs='+', type=int, default=[640], 
help='inference size h,w')
    parser.add_argument('--conf-thres', type=float, default=0.25, help='confidence 
threshold')
    parser.add_argument('--iou-thres', type=float, default=0.45, help='NMS IoU threshold')
    parser.add_argument('--max-det', type=int, default=1000, help='maximum detections per 
image')
    parser.add_argument('--device', default='', help='cuda device, i.e. 0 or 0,1,2,3 or cpu')
    parser.add_argument('--view-img', action='store_true', help='show results')
    parser.add_argument('--save-txt', action='store_true', help='save results to *.txt')
    parser.add_argument('--save-conf', action='store_true', help='save confidences in --save-
txt labels')
    parser.add_argument('--save-crop', action='store_true', help='save cropped prediction 
boxes')
    parser.add_argument('--nosave', action='store_true', help='do not save images/videos')
    parser.add_argument('--classes', nargs='+', type=int, help='filter by class: --classes 0, 
or --classes 0 2 3')
```

```
    parser.add_argument('--agnostic-nms', action='store_true', help='class-agnostic NMS')
    parser.add_argument('--augment', action='store_true', help='augmented inference')
    parser.add_argument('--visualize', action='store_true', help='visualize features')
    parser.add_argument('--update', action='store_true', help='update all models')
    parser.add_argument('--project', default=ROOT / 'runs/detect', help='save results to
project/name')
    parser.add_argument('--name', default='exp', help='save results to project/name')
    parser.add_argument('--exist-ok', action='store_true', help='existing project/name ok, do
not increment')
    parser.add_argument('--line-thickness', default=3, type=int, help='bounding box thickness
(pixels)')
    parser.add_argument('--hide-labels', default=False, action='store_true', help='hide
labels')
    parser.add_argument('--hide-conf', default=False, action='store_true', help='hide
confidences')
    parser.add_argument('--half', action='store_true', help='use FP16 half-precision
inference')
    parser.add_argument('--dnn', action='store_true', help='use OpenCV DNN for ONNX
inference')
    opt = parser.parse_args()
    opt.imgsz *= 2 if len(opt.imgsz) == 1 else 1  # expand
    print_args(FILE.stem, opt)
    return opt


def main(opt):
    check_requirements(exclude=('tensorboard', 'thop'))
    run(**vars(opt))


if __name__ == "__main__":
    opt = parse_opt()
    main(opt)
```

CHAPTER 6

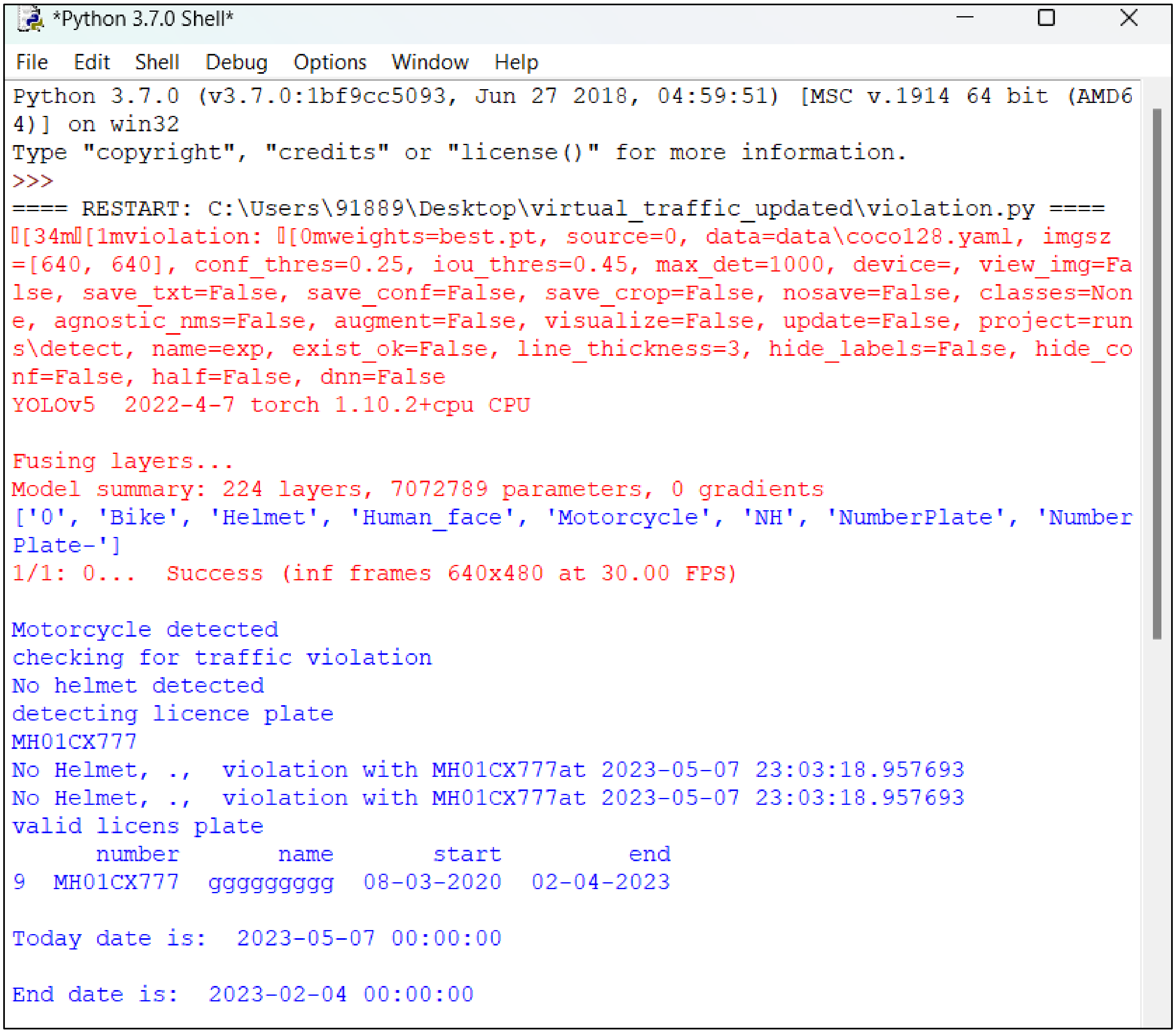
RESULT AND SNAPSHOTS



```
Python 3.6.0 (v3.6.0:41df79263a11, Dec 23 2016, 08:06:12) [MSC v.1900 64 bit (AMD64)]
on win32
Type "copyright", "credits" or "license()" for more information.
>>>
============== RESTART: C:\Users\depu\Desktop\gun (2)\gun\ui.py ==============
```

Figure 6.1 Home Page

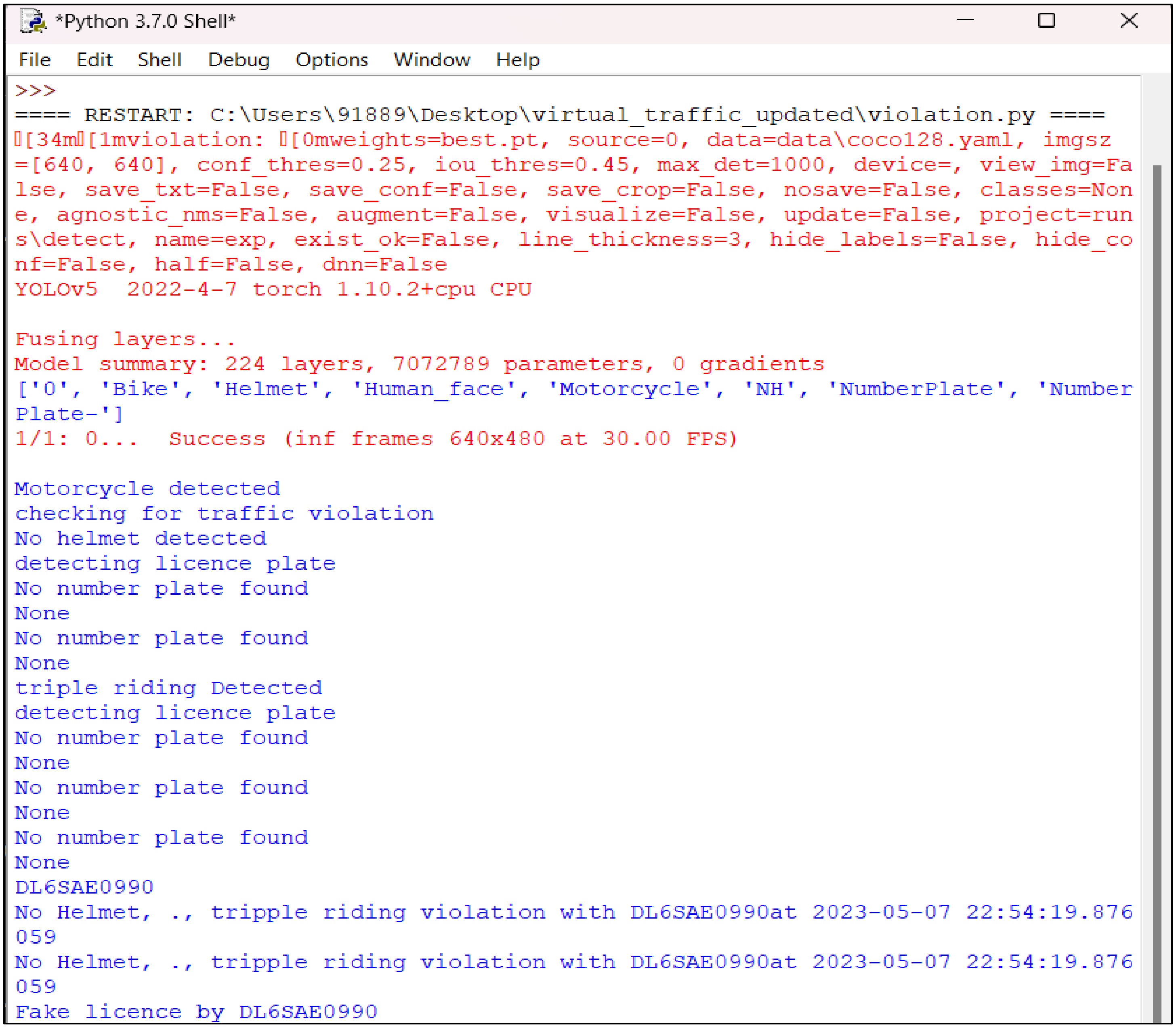
The above Figure 6.1 shows the Home Page of the proposed system. When the system gets start to work this page will be displayed. This will display that proposed system is works by using the Python . This also displaying the path in which the entire information like training, testing and other models related to run this system data is stored. It will also show the options to proceed next steps.



```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD6
4)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> 
==== RESTART: C:\Users\91889\Desktop\virtual_traffic_updated\violation.py ====
[34m[1mviolation: [0mweights=best.pt, source=0, data=data\coco128.yaml, imgsz
=[640, 640], conf_thres=0.25, iou_thres=0.45, max_det=1000, device=, view_img=Fa
lse, save_txt=False, save_conf=False, save_crop=False, nosave=False, classes=Non
e, agnostic_nms=False, augment=False, visualize=False, update=False, project=run
s\detect, name=exp, exist_ok=False, line_thickness=3, hide_labels=False, hide_co
nf=False, half=False, dnn=False
YOLOv5  2022-4-7 torch 1.10.2+cpu CPU

Fusing layers...
Model summary: 224 layers, 7072789 parameters, 0 gradients
['0', 'Bike', 'Helmet', 'Human_face', 'Motorcycle', 'NH', 'NumberPlate', 'Number
Plate-']
1/1: 0...  Success (inf frames 640x480 at 30.00 FPS)

Motorcycle detected
checking for traffic violation
No helmet detected
detecting licence plate
MH01CX777
No Helmet, .,  violation with MH01CX777at 2023-05-07 23:03:18.957693
No Helmet, .,  violation with MH01CX777at 2023-05-07 23:03:18.957693
valid licens plate
      number        name       start         end
9  MH01CX777  gggggggggg  08-03-2020  02-04-2023

Today date is:  2023-05-07 00:00:00

End date is:  2023-02-04 00:00:00
```

Figure 6.2 A Runtime Environment for Executing Programs (Valid)

The above Figure 6.2 shows Runtime Environment for Executing Programs is a crucial component in the development and execution of software applications. In the specific case of Python-based number plate detection, the runtime environment provides the necessary resources and tools to ensure that the program can run efficiently and reliably



```
>>>
==== RESTART: C:\Users\91889\Desktop\virtual_traffic_updated\violation.py ====
[34m[1mviolation: [0mweights=best.pt, source=0, data=data\coco128.yaml, imgsz
=[640, 640], conf_thres=0.25, iou_thres=0.45, max_det=1000, device=, view_img=Fa
lse, save_txt=False, save_conf=False, save_crop=False, nosave=False, classes=Non
e, agnostic_nms=False, augment=False, visualize=False, update=False, project=run
s\detect, name=exp, exist_ok=False, line_thickness=3, hide_labels=False, hide_co
nf=False, half=False, dnn=False
YOLOv5  2022-4-7 torch 1.10.2+cpu CPU

Fusing layers...
Model summary: 224 layers, 7072789 parameters, 0 gradients
['0', 'Bike', 'Helmet', 'Human_face', 'Motorcycle', 'NH', 'NumberPlate', 'Number
Plate-']
1/1: 0...  Success (inf frames 640x480 at 30.00 FPS)

Motorcycle detected
checking for traffic violation
No helmet detected
detecting licence plate
No number plate found
None
No number plate found
None
triple riding Detected
detecting licence plate
No number plate found
None
No number plate found
None
No number plate found
None
DL6SAE0990
No Helmet, ., tripple riding violation with DL6SAE0990at 2023-05-07 22:54:19.876
059
No Helmet, ., tripple riding violation with DL6SAE0990at 2023-05-07 22:54:19.876
059
Fake licence by DL6SAE0990
```

Figure 6.3 A Runtime Environment for Executing Programs(Fake)

The above Figure 6.3 shows the shows Runtime Environment for Executing Programs of the Number Plate Detection. It shows that first it detects the motorcycle, then checks for any traffic violation for the vehicle, checks if the person is wearing helmet or not, and then checks for the number plate

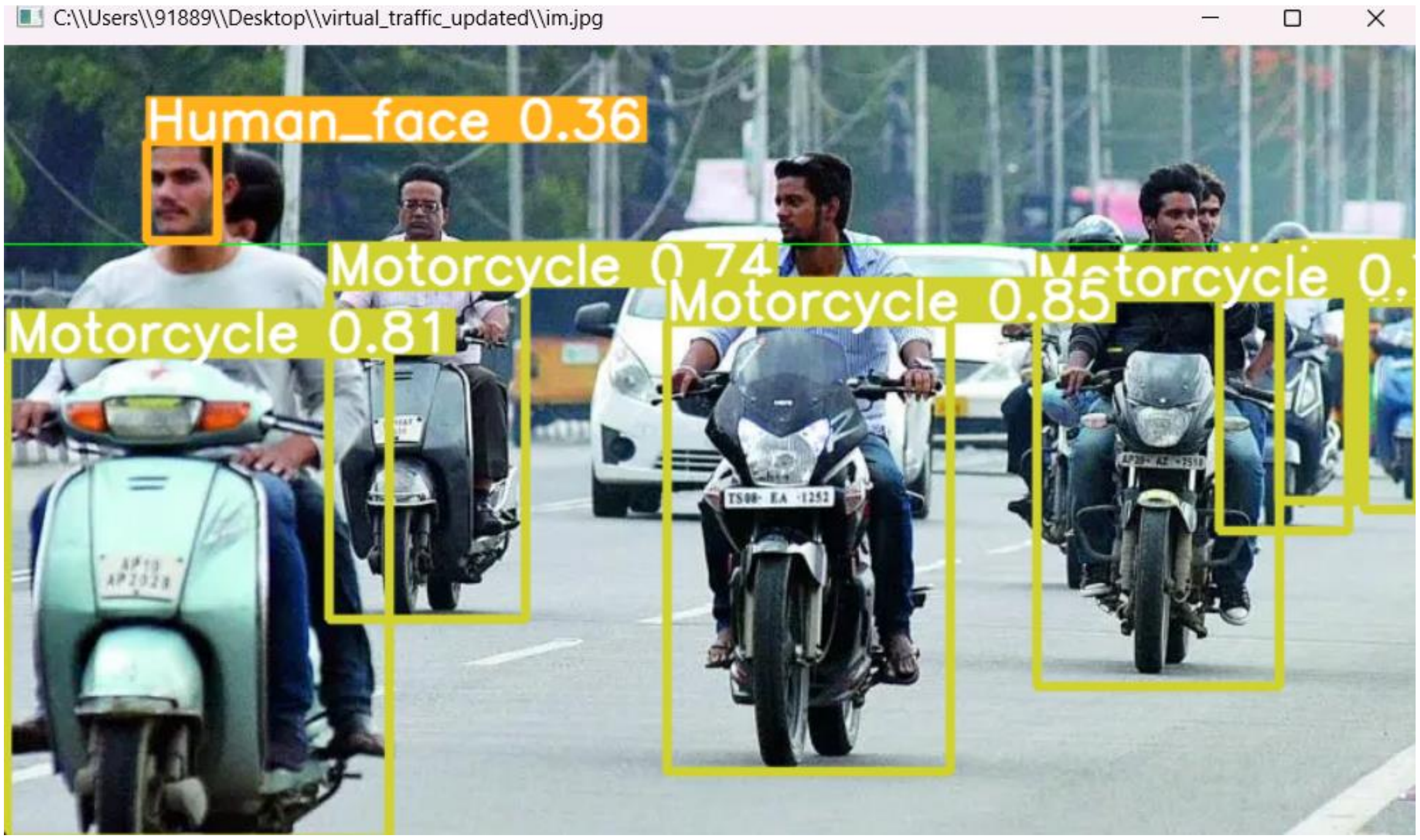


Figure 6.4 Snapshot of the Output

The above Figure 6.4 shows the Snapshot of the Output of the project. Here we can see that multiple motorcycles being detected at the same time and checking for any violation.

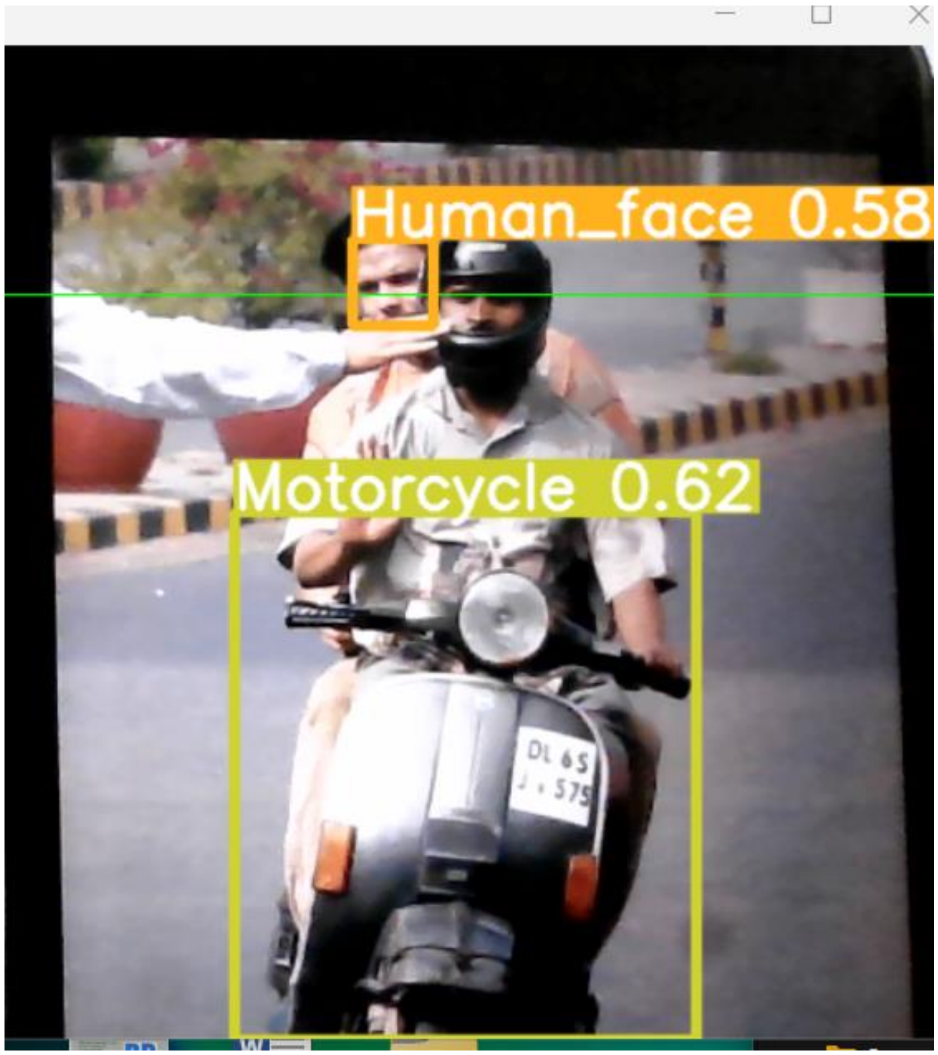


Figure 6.5 Snapshot of the Output

The above Figure 6.5 shows the Snapshot of the Output of the project. Here we can see that motorcycles being detected at the same time and checking for any violation and helmet detection.

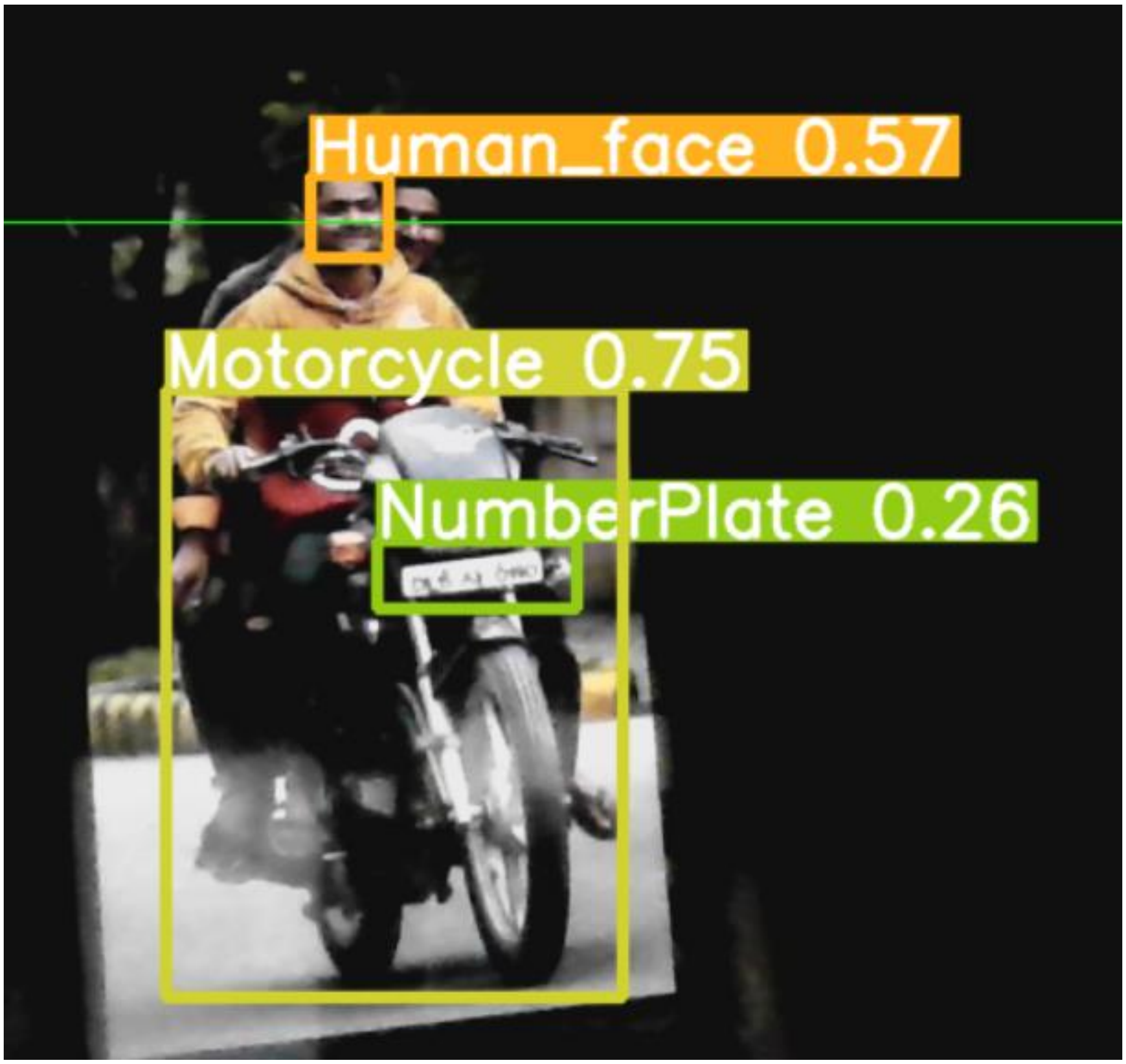
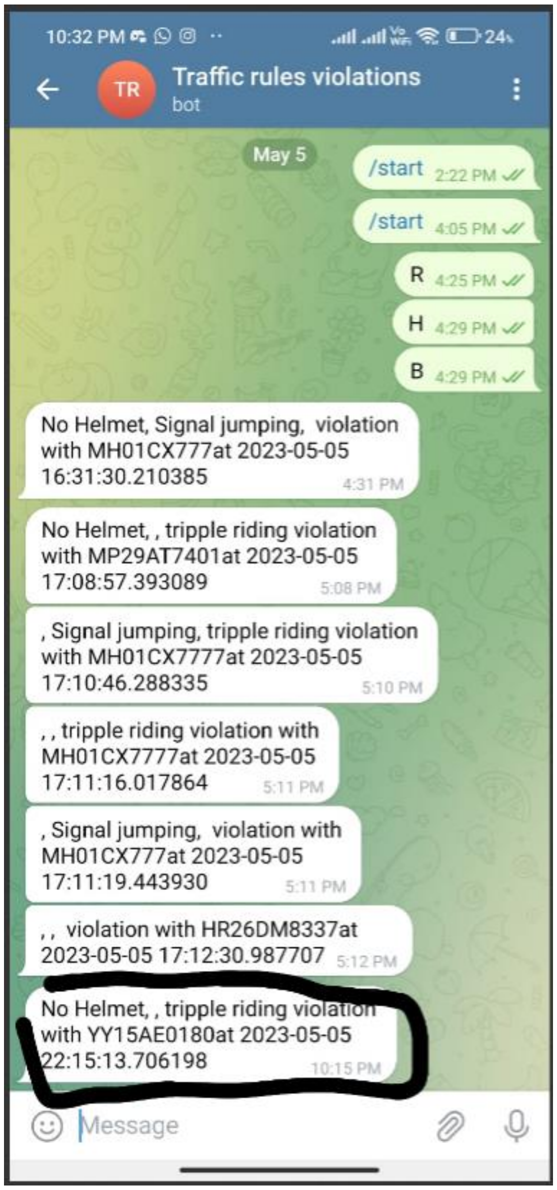


Figure 6.6 Snapshot of the Output

The above Figure 6.6 shows the Snapshot of the Output of the project. Here we can see that motorcycles being detected without helmet and triple riding and number plate with accuracy rate.



/start 2:22 PM

/start 4:05 PM

R 4:25 PM

H 4:29 PM

B 4:29 PM

No Helmet, Signal jumping, violation with MH01CX777at 2023-05-05 16:31:30.210385 4:31 PM

No Helmet, , tripple riding violation with MP29AT7401at 2023-05-05 17:08:57.393089 5:08 PM

, Signal jumping, tripple riding violation with MH01CX7777at 2023-05-05 17:10:46.288335 5:10 PM

, , tripple riding violation with MH01CX7777at 2023-05-05 17:11:16.017864 5:11 PM

, Signal jumping, violation with MH01CX777at 2023-05-05 17:11:19.443930 5:11 PM

, , violation with HR26DM8337at 2023-05-05 17:12:30.987707 5:12 PM

No Helmet, , tripple riding violation with YY15AE0180at 2023-05-05 22:15:13.706198 10:15 PM

Figure 6.7 Alert Notification

The above Figure 6.7 shows the Alert Notification. When a violation is detected it sends a message with what type of violation has been done along with the number plate and time and date of the violation occurred

CHAPTER 7

TESTING

7.1 Test Units

Testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not. Testing is executing a system in order to identify any gaps, errors, or missing requirements in contrary to the actual requirements. All the tests should be traceable to customer requirements. There are two types of testing as follows.

Black-Box Testing:

The technique of testing without having any knowledge of the interior workings of the application is called black-box testing. The tester is oblivious to the system architecture and does not have access to the source code. Typically, while performing a black-box test, a tester will interact with the system's user interface by providing inputs and examining outputs without knowing how and where the inputs are worked upon.

White-Box Testing:

White-box testing is the detailed investigation of internal logic and structure of the code. White-box testing is also called glass testing or open-box testing. In order to perform white-box testing on an application, a tester needs to know the internal workings of the code. The tester needs to have a look inside the source code and find out which unit/chunk of the code is behaving inappropriately.

7.2 Approaches for Testing

Unit Testing:

Unit testing is a software development process in which the smallest testable parts of an application, called units, are individually and independently scrutinized for proper operation. Unit testing is often automated but it can also be done manually. The goal of unit testing is to isolate each part of the program and show that individual parts are correct.

Integration Testing:

Integration testing is a level of software testing where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units. Test drivers and test stubs are used to assist in Integration Testing. Integration testing is defined as the testing of combined parts of an application to determine if they function correctly. It occurs after unit testing and before validation testing. Integration testing can be done in two ways: Bottom-up integration testing and Top-down integration testing.

System Testing:

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black-box testing, and as such, should require no knowledge of the inner design of the code or logic.

CHAPTER 8

CONCLUSION AND FUTURE SCOPE

8.1 Conclusion

The system uses a sequence of image processing techniques for recognizing the vehicle from the database stored on the system. The system is implemented in the MATLAB platform and its performance is tested on real license plate images. The simulation output display that the system vigorously detects and identifies the vehicle using a license number plate against different lighting situations and can be implemented at the entrance of a highly restricted area. The system works adequately for wide variations in illumination circumstances and various types of number plates commonly found in India. It is a great replacement to the existing proprietary systems, even though there are some restrictions with high resolution to detect the number plate using OpenCV and python which are easy to comprehend and make changes.

8.2 Future Scope

The future scope is that the automated vehicle recognition system plays an important role in detecting dangers to defense. Also, it can enhance the security related to women as they can easily detect the number plate before using a taxi. The scheme's robustness can be increased if a radiant and sharp camera is used. Government should take some interest in inventing this system as this system is moneysaving and eco-friendly if used effectively in diverse areas.

REFERENCES

[1] Traffic Rules Violation Detection using Machine Learning Techniques. Authors: P. Srinivas Reddy, T. Nishwa, R. Shiva Kiran Reddy, Ch Sadviq , K. Rithvik, IEEE 2021

[2] An Efficient FPGA Implementation of Optical Character Recognition for License Plate Recognition, Authors: Yuan Jing, Bahar Youssefi, Mitra Mirhassani, 2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE)

[3] Autonomous Vehicle And Real Time Road Lanes Detection And Tracking, Authors: Farid Bounini, Denis Gingras, Vincent Lapointe, Herve Pollart, 2015 IEEE

[4] Vehicle Number Plate Detection and Recognition using Bounding Box Method, Authors: Mahesh Babu K, M V Raghunadh, 2016 International Conference on Advanced Communication Control and Computing Technologies (ICACCCT)

[5] Dynamic Traffic Rule Violation Monitoring System Using Automatic Number Plate Recognition with SMS Feedback, Authors: R Shreyas, Pradeep Kumar B V, Adithya H B, Padmaja B, 2017 2nd International Conference on Telecommunication and Networks

[6] Image Recognition for Automatic Number Plate Surveillance, Authors: P.Meghana, S. Sagar Imambi, P. Sivateja, K. Sairam, International Journal of Innovative Technology and Exploring Engineering (IJITEE), ISSN: 2278-3075, Volume-8 Issue-4, February 2019