

**AUTOMATIC DETECTION OF LICENSE PLATE NUMBER OF MOTORCYCLISTS**

**WITHOUT HELMET**

##### A PROJECT REPORT

###### ***Submitted by***

##### BANGALORE HARIKA [REGISTER NO:211417104031]

**BRIGHTLIN SELVAMARY [REGISTER NO:211417104038]**

**MADDINA SREELEKHA [REGISTER NO:211417104135]**

***in partial fulfillment for the award of the degree***

***of***

**BACHELOR OF ENGINEERING**

IN

# COMPUTER SCIENCE AND ENGINEERING

PANIMALAR ENGINEERING COLLEGE, CHENNAI-600123.

ANNA UNIVERSITY: CHENNAI 600 025

##### MARCH 2021

**BONAFIDE CERTIFICATE**

##### Certified that this project report “AUTOMATIC DETECTION OF LICENSE PLATE NUMBER OF MOTORCYCLISTS WITHOUT HELMET” is the bonafide work of “BANGALORE HARIKA [REGISTER NO:211417104031], BRIGHTLIN SELVAMARY A [REGISTER NO:211417104038], MADDINA SREELEKHA [REGISTERNO:211417104135]” who carried out the project work under my supervision.

**SIGNATURE SIGNATURE**

**Dr.S.MURUGAVALLI,M.E.,Ph.D., ANITHA MOSES V**

**HEAD OF THE DEPARTMENT SUPERVISOR**

**ASSOCIATE PROFESSOR**

DEPARTMENT OF CSE, DEPARTMENT OF CSE,

PANIMALAR ENGINEERING COLLEGE, PANIMALAR ENGINEERING COLLEGE,

NASARATHPETTAI, NASARATHPETTAI,

POONAMALLEE, POONAMALLEE,

CHENNAI-600 123. CHENNAI-600 123.

Certified that the above candidate(s) was/ were examined in the Anna University Project Viva-Voce Examination held on...........................

**INTERNAL EXAMINER EXTERNAL EXAMINER**

**ACKNOWLEDGEMENT**

We express our deep gratitude to our respected Secretary and Correspondent **Dr.P.CHINNADURAI, M.A., Ph.D.** for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We would like to extend our heartfelt and sincere thanks to our Directors **Tmt.C.VIJAYARAJESWARI, Dr.C.S.SAKTHIKUMAR,M.E.,Ph.D.,** and

**Tmt. SARANYASREE SAKTHIKUMAR B.E.,M.B.A.,** for providing us with the necessary facilities for completion of this project.

We also express our gratitude to our Principal **Dr.K.Mani, M.E., Ph.D.** for his timely concern and encouragement provided to us throughout the course.

We thank the HOD of CSE Department, **Dr. S.MURUGAVALLI , M.E.,Ph.D.,**for the support extended throughout the project.

We would like to thank my Project Guide, **ANITHA MOSES V** and all the faculty members of the Department of CSE for their advice and suggestions for the successful completion of the project.

BANGALORE HARIKA

BRIGHTLIN SELVAMAR. A

MADDINA SREELEKHA

**ABSTRACT**

We develop an helmet detection method combining classification and cluster. Helmet detection is an important, yet challenging vision task. It is a critical part in many applications such as traffic surveillance. Our proposed method work is as follows, Pre-processing, Feature Extraction and classification. We demonstrate our proposed work by using live images and stored images . Then, our method will classify whether the person is wearing helmet or not. As far as the robustness and effectiveness are concerned, our method is better than the existing algorithms. The project presents license plate recognition system using connected component analysis and template matching model for accurate identification. Automatic license plate recognition (ALPR) is the extraction of vehicle license plate information from an image.The system model uses already captured images for this recognition process. First the recognition system starts with character identification based on number plate extraction, Splitting characters and template matching. ALPR as a real life application has to quickly and successfully process license plates under different environmental conditions, such as day time. Finally, our project determines the license plate number of motorcyclists without helmet and prevent many accidents .

TABLE OF CONTENTS

| **CHAPTERNO.** | **TITLE** | **PAGE NO.** |
| --- | --- | --- |
|  | **ABSTRACT** | Iv |
|  | **LIST OF FIGURES** | Vii |
| **1.** | **INTRODUCTION** |  |
|  | 1.1 Overview | 1 |
|  | 1.2 Problem Definition | 1 |
| **2.** | **LITERATURE SURVEY** | 2 |
| **3.** | **SYSTEM ANALYSIS** |  |
|  | 3.1 Existing System | 6 |
|  | 3.2 Proposed system | 6 |
|  | 3.3 Hardware Environment | 7 |
|  | 3.4 Software Environment | 7 |
| **4.** | **SYSTEM DESIGN** |  |
|  | 4.1 Data Flow Diagram | 13 |
|  | 4.2 UML Diagrams | 14 |
|  |  |  |
|  |  |  |
|  |  |  |
| **5.** | **SYSTEM ARCHITECTURE** |  |
|  | 5.2 Architecture overview | 18 |
|  | 5.2 Module Design Specification | 19 |
| **6.** | **SYSTEM IMPLEMENTATION** | 22 |
| **7.** | **SYSTEM TESTING** |  |
|  | 7.1 Unit Testing | 42 |
|  | 7.2 Integration Testing | 43 |
| **8.** | **CONCLUSION** |  |
|  | 8.1 Conclusion and Future Enhancements | 45 |
|  | **APPENDICES** |  |
|  | A.1 Sample Screens | 47 |
|  | A.2 Publications | 48 |
|  | **REFERENCES** | 48 |

**LIST OF FIGURES**

4.1 Data Flow Diagram for helmet and license plate detection

4.2.1 Use case Diagram for helmet and license plate detection

4.2.3 Activity Diagram for helmet and license plate detection

5.1.1 Architecture Diagram for helmet detection

5.1.2 Architecture Diagram for license plate detection

A.1.1 Screenshot for helmet detected

A.1.2 Screenshot for helmet not detected

A.1.3 Screenshot for license plate detection

**1. INTRODUCTION**

**1.1 OVERVIEW**

The Helmet detection system is recommended for the identification of a particular person with no helmet. The input to the system is captured video which is then converted into images. Then preprocessing functions are applied to the image such as background noise, enhancing contrast and binarization of images. In order to know the characteristics of the image, the Feature descriptor algorithm is used to extract the exact feature and to differentiate one feature from another. CNN classifier is used to split the images into two groups, one for training data and another for test data to use in classification. After extracting the Region of Interest (RoI), the CNN classifier is being trained by a certain number of pictures wearing a helmet is provided. By matching RoI and trained features, it will be determined whether motorcyclists are wearing a helmet or not. Convolutional Neural Network is used to solve the classification problem efficiently. Then the license plate number is detected.

**1.2 PROBLEM DEFINITION**

Two-wheeler is the most convenient and easy mode of transportation. It is mandatory to wear a helmet in heavy traffic areas to prevent accidents. By considering the use of helmet, Governments have made it a punishable offense to ride a bike without a helmet and have adopted manual strategies to catch the violators. Image processing means processing the images based on the application with the specific parameters. Pre- processing is the first step to improve the quality of the images. The feature descriptor algorithm is used to extract the exact feature and to differentiate one feature from another. CNN classifier is used to split the images into two groups, one for training data and another for test data to use in classification. A Convolutional Neural Network (CNN) is a class of artificial neural networks used in image processing that is specifically designed to process pixel data. Then the license plate is detected.

**2.LITERATURE SURVEY**

**1. Automatic detection of bike-riders without helmet using surveillance videos in real-time [8]**

**AUTHOR**: Kunal Dahiya

**PUBLISH**: 2016

In this paper, we propose an approach for automatic detection of bike-riders without helmet using surveillance videos in real time. The proposed approach first detects bike riders from surveillance video using background subtraction and object segmentation. Then it determines whether bike-rider is using a helmet or not using visual features and binary classifier. Also, we present a consolidation approach for violation reporting which helps in improving reliability of the proposed approach. In order to evaluate our approach, we have provided a performance comparison of three widely used feature representations namely histogram of oriented gradients (HOG), scale-invariant feature transform (SIFT), and local binary patterns (LBP) for classification. The experimental results show detection accuracy of 93.80% on the real world surveillance data. It has also been shown that proposed approach is computationally less expensive and performs in real-time with a processing time of 11.58 ms per frame.

**DRAWBACKS**:

Video surveillance based methods are passive and require significant human assistance. It is not an efficient solution due to its requirement of dedicated hardware.

**2. Visual Big Data Analytics for Traffic Monitoring in Smart City [6]**

**AUTHOR**: Dinesh Singh

**PUBLISH**:2014

The application such as video surveillance for traffic control in smart cities needs to analyze the large amount (hours/days) of video footage in order to locate the people who are violating the traffic rules. The traditional computer vision techniques are unable to analyze such a huge amount of visual data generated in real-time. So, there is a need for visual big data analytics which involves processing and analyzing large scale visual data such as images or videos to find semantic patterns that are useful for interpretation. In this paper, we propose a framework for visual big data analytics for automatic detection of bike-riders without helmet in city traffic. We also discuss challenges involved in visual big data analytics for traffic control in a city scale surveillance data and explore opportunities for future research.

**DRAWBACKS:**

Environment conditions like illumination variance over the day, shadows, shaking tree branches, and other sudden changes make it difficult to recover and update background from continuous stream of frames.

**3. Mission on! Innovations in bike systems to provide a safe ride based on IOT [2]**

**AUTHOR**: Archana D

**PUBLISH**:2017

Intelligence applications are being developed that make machines more sophisticated in their way of learning and to make decisions. Accident is a specific, unpredicted external action that happens unexpectedly with no apparent or deliberate cause but with marked effects. With the increasing number of bike riders and the number of accidents happening each year our paper focuses on the methods that can be implemented to ensure safety while driving. Distraction of the driver's attention is the major cause of these accidents. Nowadays wearing helmet has been made mandatory. But still the rules are being violated. Message transmitting sensors are equipped in the speedometer of bike and also in the bike's helmet. The most important feature of the bike is that the bike's engine gets start only when the person wears helmet. This system also checks the approaching vehicle's speed on either side of the road and generate vibrations in the bike's handlebar. This advanced development is bringing about a new era of productivity for the latest ideas on an astounding scale, understanding their efficiency, speed and functionality.

**DRAWBACKS**

Cost is one major hindrance to the widespread use of safety systems

**4. An Improved License Plate Location Method Based On Edge [9]**

**AUTHOR**: Rongbao Chen

**PUBLISH** : 2012

License plate location as one of the key steps in license plate recognition system, the positioning accuracy is direct impact on the effect of license plate recognition. In this paper, for the plate image that is under different backgrounds and lighting conditions,a license plate location method based on an improved prewitt arithmetic operator was proposed. Firstly,do improved prewitt operation on the preprocessed plate image. Then, use the characteristics of vehicle license, adopt the horizontal and vertical projection method to determine the location of the upper and lower edge around the edge position, in order to achieve the positioning of vehicle license. The experiments results show that the algorithm with high accuracy, positioning speed and have good practical value.

**DRAWBACKS**

More similar database is needed to compare all edge-based methods.

**5. An Automatic Number Plate Recognition System under Image Processing [12]**

**AUTHOR**: Sarbjit Kaur

**PUBLISH**: 2017

Automatic Number Plate Recognition system is an application of computer vision and image processing technology that takes photograph of vehicles as input image and by extracting their number plate from whole vehicle image , it display the number plate information into text. Mainly the ANPR system consists of 4 phases: - Acquisition of Vehicle Image and Pre-Processing, Extraction of Number Plate Area, Character Segmentation and Character Recognition. The overall accuracy and efficiency of whole ANPR system depends on number plate extraction phase as character segmentation and character recognition phases are also depend on the output of this phase. Further the accuracy of Number Plate Extraction phase depends on the quality of captured vehicle image. Higher be the quality of captured input vehicle image more will be the chances of proper extraction of vehicle number plate area. The existing methods of ANPR works well for dark and bright/light categories image but it does not work well for Low Contrast, Blurred and Noisy images and the detection of exact number plate area by

using the existing ANPR approach is not successful even after applying existing filtering and enhancement technique for these types of images. Due to wrong extraction of number plate area, the character segmentation and character recognition are also not successful in this case by using the existing method. To overcome these drawbacks I proposed an efficient approach for ANPR in which the input vehicle image is pre-processed firstly by iterative bilateral filtering , adaptive histogram equalization and number plate is extracted from pre-processed vehicle image using morphological operations, image subtraction, image binarization/thresholding, sobel vertical edge detection and by boundary box analysis. Sometimes the extracted plate area also contains noise, bolts, frames etc. So the extracted plate area is enhanced by using morphological operations to improve the quality of extracted plate so that the segmentation phase gives more successful output. The character segmentation is done by connected component analysis and boundary box analysis and finally in the last character recognition phase, the characters are recognized by matching with the template database using correlation and output results are displayed. This approach works well for low contrast, blurred, noisy as well as for dark and light/bright category images. The comparison is done between the ANPR with Adaptive Histogram Equalization and Iterative Bilateral Filtering that is the proposed approach and the existing ANPR approach using metrics: MSE, PSNR and Success rate.

**DRAWBACKS**:

In some cases, bad weather and hindrances can make automatic license plate recognition systems not completely effective.

**3**. **SYSTEM ANALYSIS**

**3.1 EXISTING SYSTEM**

An object is seperated from its background by using a segmentation technique called thresholding. This process  involves, comparing each pixel value of the image (pixel intensity) to a specified threshold. Neighbouring pixels are combined after thresholding into a ternary pattern. Computing a histogram of these ternary values will result in a large range, so the ternary pattern is split into two binary patterns. Histograms are concatenated to generate a descriptor double the size of LBP. Then object recognition is the technology used for finding and identifying objects in an image or video sequence. Then the license plate extraction is based on color and character features followed by texture based segmentation.

**3.2 PROPOSED SYSTEM**

The identification of objects in an image would probably start with image processing techniques such as noise removal, followed by (low-level) feature extraction to locate lines, regions and possibly areas with certain textures. Then an algorithm called feature descriptor is used  which takes an image and outputs feature descriptors/feature vectors. Then a neural network called convolutional neural network (CNN) is used that has one or more convolutional layers and are used mainly for image processing, classification, segmentation and also for other auto correlated data. Then the license plate recognition system based on thresholding and Template matching using optical character recognition for automatic number identification is performed.

**3.3 HARDWARE ENVIRONMENT**

* Memory of 4 GB RAM.
* 64 bit distribution capable of running 32 bit application.
* 1200\*800 minimum screen resolution.
* 2 GB of available disk space minimum 4 GB recommended.

**3.4 SOFTWARE ENVIRONMENT**

* Backend Language: Python
* Technology: Deep Learning
* Operating System: Windows 10
* Platform: IDLE (Python's Integrated Development and Learning Environment)

**OPENCV-PYTHON**

Python is a general purpose programming language started by Guido van Rossum, which became very popular in short time mainly because of its simplicity and code readability. It enables the programmer to express his ideas in fewer lines of code without reducing any readability. Compared to other languages like C/C++, Python is slower. But another important feature of Python is that it can be easily extended with C/C++. This feature helps us to write computationally intensive codes in C/C++ and create a Python wrapper for it so that we can use these wrappers as Python modules. This gives us two advantages: first, our code is as fast as original C/C++ code (since it is the actual C++ code working in background) and second, it is very easy to code in Python. This is how OpenCV-Python works, it is a Python wrapper around original C++ implementation. And the support of Numpy makes the task more easier. Numpy is a highly optimized library for numerical operations. It gives a MATLAB-style syntax. All the OpenCV array structures are converted to-and-from Numpy arrays. So whatever operations you can do in Numpy, you can combine it with OpenCV, which increases number of weapons in your arsenal. Besides that, several other libraries like SciPy, Matplotlib which supports Numpy can be used with this. So OpenCV-Python is an appropriate tool for fast prototyping of computer vision problems.

Since OpenCV is an open source initiative, all are welcome to make contributions to this library. And it is same for this tutorial also. So, if you find any mistake in this tutorial (whether it be a small spelling mistake or a big error in code or concepts, whatever), feel free to correct it

And that will be a good task for freshers who begin to contribute to open source projects. Just fork the OpenCV in github, make necessary corrections and send a pull request to OpenCV. OpenCV developers will check your pull request, give you important feedback and once it passes the approval of the reviewer, it will be merged to OpenCV. Then you become a open source contributor. Similar is the case with other tutorials, documentation etc. As new modules are added to OpenCV-Python, this tutorial will have to be expanded. So those who knows about particular algorithm can write up a tutorial which includes a basic theory of the algorithm and a code showing basic usage of the algorithm and submit it to OpenCV. Remember, we together can make this project a great success !!!

**ARRAY ATTRIBUTES**

Array attributes reflect information that is intrinsic to the array itself. Generally, accessing an array through its attributes allows you to get and sometimes set intrinsic properties of the array without creating a new array. The exposed attributes are the core parts of an array and only some of them can be reset meaningfully without creating a new array.

For a 1-D array, this has no effect. (To change between column and row vectors, first cast the 1-D array into a matrix object.) For a 2-D array, this is the usual matrix transpose. For an n-D array, if axes are given, their order indicates how the axes are permuted (see Examples). If axes are not provided and a.shape = (i[0], i[1], ... i[n-2], i[n-1]), then a.transpose().shape = (i[n-1], i[n-2], ... i[1], i[0]).

**SCALARS**

Python defines only one type of a particular data class (there is only one integer type, one floating-point type, etc.). This can be convenient in applications that don’t need to be concerned with all the ways data can be represented in a computer. For scientific computing, however, more control is often needed. In NumPy, there are 24 new fundamental Python types to describe different types of scalars. These type descriptors are mostly based on the types available in the C language that CPython is written in, with several additional types compatible with Python’s types.

**METHODS**

Array scalars have exactly the same methods as arrays. The default behavior of these methods is to internally convert the scalar to an equivalent 0-dimensional array and to call the corresponding array method. In addition, math operations on array scalars are defined so that the same hardware flags are set and used to interpret the results as for ufunc, so that the error state used for ufuncs also carries over to the math on array scalars.

**DATA TYPE OBJECTS (DTYPE)**

A data type object (an instance of numpy.dtype class) describes how the bytes in the fixed-size block of memory corresponding to an array item should be interpreted. It describes the following aspects of the data: 1. Type of the data (integer, float, Python object, etc.) 2. Size of the data (how many bytes is in e.g. the integer) 3. Byte order of the data (little-endian or big-endian) 4. If the data type is structured, an aggregate of other data types, (e.g., describing an array item consisting of an integer and a float), (a) what are the names of the “fields” of the structure, by which they can be accessed, (b) what is the data-type of each field, and (c) which part of the memory block each field takes. 5. If the data type is a sub-array, what is its shape and data type. To describe the type of scalar data, there are several built-in scalar types in Numpy for various precision of integers, floating-point numbers, etc. An item extracted from an array, e.g., by indexing, will be a Python object whose type is the scalar type associated with the data type of the array. Note that the scalar types are not dtype objects, even though they can be used in place of one whenever a data type specification is needed in Numpy. Structured data types are formed by creating a data type whose fields contain other data types. Each field has a name by which it can be accessed. The parent data type should be of sufficient size to contain all its fields; the parent is nearly always based on the void type which allows an arbitrary item size. Structured data types may also contain nested structured sub-array data types in their fields. Finally, a data type can describe items that are themselves arrays of items of another data type. These sub-arrays must, however, be of a fixed size. If an array is created using a data-type describing a sub-array, the dimensions of the sub-array are appended to the shape of the array when the array is created. Sub-arrays in a field of a structured type behave differently, see Field Access. Sub-arrays always have a C-contiguous memory layout.

**PYTHON**

Python is an [interpreted](https://en.wikipedia.org/wiki/Interpreted_language) [high-level programming language](https://en.wikipedia.org/wiki/High-level_programming_language) for programming Python offers multiple options for developing GUI (Graphical User Interface). Out of all the GUI methods, tkinter is most commonly used method. It is a standard Python interface to the Tk GUI toolkit shipped with Python. Python with tkinter outputs the fastest and easiest way to create the GUI applications. Creating a GUI using tkinter is an easy task.

**PYTHON FEATURES:**

Python features a [dynamic type](https://en.wikipedia.org/wiki/Dynamic_type) system and automatic [memory management](https://en.wikipedia.org/wiki/Memory_management). It supports multiple programming paradigms, including object-oriented , [imperative](https://en.wikipedia.org/wiki/Imperative_programming), [functional](https://en.wikipedia.org/wiki/Functional_programming) and [procedural](https://en.wikipedia.org/wiki/Procedural_programming), and has a large and comprehensive library. Python is a [multi-paradigm programming language](https://en.wikipedia.org/wiki/Multi-paradigm_programming_language). [Object-oriented programming](https://en.wikipedia.org/wiki/Object-oriented_programming) and [structured programming](https://en.wikipedia.org/wiki/Structured_programming) are fully supported, and many of its features support functional programming and aspect [-oriented programming](https://en.wikipedia.org/wiki/Aspect-oriented_programming) (including by meta [programming](https://en.wikipedia.org/wiki/Metaprogramming) and [meta objects](https://en.wikipedia.org/wiki/Metaobject" \o "Metaobject) (magic methods)). Many other paradigms are supported via extensions, including [design by contract](https://en.wikipedia.org/wiki/Design_by_contract) and [logic programming](https://en.wikipedia.org/wiki/Logic_programming).

**PYTHON LIBRARIES**

Python's large [standard library](https://en.wikipedia.org/wiki/Standard_library), commonly cited as one of its greatest strengths, provides tools suited too many tasks. For Internet-facing applications, many standard formats and protocols such as [MIME](https://en.wikipedia.org/wiki/MIME) and [HTTP](https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol) are supported. It

includes modules for creating [graphical user interfaces](https://en.wikipedia.org/wiki/Graphical_user_interface), connecting to [relational databases](https://en.wikipedia.org/wiki/Relational_database), [generating pseudorandom numbers](https://en.wikipedia.org/wiki/Pseudorandom_number_generator), arithmetic with arbitrary precision decimals, manipulating [regular expressions](https://en.wikipedia.org/wiki/Regular_expression), and [unit testin](https://en.wikipedia.org/wiki/Unit_testing)g.

As of March 2018, the [Python Package Index](https://en.wikipedia.org/wiki/Python_Package_Index) (PyPI), the official repository for third-party Python software, contains over 130,000 packages with a wide range of functionality, including:

* Graphical user interfaces
* Web frameworks
* Multimedia
* Databases
* Networking
* Test frameworks
* Automation
* Web scraping
* Documentation
* System administration
* Scientific computing
* Text processing
* Image processing

**DEEP LEARNING:**

Deep learning is a subset of [machine learning](https://www.ibm.com/cloud/learn/machine-learning), which is essentially a neural network with three or more layers. These neural networks attempt to simulate the behavior of the human brain—albeit far from matching its ability—allowing it to “learn” from large amounts of data. While a neural network with a single layer can still make approximate predictions, additional hidden layers can help to optimize and refine for accuracy.Deep learning drives many [artificial intelligence (AI)](https://www.ibm.com/cloud/learn/what-is-artificial-intelligence) applications and services that improve automation, performing analytical and physical tasks without human intervention.

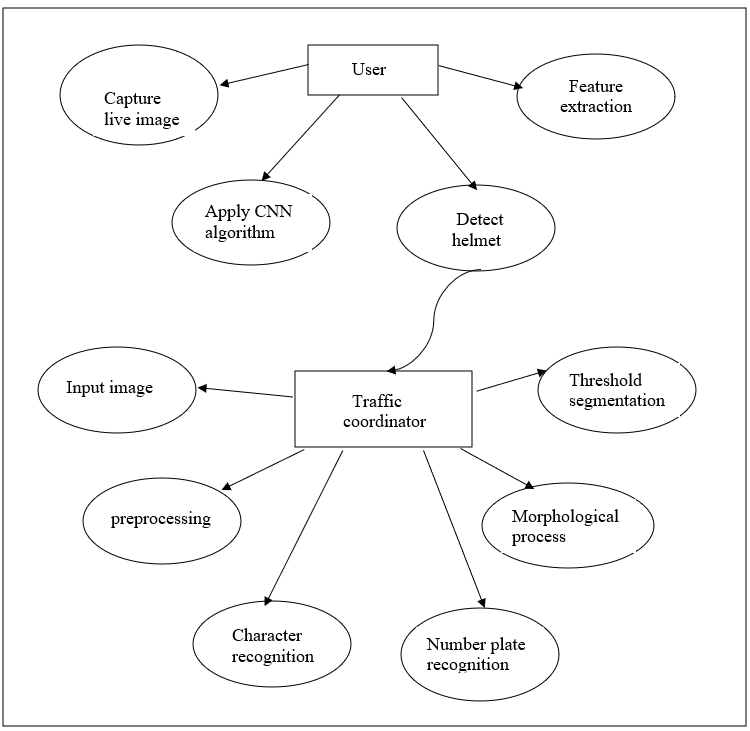
**CONVOLUTION NEURAL NETWORK:**

In [deep learning](https://en.wikipedia.org/wiki/Deep_learning), a convolutional neural network (CNN, or ConvNet) is a class of [deep neural network](https://en.wikipedia.org/wiki/Deep_neural_network), most commonly applied to analyze visual imagery. They are also known as shift invariant or space invariant artificial neural networks (SIANN), based on the shared-weight architecture of the convolution kernels or filters that slide along input features and provide translation [equivariant](https://en.wikipedia.org/wiki/Equivariant_map) responses known as feature maps. Counter-intuitively, most convolutional neural networks are only [equivariant](https://en.wikipedia.org/wiki/Equivariant_map), as opposed to [invariant](https://en.wikipedia.org/wiki/Translation_invariant), to translation. They have applications in [image and video recognition](https://en.wikipedia.org/wiki/Computer_vision), [recommender systems](https://en.wikipedia.org/wiki/Recommender_system), [image classification](https://en.wikipedia.org/wiki/Image_classification), [image segmentation](https://en.wikipedia.org/wiki/Image_segmentation), [medical image analysis](https://en.wikipedia.org/wiki/Medical_image_computing), [natural language processing](https://en.wikipedia.org/wiki/Natural_language_processing), [brain-computer interfaces](https://en.wikipedia.org/wiki/Brain%E2%80%93computer_interface), and financial [time series](https://en.wikipedia.org/wiki/Time_series).

**4. SYSTEM DESIGN**

* 1. **DATA FLOW DIAGRAM**

A data-flow diagram is a way of representing a flow of data through a process or a system. The DFD also provides information about the outputs and inputs of each entity and the process itself. A data-flow diagram has no control flow, there are no decision rules and no loops. Specific operations based on the data can be represented by a flowchart.

****

4.1 Dataflow diagram for helmet and license plate detection (Level 1)

**4.2 UML DIAGRAMS**

Unified Modeling Language (UML) is a standardized general-purpose modeling language in the field of software engineering. The standard is managed and was created by the Object Management Group. UML includes a set of graphic notation techniques to create visual models of software intensive systems. This language is used to specify, visualize, modify, construct and document the artifacts of an object oriented software intensive system under development.

* + 1. **USE CASE**

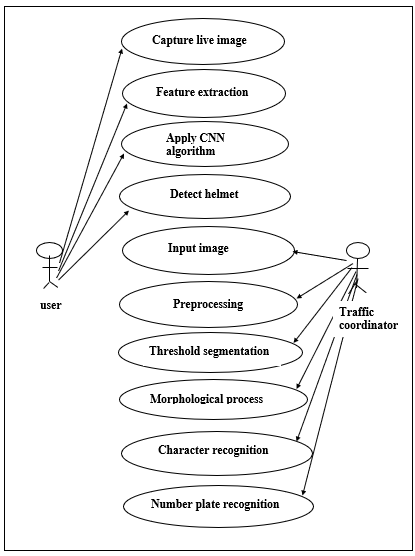
A use case is a set of scenarios that describing an interaction between a user and a system. A use case diagram displays the relationship among actors and use cases.A Use case Diagram is used to present a graphical overview of the functionality provided by a system in terms of actors, their goals and any dependencies between those use cases.

Use case diagram consists of two parts:

**Use case:** A use case describes a sequence of actions that provided something of measurable value to an actor and is drawn as a horizontal ellipse.

**Actor:** An actor is a person, organization or external system that plays a role in one or more interaction with the system.

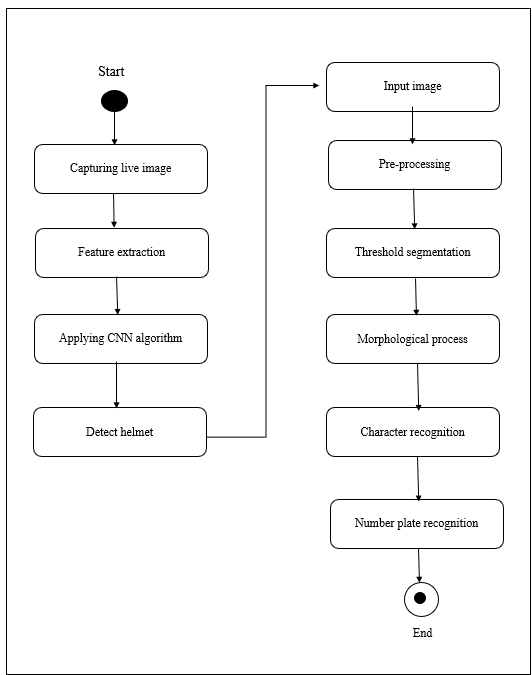
**Communication Link:** The participation of an actor in a use case is shown by connecting an actor to a use case by a solid link. Actors may be connected to use cases by associations, indicating that the actor and the use case communicate with one another using message.



4.2.1 Use case diagram for helmet and license plate detection

* + 1. **ACTIVITY DIAGRAM**

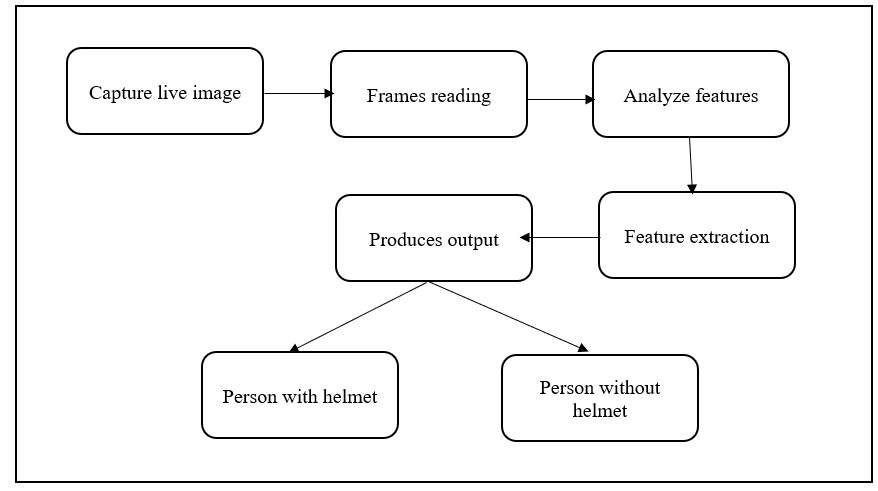
Activity diagram is a graphical representation of workflows of stepwise activities and actions with support for choice, iteration and concurrency. An activity diagram shows the overall flow of control.



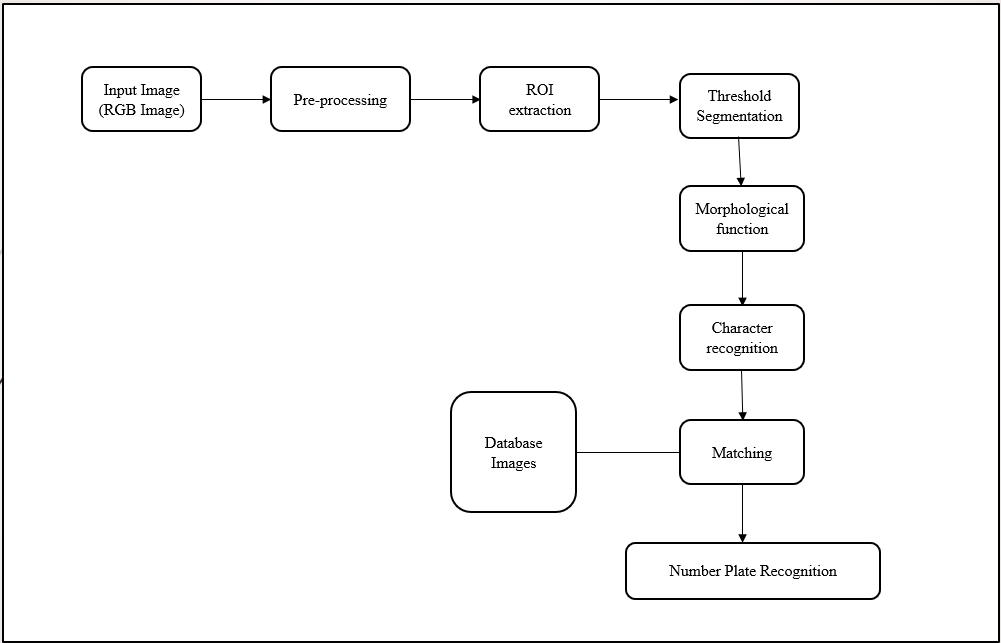
4.2.2 Activity diagram for helmet and license plate detection

1. **SYSTEM ARCHITECTURE**
   1. **ARCHITECTURE OVERVIEW**

This is a helmet and number plate detection system in which we get an input images and videos from traffic surveillance system which is given as a input to convolutional neural networks and process accurate output whether the person is wearing helmet or not .In the next process if the person is not wearing helmet then number plate detection will be done using convolutional neural networks and determines number plate characters of bike rider accurately and produces effective output .



5.1.1 Architecture diagram for helmet detection



5.1.2 Architecture diagram for license plate detection

**5.2 MODULE DESIGN SPECIFICATION**

**HELMET DETECTION:**

1.Input Video

2.Image Classification

3.CNN Classifier

4.Result Interpretation

**LICENSE PLATE DETECTION:**

1.Segmentation

2.Connected Component Analysis

3.Template Matching

4.Number Plate Recognition

**HELMET DETECTION**

* 1. **INPUT VIDEO**

The input video has been captured by using either ipcam or webcam, From this the bike is detected. This methods to detect the photo of motorcycle and driver from the image and then detect an area of the biker head before classify that this person is wearing a helmet or not.

* 1. **IMAGE CLASSIFICATION**

After gathering images for our training dataset, we split our images into two groups, one for training data and another for test data to use in classification experiment. This experiment we test them with CNN models for image classification. All videos will be tested and calculated the accuracy of the biker with helmet and no helmet detection in the video.

* 1. **CNN CLASSIFIER**

A CNN is a neural network with some convolutional layers (and some other layers). A convolutional layer has a number of filters that does convolutional operation. CNN can take in an input image, assign importance to various objects in the image and able to differentiate one from the other.

**4.RESULT INTERPRETATION**

The accuracy of the experiments will show the performance of each technique in terms of image classification and image detection. Deep Learning or CNN techniques are the good algorithms that we can apply on the problem of image detection and classification about bikers wearing a helmet or no helmet problem.

**LICENSE PLATE DETECTION**

**1.SEGMENTATION**

The goal of image segmentation is to cluster pixels intosalient image regions, i.e., regions corresponding to individual surfaces, objects, or natural parts of objects. In computer vision segmentation refers to the process of partitioning a digital image to multiple segment. The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier. The result of image segmentation is a set of segments that collectively cover the entire image, or a set of contours extracted from the image.

1. **CONNECTED COMPONENT ANALYSIS**

In order to find the objects in an image, we want to employ an operation that is called Connected Component Analysis (CCA). This operation takes a binary image as an input. Usually, the False value in this image is associated with background pixels, and the True value indicates foreground, or object pixels. Such an image can be e.g. produced with thresholding. Given a thresholded image, CCA produces a new *labeled* image with integer pixel values. Pixels with the same value, belong to the same object.

1. **TEMPLATE MATCHING**

Template matching refers to the image processing where we find similar templates in a source image by giving a base template to compared on.The process of template matching is done by comparing each of the pixel values of the source image one at a time to the template image. The output would be an array of similarity values when compared to the template image.

1. **NUMBER PLATE RECOGNITION**

After applying above processes, we will get the foreground and background separated output , From this number plate is extracted.

1. **SYSTEM IMPLEMENTATION**

**helmet\_detection.py:**

from time import sleep

from utils import postprocess

import cv2 as cv

frame\_count = 0

frame\_count\_out=0

confThreshold = 0.5

nmsThreshold = 0.4

inpWidth = 416

inpHeight = 416

classesFile = "obj.names"

with open(classesFile, 'rt') as f:

classes = f.read().rstrip('\n').split('\n').

modelConfiguration = "yolov3-obj.cfg";

modelWeights = "yolov3-obj\_2400.weights";

net = cv.dnn.readNetFromDarknet(modelConfiguration, modelWeights)

net.setPreferableBackend(cv.dnn.DNN\_BACKEND\_OPENCV)

net.setPreferableTarget(cv.dnn.DNN\_TARGET\_CPU)

layersNames = net.getLayerNames()

output\_layer = [layersNames[i[0] - 1] for i in net.getUnconnectedOutLayers()]

cap = cv.VideoCapture(0)

while True:

ret, frame = cap.read()

if cv.waitKey(1) & 0xFF == ord('q'):

break

blob = cv.dnn.blobFromImage(frame, 1/255, (inpWidth, inpHeight), [0,0,0], 1, crop=False)

net.setInput(blob)

outs = net.forward(output\_layer)

postprocess(frame, outs, confThreshold, nmsThreshold, classes)

cv.imshow('img', frame)

t, \_ = net.getPerfProfile()

label = 'Inference time: %.2f ms' % (t \* 1000.0 / cv.getTickFrequency())

cv.putText(frame, label, (0, 15), cv.FONT\_HERSHEY\_SIMPLEX, 0.5, (0, 0, 255))

**Utils.py:**

import cv2 as cv

import numpy as np

def draw\_bounding\_box(classId, conf, left, top, right, bottom, frame, classes):

frame\_count = 0

cv.rectangle(frame, (left, top), (right, bottom), (255, 178, 50), 3)

label = '%.2f' % conf

if classes:

assert(classId < len(classes))

label = '%s:%s' % (classes[classId], label)

label\_size, base\_line = cv.getTextSize(label, cv.FONT\_HERSHEY\_SIMPLEX, 0.5, 1)

top = max(top, label\_size[1])

label\_name,label\_conf = label.split(':')

if label\_name == 'Helmet':

cv.rectangle(frame, (left, top - round(1.5\*label\_size[1])), (left + round(1.5\*label\_size[0]), top + base\_line),

(255, 255, 255), cv.FILLED)

cv.putText(frame, label, (left, top), cv.FONT\_HERSHEY\_SIMPLEX, 0.75, (0,0,0), 1)

frame\_count+=1

def postprocess(frame, outs, conf\_threshold, nms\_threshold, classes):

frameHeight = frame.shape[0]

frame\_width = frame.shape[1]

classIds = []

confidences = []

boxes = []

for out in outs:

for detection in out:

scores = detection[5:]

classId = np.argmax(scores)

confidence = scores[classId]

if confidence > conf\_threshold:

center\_x = int(detection[0] \* frame\_width)

center\_y = int(detection[1] \* frameHeight)

width = int(detection[2] \* frame\_width)

height = int(detection[3] \* frameHeight)

left = int(center\_x - width / 2)

top = int(center\_y - height / 2)

classIds.append(classId)

confidences.append(float(confidence))

boxes.append([left, top, width, height])

indices = cv.dnn.NMSBoxes(boxes, confidences, conf\_threshold, nms\_threshold)

count\_person=0 # for counting the classes in this loop.

for i in indices:

i = i[0]

box = boxes[i]

left = box[0]

top = box[1]

width = box[2]

height = box[3]

draw\_bounding\_box(classIds[i], confidences[i], left, top, left + width, top + height, frame, classes)

my\_class='Helmet'

unknown\_class = classes[classId]

if my\_class == unknown\_class:

count\_person += 1

if(count\_person > 0):

print('Helmet detected')

else:

print('No helmet')

**Main.py:**

import cv2

import numpy as np

import os

import DetectChars

import DetectPlates

import PossiblePlate

SCALAR\_BLACK = (0.0, 0.0, 0.0)

SCALAR\_WHITE = (255.0, 255.0, 255.0)

SCALAR\_YELLOW = (0.0, 255.0, 255.0)

SCALAR\_GREEN = (0.0, 255.0, 0.0)

SCALAR\_RED = (0.0, 0.0, 255.0)

showSteps = False

def main():

blnKNNTrainingSuccessful = DetectChars.loadKNNDataAndTrainKNN()

if blnKNNTrainingSuccessful == False:

print("\nerror: KNN traning was not successful\n")

return

imgOriginalScene = cv2.imread("1.png

if imgOriginalScene is None:

print("\nerror: image not read from file \n\n")

os.system("pause")

return

listOfPossiblePlates = DetectPlates.detectPlatesInScene(imgOriginalScene)

listOfPossiblePlates = DetectChars.detectCharsInPlates(listOfPossiblePlates)

cv2.imshow("imgOriginalScene", imgOriginalScene)

if len(listOfPossiblePlates) == 0:

print("\nno license plates were detected\n")

else:

listOfPossiblePlates.sort(key = lambda possiblePlate: len(possiblePlate.strChars), reverse = True)

licPlate = listOfPossiblePlates[0]

cv2.imshow("imgPlate", licPlate.imgPlate)

cv2.imshow("imgThresh", licPlate.imgThresh)

if len(licPlate.strChars) == 0:

print("\nno characters were detected\n\n")

return

drawRedRectangleAroundPlate(imgOriginalScene, licPlate)

print("\nlicense plate read from image = " + licPlate.strChars + "\n")

print("----------------------------------------")

writeLicensePlateCharsOnImage(imgOriginalScene, licPlate)

cv2.imshow("imgOriginalScene", imgOriginalScene)

cv2.imwrite("imgOriginalScene.png", imgOriginalScene)

cv2.waitKey(0)

return

def drawRedRectangleAroundPlate(imgOriginalScene, licPlate):

p2fRectPoints = cv2.boxPoints(licPlate.rrLocationOfPlateInScene)

cv2.line(imgOriginalScene, tuple(p2fRectPoints[0]), tuple(p2fRectPoints[1]), SCALAR\_RED, 2)

cv2.line(imgOriginalScene, tuple(p2fRectPoints[1]), tuple(p2fRectPoints[2]), SCALAR\_RED, 2)

cv2.line(imgOriginalScene, tuple(p2fRectPoints[2]), tuple(p2fRectPoints[3]), SCALAR\_RED, 2)

cv2.line(imgOriginalScene, tuple(p2fRectPoints[3]), tuple(p2fRectPoints[0]), SCALAR\_RED, 2)

def writeLicensePlateCharsOnImage(imgOriginalScene, licPlate):

ptCenterOfTextAreaX = 0

ptCenterOfTextAreaY = 0

ptLowerLeftTextOriginX = 0

ptLowerLeftTextOriginY = 0

sceneHeight, sceneWidth, sceneNumChannels = imgOriginalScene.shape

plateHeight, plateWidth, plateNumChannels = licPlate.imgPlate.shape

intFontFace = cv2.FONT\_HERSHEY\_SIMPLEX

fltFontScale = float(plateHeight) / 30.0

intFontThickness = int(round(fltFontScale \* 1.5))

textSize, baseline = cv2.getTextSize(licPlate.strChars, intFontFace, fltFontScale, intFontThickness)

fltCorrectionAngleInDeg ) = licPlate.rrLocationOfPlateInScene

intPlateCenterX = int(intPlateCenterX)

intPlateCenterY = int(intPlateCenterY)

ptCenterOfTextAreaX = int(intPlateCenterX)

if intPlateCenterY < (sceneHeight \* 0.75):

ptCenterOfTextAreaY = int(round(intPlateCenterY)) + int(round(plateHeight \* 1.6))

else:

ptCenterOfTextAreaY = int(round(intPlateCenterY)) - int(round(plateHeight \* 1.6))

textSizeWidth, textSizeHeight = textSize

ptLowerLeftTextOriginX = int(ptCenterOfTextAreaX - (textSizeWidth / 2))

ptLowerLeftTextOriginY = int(ptCenterOfTextAreaY + (textSizeHeight / 2))

cv2.putText(imgOriginalScene, licPlate.strChars, (ptLowerLeftTextOriginX, ptLowerLeftTextOriginY), intFontFace, fltFontScale, SCALAR\_YELLOW, intFontThickness)

if \_\_name\_\_ == "\_\_main\_\_":

main()

**DetectChars.py:**

import os

import cv2

import numpy as np

import math

import random

import Main

import Preprocess

import PossibleChar

kNearest = cv2.ml.KNearest\_create()

MIN\_PIXEL\_WIDTH = 2

MIN\_PIXEL\_HEIGHT = 8

MIN\_ASPECT\_RATIO = 0.25

MAX\_ASPECT\_RATIO = 1.0

MIN\_PIXEL\_AREA = 80

MIN\_DIAG\_SIZE\_MULTIPLE\_AWAY = 0.3

MAX\_DIAG\_SIZE\_MULTIPLE\_AWAY = 5.0

MAX\_CHANGE\_IN\_AREA = 0.5

MAX\_CHANGE\_IN\_WIDTH = 0.8

MAX\_CHANGE\_IN\_HEIGHT = 0.2

MAX\_ANGLE\_BETWEEN\_CHARS = 12.0

MIN\_NUMBER\_OF\_MATCHING\_CHARS = 3

RESIZED\_CHAR\_IMAGE\_WIDTH = 20

RESIZED\_CHAR\_IMAGE\_HEIGHT = 30

MIN\_CONTOUR\_AREA = 100

def loadKNNDataAndTrainKNN():

allContoursWithData = [] # declare empty lists,

validContoursWithData = [] # we will fill these shortly

try:

npaClassifications = np.loadtxt("classifications.txt", np.float32)

except:

print("error, unable to open classifications.txt, exiting program\n")

os.system("pause")

return False

try:

npaFlattenedImages = np.loadtxt("flattened\_images.txt", np.float32)

except:

print("error, unable to open flattened\_images.txt, exiting program\n")

os.system("pause")

return False

npaClassifications = npaClassifications.reshape((npaClassifications.size, 1))

kNearest.setDefaultK(1)

kNearest.train(npaFlattenedImages, cv2.ml.ROW\_SAMPLE, npaClassifications)

return True

def detectCharsInPlates(listOfPossiblePlates):

intPlateCounter = 0

imgContours = None

contours = []

if len(listOfPossiblePlates) == 0:

return listOfPossiblePlates

for possiblePlate in listOfPossiblePlates:

possiblePlate.imgGrayscale, possiblePlate.imgThresh = Preprocess.preprocess(possiblePlate.imgPlate)

if Main.showSteps == True:

cv2.imshow("5a", possiblePlate.imgPlate)

cv2.imshow("5b", possiblePlate.imgGrayscale)

cv2.imshow("5c", possiblePlate.imgThresh)

possiblePlate.imgThresh = cv2.resize(possiblePlate.imgThresh, (0, 0), fx = 1.6, fy = 1.6)

thresholdValue, possiblePlate.imgThresh = cv2.threshold(possiblePlate.imgThresh, 0.0, 255.0, cv2.THRESH\_BINARY | cv2.THRESH\_OTSU)

if Main.showSteps == True:

cv2.imshow("5d", possiblePlate.imgThresh)

be chars (without comparison to other chars yet)

listOfPossibleCharsInPlate = findPossibleCharsInPlate(possiblePlate.imgGrayscale, possiblePlate.imgThresh)

if Main.showSteps == True:

height, width, numChannels = possiblePlate.imgPlate.shape

imgContours = np.zeros((height, width, 3), np.uint8)

del contours[:]

for possibleChar in listOfPossibleCharsInPlate:

contours.append(possibleChar.contour)

cv2.drawContours(imgContours, contours, -1, Main.SCALAR\_WHITE)

cv2.imshow("6", imgContours)

listOfListsOfMatchingCharsInPlate = findListOfListsOfMatchingChars(listOfPossibleCharsInPlate)

if Main.showSteps == True:

imgContours = np.zeros((height, width, 3), np.uint8)

del contours[:]

for listOfMatchingChars in listOfListsOfMatchingCharsInPlate:

intRandomBlue = random.randint(0, 255)

intRandomGreen = random.randint(0, 255)

intRandomRed = random.randint(0, 255)

for matchingChar in listOfMatchingChars:

contours.append(matchingChar.contour)

cv2.drawContours(imgContours, contours, -1, (intRandomBlue, intRandomGreen, intRandomRed))

cv2.imshow("7", imgContours)

if (len(listOfListsOfMatchingCharsInPlate) == 0):

if Main.showSteps == True:

print("chars found in plate number " + str(

intPlateCounter) + " = (none), click on any image and press a key to continue . . .")

intPlateCounter = intPlateCounter + 1

cv2.destroyWindow("8")

cv2.destroyWindow("9")

cv2.destroyWindow("10")

cv2.waitKey(0)

possiblePlate.strChars = ""

continue

for i in range(0, len(listOfListsOfMatchingCharsInPlate)): # within each list of matching chars

listOfListsOfMatchingCharsInPlate[i].sort(key = lambda matchingChar: matchingChar.intCenterX) # sort chars from left to right

listOfListsOfMatchingCharsInPlate[i] = removeInnerOverlappingChars(listOfListsOfMatchingCharsInPlate[i])

if Main.showSteps == True:

imgContours = np.zeros((height, width, 3), np.uint8)

for listOfMatchingChars in listOfListsOfMatchingCharsInPlate:

intRandomBlue = random.randint(0, 255)

intRandomGreen = random.randint(0, 255)

intRandomRed = random.randint(0, 255)

del contours[:]

for matchingChar in listOfMatchingChars:

contours.append(matchingChar.contour)

cv2.drawContours(imgContours, contours, -1, (intRandomBlue, intRandomGreen, intRandomRed))

cv2.imshow("8", imgContours)

intLenOfLongestListOfChars = 0

intIndexOfLongestListOfChars = 0

for i in range(0, len(listOfListsOfMatchingCharsInPlate)):

if len(listOfListsOfMatchingCharsInPlate[i]) > intLenOfLongestListOfChars:

intLenOfLongestListOfChars = len(listOfListsOfMatchingCharsInPlate[i])

intIndexOfLongestListOfChars = i

longestListOfMatchingCharsInPlate = listOfListsOfMatchingCharsInPlate[intIndexOfLongestListOfChars]

if Main.showSteps == True:

imgContours = np.zeros((height, width, 3), np.uint8)

del contours[:]

for matchingChar in longestListOfMatchingCharsInPlate:

contours.append(matchingChar.contour)

cv2.drawContours(imgContours, contours, -1, Main.SCALAR\_WHITE)

cv2.imshow("9", imgContours)

possiblePlate.strChars = recognizeCharsInPlate(possiblePlate.imgThresh, longestListOfMatchingCharsInPlate)

if Main.showSteps == True:

print("chars found in plate number " + str(

intPlateCounter) + " = " + possiblePlate.strChars + ", click on any image and press a key to continue . . .")

intPlateCounter = intPlateCounter + 1

cv2.waitKey(0)

if Main.showSteps == True:

print("\nchar detection complete, click on any image and press a key to continue . . .\n")

cv2.waitKey(0)

return listOfPossiblePlates

def findPossibleCharsInPlate(imgGrayscale, imgThresh):

listOfPossibleChars = []

contours = []

imgThreshCopy = imgThresh.copy()

contours, npaHierarchy = cv2.findContours(imgThreshCopy, cv2.RETR\_LIST, cv2.CHAIN\_APPROX\_SIMPLE)

for contour in contours:

possibleChar = PossibleChar.PossibleChar(contour)

if checkIfPossibleChar(possibleChar):

listOfPossibleChars.append(possibleChar)

return listOfPossibleChars

def checkIfPossibleChar(possibleChar):

if (possibleChar.intBoundingRectArea > MIN\_PIXEL\_AREA and

possibleChar.intBoundingRectWidth > MIN\_PIXEL\_WIDTH and possibleChar.intBoundingRectHeight > MIN\_PIXEL\_HEIGHT and

MIN\_ASPECT\_RATIO < possibleChar.fltAspectRatio and possibleChar.fltAspectRatio < MAX\_ASPECT\_RATIO):

return True

else:

return False

def findListOfListsOfMatchingChars(listOfPossibleChars):

listOfListsOfMatchingChars = []

for possibleChar in listOfPossibleChars:

listOfMatchingChars = findListOfMatchingChars(possibleChar, listOfPossibleChars)

listOfMatchingChars.append(possibleChar)

if len(listOfMatchingChars) < MIN\_NUMBER\_OF\_MATCHING\_CHARS:

continue

listOfListsOfMatchingChars.append(listOfMatchingChars)

listOfPossibleCharsWithCurrentMatchesRemoved = []

listOfPossibleCharsWithCurrentMatchesRemoved = list(set(listOfPossibleChars) - set(listOfMatchingChars))

recursiveListOfListsOfMatchingChars = findListOfListsOfMatchingChars(listOfPossibleCharsWithCurrentMatchesRemoved)

for recursiveListOfMatchingChars in recursiveListOfListsOfMatchingChars:

listOfListsOfMatchingChars.append(recursiveListOfMatchingChars)

break

return listOfListsOfMatchingChars

def findListOfMatchingChars(possibleChar, listOfChars):

listOfMatchingChars = []

for possibleMatchingChar in listOfChars:

if possibleMatchingChar == possibleChar:

continue

fltDistanceBetweenChars = distanceBetweenChars(possibleChar, possibleMatchingChar)

fltAngleBetweenChars = angleBetweenChars(possibleChar, possibleMatchingChar)

fltChangeInArea = float(abs(possibleMatchingChar.intBoundingRectArea - possibleChar.intBoundingRectArea)) / float(possibleChar.intBoundingRectArea)

fltChangeInWidth = float(abs(possibleMatchingChar.intBoundingRectWidth - possibleChar.intBoundingRectWidth)) / float(possibleChar.intBoundingRectWidth)

fltChangeInHeight = float(abs(possibleMatchingChar.intBoundingRectHeight - possibleChar.intBoundingRectHeight)) / float(possibleChar.intBoundingRectHeight)

if (fltDistanceBetweenChars < (possibleChar.fltDiagonalSize \* MAX\_DIAG\_SIZE\_MULTIPLE\_AWAY) and

fltAngleBetweenChars < MAX\_ANGLE\_BETWEEN\_CHARS and

fltChangeInArea < MAX\_CHANGE\_IN\_AREA and

fltChangeInWidth < MAX\_CHANGE\_IN\_WIDTH and

fltChangeInHeight < MAX\_CHANGE\_IN\_HEIGHT):

listOfMatchingChars.append(possibleMatchingChar) # if the chars are a match, add the current char to list of matching chars

return listOfMatchingChars

def distanceBetweenChars(firstChar, secondChar):

intX = abs(firstChar.intCenterX - secondChar.intCenterX)

intY = abs(firstChar.intCenterY - secondChar.intCenterY)

return math.sqrt((intX \*\* 2) + (intY \*\* 2))

def angleBetweenChars(firstChar, secondChar):

fltAdj = float(abs(firstChar.intCenterX - secondChar.intCenterX))

fltOpp = float(abs(firstChar.intCenterY - secondChar.intCenterY))

if fltAdj != 0.0:

fltAngleInRad = math.atan(fltOpp / fltAdj)

else:

fltAngleInRad = 1.5708

fltAngleInDeg = fltAngleInRad \* (180.0 / math.pi) # calculate angle in degrees

return fltAngleInDeg

def removeInnerOverlappingChars(listOfMatchingChars):

listOfMatchingCharsWithInnerCharRemoved = list(listOfMatchingChars) # this will be the return value

for currentChar in listOfMatchingChars:

for otherChar in listOfMatchingChars:

if currentChar != otherChar:

if distanceBetweenChars(currentChar, otherChar) < (currentChar.fltDiagonalSize \* MIN\_DIAG\_SIZE\_MULTIPLE\_AWAY):

if currentChar.intBoundingRectArea < otherChar.intBoundingRectArea:

if currentChar in listOfMatchingCharsWithInnerCharRemoved:

listOfMatchingCharsWithInnerCharRemoved.remove(currentChar)

else:

if otherChar in listOfMatchingCharsWithInnerCharRemoved:

listOfMatchingCharsWithInnerCharRemoved.remove(otherChar)

return listOfMatchingCharsWithInnerCharRemoved

def recognizeCharsInPlate(imgThresh, listOfMatchingChars):

strChars = ""

height, width = imgThresh.shape

imgThreshColor = np.zeros((height, width, 3), np.uint8)

listOfMatchingChars.sort(key = lambda matchingChar: matchingChar.intCenterX)

cv2.cvtColor(imgThresh, cv2.COLOR\_GRAY2BGR, imgThreshColor)

for currentChar in listOfMatchingChars:

pt1 = (currentChar.intBoundingRectX, currentChar.intBoundingRectY)

pt2 = ((currentChar.intBoundingRectX + currentChar.intBoundingRectWidth), (currentChar.intBoundingRectY + currentChar.intBoundingRectHeight))

cv2.rectangle(imgThreshColor, pt1, pt2, Main.SCALAR\_GREEN, 2)

imgROI = imgThresh[currentChar.intBoundingRectY : currentChar.intBoundingRectY + currentChar.intBoundingRectHeight,

currentChar.intBoundingRectX : currentChar.intBoundingRectX + currentChar.intBoundingRectWidth]

imgROIResized = cv2.resize(imgROI, (RESIZED\_CHAR\_IMAGE\_WIDTH, RESIZED\_CHAR\_IMAGE\_HEIGHT)) # resize image, this is necessary for char recognition

npaROIResized = imgROIResized.reshape((1, RESIZED\_CHAR\_IMAGE\_WIDTH \* RESIZED\_CHAR\_IMAGE\_HEIGHT))

npaROIResized = np.float32(npaROIResized)

retval, npaResults, neigh\_resp, dists = kNearest.findNearest(npaROIResized, k = 1)

strCurrentChar = str(chr(int(npaResults[0][0])))

strChars = strChars + strCurrentChar

if Main.showSteps == True:

cv2.imshow("10", imgThreshColor)

return strChars

**DetectPlates.py:**

import cv2

import numpy as np

import math

import Main

import random

import Preprocess

import DetectChars

import PossiblePlate

import PossibleChar

PLATE\_WIDTH\_PADDING\_FACTOR = 1.3

PLATE\_HEIGHT\_PADDING\_FACTOR = 1.5

def detectPlatesInScene(imgOriginalScene):

listOfPossiblePlates = []

height, width, numChannels = imgOriginalScene.shape

imgGrayscaleScene = np.zeros((height, width, 1), np.uint8)

imgThreshScene = np.zeros((height, width, 1), np.uint8)

imgContours = np.zeros((height, width, 3), np.uint8)

cv2.destroyAllWindows()

if Main.showSteps == True:

cv2.imshow("0", imgOriginalScene)

imgGrayscaleScene, imgThreshScene = Preprocess.preprocess(imgOriginalScene)

if Main.showSteps == True:

cv2.imshow("1a", imgGrayscaleScene)

cv2.imshow("1b", imgThreshScene)

listOfPossibleCharsInScene = findPossibleCharsInScene(imgThreshScene)

if Main.showSteps == True:

print("step 2 - len(listOfPossibleCharsInScene) = " + str(

len(listOfPossibleCharsInScene)))

imgContours = np.zeros((height, width, 3), np.uint8)

contours = []

for possibleChar in listOfPossibleCharsInScene:

contours.append(possibleChar.contour)

cv2.drawContours(imgContours, contours, -1, Main.SCALAR\_WHITE)

cv2.imshow("2b", imgContours)

listOfListsOfMatchingCharsInScene = DetectChars.findListOfListsOfMatchingChars(listOfPossibleCharsInScene)

if Main.showSteps == True:

print("step 3 - listOfListsOfMatchingCharsInScene.Count = " + str(

len(listOfListsOfMatchingCharsInScene))) # 13 with MCLRNF1 image

imgContours = np.zeros((height, width, 3), np.uint8)

for listOfMatchingChars in listOfListsOfMatchingCharsInScene:

intRandomBlue = random.randint(0, 255)

intRandomGreen = random.randint(0, 255)

intRandomRed = random.randint(0, 255)

contours = []

for matchingChar in listOfMatchingChars:

contours.append(matchingChar.contour)

cv2.drawContours(imgContours, contours, -1, (intRandomBlue, intRandomGreen, intRandomRed))

cv2.imshow("3", imgContours)

for listOfMatchingChars in listOfListsOfMatchingCharsInScene:

possiblePlate = extractPlate(imgOriginalScene, listOfMatchingChars)

if possiblePlate.imgPlate is not None:

listOfPossiblePlates.append(possiblePlate)

print("\n" + str(len(listOfPossiblePlates)) + " possible plates found")

if Main.showSteps == True:

print("\n")

cv2.imshow("4a", imgContours)

for i in range(0, len(listOfPossiblePlates)):

p2fRectPoints = cv2.boxPoints(listOfPossiblePlates[i].rrLocationOfPlateInScene)

cv2.line(imgContours, tuple(p2fRectPoints[0]), tuple(p2fRectPoints[1]), Main.SCALAR\_RED, 2)

cv2.line(imgContours, tuple(p2fRectPoints[1]), tuple(p2fRectPoints[2]), Main.SCALAR\_RED, 2)

cv2.line(imgContours, tuple(p2fRectPoints[2]), tuple(p2fRectPoints[3]), Main.SCALAR\_RED, 2)

cv2.line(imgContours, tuple(p2fRectPoints[3]), tuple(p2fRectPoints[0]), Main.SCALAR\_RED, 2)

cv2.imshow("4a", imgContours)

print("possible plate " + str(i) + ", click on any image and press a key to continue . . .")

cv2.imshow("4b", listOfPossiblePlates[i].imgPlate)

cv2.waitKey(0)

print("\nplate detection complete, click on any image and press a key to begin char recognition . . .\n")

cv2.waitKey(0)

return listOfPossiblePlates

def findPossibleCharsInScene(imgThresh):

listOfPossibleChars = []

intCountOfPossibleChars = 0

imgThreshCopy = imgThresh.copy()

contours, npaHierarchy = cv2.findContours(imgThreshCopy, cv2.RETR\_LIST, cv2.CHAIN\_APPROX\_SIMPLE)

height, width = imgThresh.shape

imgContours = np.zeros((height, width, 3), np.uint8)

for i in range(0, len(contours)):

if Main.showSteps == True:

cv2.drawContours(imgContours, contours, i, Main.SCALAR\_WHITE)

possibleChar = PossibleChar.PossibleChar(contours[i])

if DetectChars.checkIfPossibleChar(possibleChar):

intCountOfPossibleChars = intCountOfPossibleChars + 1

listOfPossibleChars.append(possibleChar)

if Main.showSteps == True:

print("\nstep 2 - len(contours) = " + str(len(contours)))

print("step 2 - intCountOfPossibleChars = " + str(intCountOfPossibleChars)) # 131 with MCLRNF1 image

cv2.imshow("2a", imgContours)

return listOfPossibleChars

def extractPlate(imgOriginal, listOfMatchingChars):

possiblePlate = PossiblePlate.PossiblePlate()

listOfMatchingChars.sort(key = lambda matchingChar: matchingChar.intCenterX)

fltPlateCenterX = (listOfMatchingChars[0].intCenterX + listOfMatchingChars[len(listOfMatchingChars) - 1].intCenterX) / 2.0

fltPlateCenterY = (listOfMatchingChars[0].intCenterY + listOfMatchingChars[len(listOfMatchingChars) - 1].intCenterY) / 2.0

ptPlateCenter = fltPlateCenterX, fltPlateCenterY

intPlateWidth = int((listOfMatchingChars[len(listOfMatchingChars) - 1].intBoundingRectX + listOfMatchingChars[len(listOfMatchingChars) - 1].intBoundingRectWidth - listOfMatchingChars[0].intBoundingRectX) \* PLATE\_WIDTH\_PADDING\_FACTOR)

intTotalOfCharHeights = 0

for matchingChar in listOfMatchingChars:

intTotalOfCharHeights = intTotalOfCharHeights + matchingChar.intBoundingRectHeight

fltAverageCharHeight = intTotalOfCharHeights / len(listOfMatchingChars)

intPlateHeight = int(fltAverageCharHeight \* PLATE\_HEIGHT\_PADDING\_FACTOR)

fltOpposite = listOfMatchingChars[len(listOfMatchingChars) - 1].intCenterY - listOfMatchingChars[0].intCenterY

fltHypotenuse = DetectChars.distanceBetweenChars(listOfMatchingChars[0], listOfMatchingChars[len(listOfMatchingChars) - 1])

fltCorrectionAngleInRad = math.asin(fltOpposite / fltHypotenuse)

fltCorrectionAngleInDeg = fltCorrectionAngleInRad \* (180.0 / math.pi)

possiblePlate.rrLocationOfPlateInScene = ( tuple(ptPlateCenter), (intPlateWidth, intPlateHeight), fltCorrectionAngleInDeg )

rotationMatrix = cv2.getRotationMatrix2D(tuple(ptPlateCenter), fltCorrectionAngleInDeg, 1.0)

height, width, numChannels = imgOriginal.shape

imgRotated = cv2.warpAffine(imgOriginal, rotationMatrix, (width, height))

imgCropped = cv2.getRectSubPix(imgRotated, (intPlateWidth, intPlateHeight), tuple(ptPlateCenter))

possiblePlate.imgPlate = imgCropped

return possiblePlate

**PossibleChar.py:**

import cv2

import numpy as np

import math

class PossibleChar:

def \_\_init\_\_(self, \_contour):

self.contour = \_contour

self.boundingRect = cv2.boundingRect(self.contour)

[intX, intY, intWidth, intHeight] = self.boundingRect

self.intBoundingRectX = intX

self.intBoundingRectY = intY

self.intBoundingRectWidth = intWidth

self.intBoundingRectHeight = intHeight

self.intBoundingRectArea = self.intBoundingRectWidth \* self.intBoundingRectHeight

self.intCenterX = (self.intBoundingRectX + self.intBoundingRectX + self.intBoundingRectWidth) / 2

self.intCenterY = (self.intBoundingRectY + self.intBoundingRectY + self.intBoundingRectHeight) / 2

self.fltDiagonalSize = math.sqrt((self.intBoundingRectWidth \*\* 2) + (self.intBoundingRectHeight \*\* 2))

self.fltAspectRatio = float(self.intBoundingRectWidth) / float(self.intBoundingRectHeight)

**PossiblePlate.py**:

import cv2

import numpy as np

class PossiblePlate:

def \_\_init\_\_(self):

self.imgPlate = None

self.imgGrayscale = None

self.imgThresh = None

self.rrLocationOfPlateInScene = None

self.strChars = ""

**Preprocess.py:**

import cv2

import numpy as np

import math

GAUSSIAN\_SMOOTH\_FILTER\_SIZE = (5, 5)

ADAPTIVE\_THRESH\_BLOCK\_SIZE = 19

ADAPTIVE\_THRESH\_WEIGHT = 9

def preprocess(imgOriginal):

imgGrayscale = extractValue(imgOriginal)

imgMaxContrastGrayscale = maximizeContrast(imgGrayscale)

height, width = imgGrayscale.shape

imgBlurred = np.zeros((height, width, 1), np.uint8)

imgBlurred = cv2.GaussianBlur(imgMaxContrastGrayscale, GAUSSIAN\_SMOOTH\_FILTER\_SIZE, 0)

imgThresh = cv2.adaptiveThreshold(imgBlurred, 255.0, cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C, cv2.THRESH\_BINARY\_INV, ADAPTIVE\_THRESH\_BLOCK\_SIZE, ADAPTIVE\_THRESH\_WEIGHT)

return imgGrayscale, imgThresh

def extractValue(imgOriginal):

height, width, numChannels = imgOriginal.shape

imgHSV = np.zeros((height, width, 3), np.uint8)

imgHSV = cv2.cvtColor(imgOriginal, cv2.COLOR\_BGR2HSV)

imgHue, imgSaturation, imgValue = cv2.split(imgHSV)

return imgValue

def maximizeContrast(imgGrayscale):

height, width = imgGrayscale.shape

imgTopHat = np.zeros((height, width, 1), np.uint8)

imgBlackHat = np.zeros((height, width, 1), np.uint8)

structuringElement = cv2.getStructuringElement(cv2.MORPH\_RECT, (3, 3))

imgTopHat = cv2.morphologyEx(imgGrayscale, cv2.MORPH\_TOPHAT, structuringElement)

imgBlackHat = cv2.morphologyEx(imgGrayscale, cv2.MORPH\_BLACKHAT, structuringElement)

imgGrayscalePlusTopHat = cv2.add(imgGrayscale, imgTopHat)

imgGrayscalePlusTopHatMinusBlackHat = cv2.subtract(imgGrayscalePlusTopHat, imgBlackHat)

return imgGrayscalePlusTopHatMinusBlackHat

1. **SYSTEM TESTING**

Testing is a process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an as-yet –undiscovered error. A successful test is one that uncovers an as-yet- undiscovered error. System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently as expected before live operation commences. It verifies that the whole set of programs hang together. System testing requires a test consisting of several key activities and steps for running a program, string, system and is important in adopting a successful new system. This is the last chance to detect and correct errors before the system is installed for user acceptance testing. Testing is performed to identify errors. It is used for quality assurance. Testing is an integral part of the entire development and maintenance process. The goal of the testing during phase is to verify that the specification has been accurately and completely incorporated into the design, as well as to ensure the correctness of the design itself. For example, the design must not have any logic faults in the design be detected before coding commences, otherwise the cost of fixing the faults will be considerably higher as reflected. Detection of design faults can be achieved by means of inspection as well as walkthrough. Testing is one of the important steps in the software development phase. Testing check for the errors, as a whole of the project testing involves the following test cases: Static analysis is used to investigate the structural properties of the Source. Dynamic testing is used to investigate the behavior of the source code by executing the program on the test data.

* 1. **UNIT TESTING**

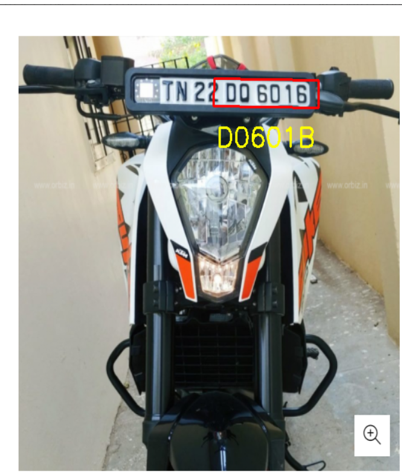
In computer programming, unit testing is a method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures are tested to determine if they are fit for use. Intuitively, one can view a unit as the smallest testable part of an application. In procedural programming, a unit could be an entire module, but is more commonly an individual function or procedure. In object-oriented programming, a unit is often an entire interface, such as a class, but could be an individual method. Unit tests are created by programmers or occasionally by white box testers during the development process. Unit testing is conducted to verify the functional performance of each modular component of the software. Unit testing focuses on the smallest unit of the software design (i.e.), the module. The white-box testing techniques were heavily employed for unit testing.

In this project

Test case 1: The input given for the license plate detection should be image file.

Test case 2: The live image should be detected for helmet detection.

RESULT: All the test cases mentioned above passed successfully. No defects encountered.





**7.2 INTEGRATION TESTING**

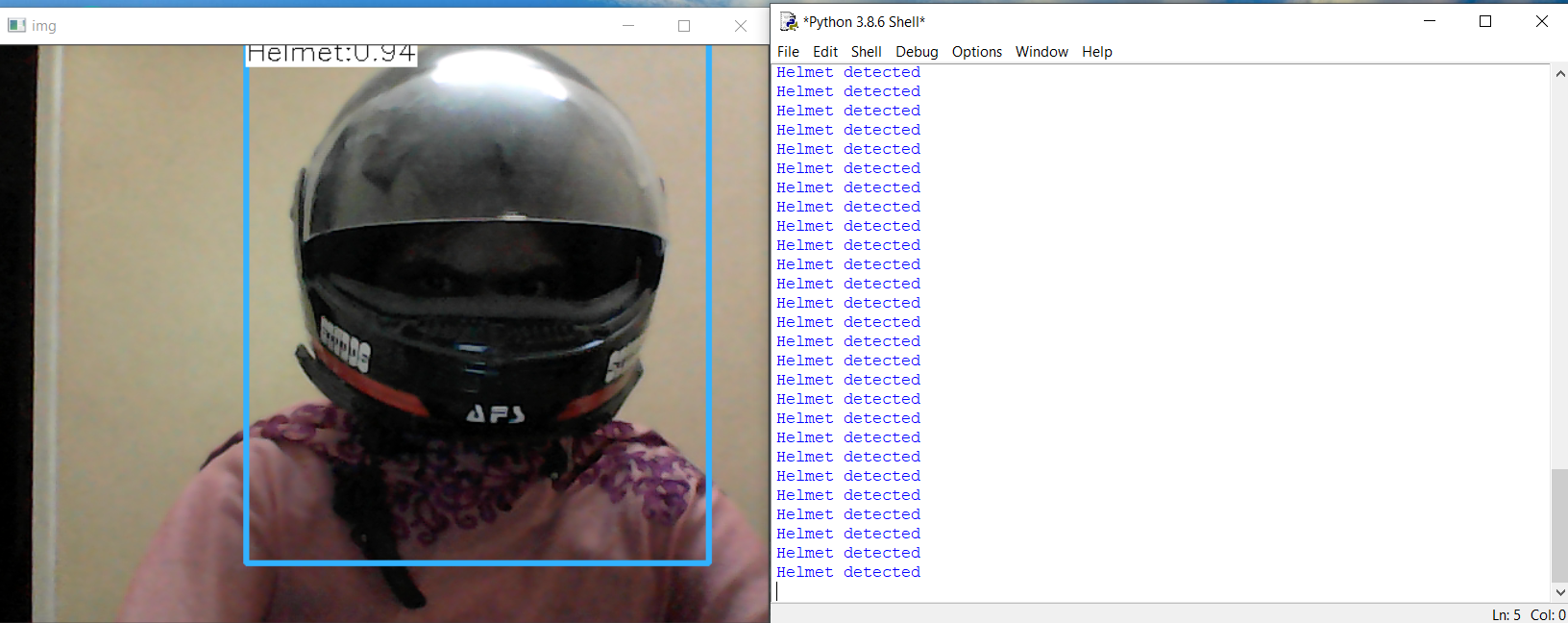
Integration testing is a systematic technique for construction the program structure while at the same time conducting tests to uncover errors associated with interfacing. i.e., integration testing is the complete testing of the set of modules which makes up the product. The objective is to take untested modules and build a program structure tester should identify critical modules. Critical modules should be tested as early as possible. One approach is to wait until all the units have passed testing, and then combine them and then tested. This approach is evolved from unstructured testing of small programs. Another strategy is to construct the product in increments of tested units. A small set of modules are integrated together and tested, to which another module is added and tested in combination. And so on. The advantages of this approach are that, interface dispenses can be easily found and corrected.

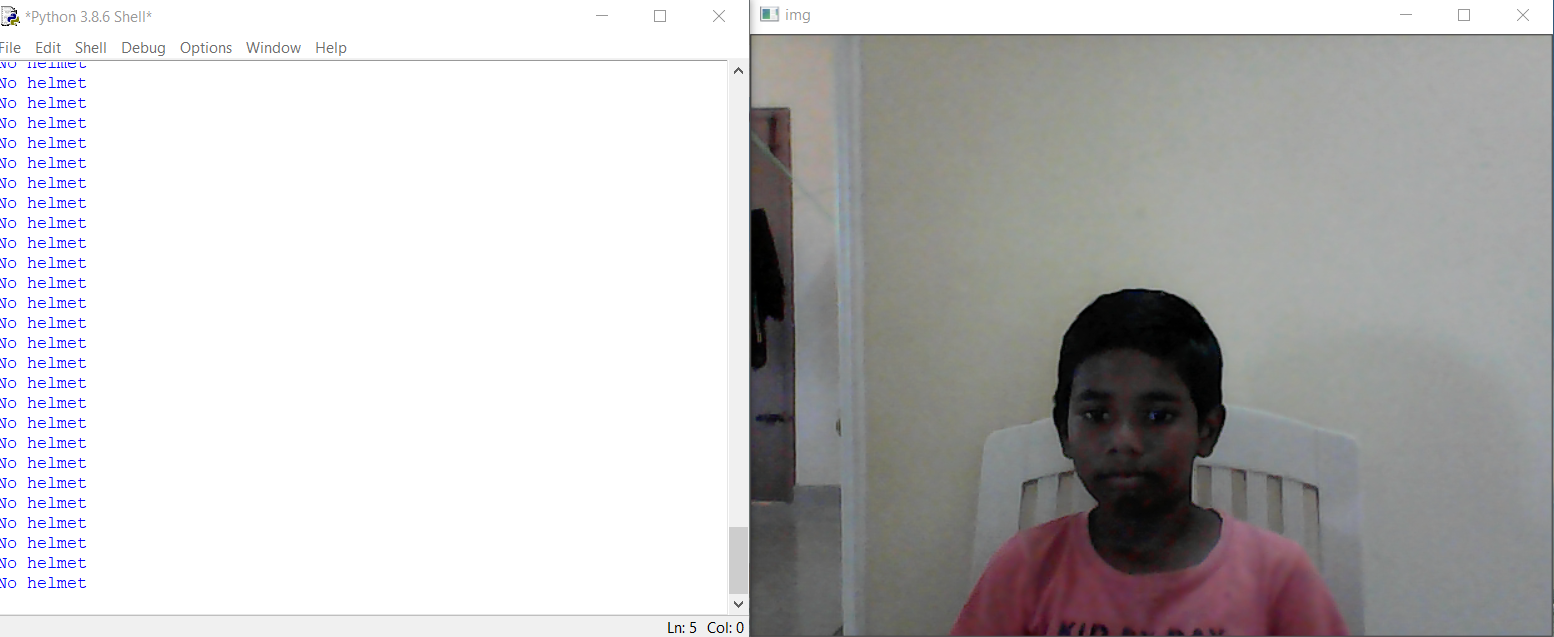
In this project

Test case 3: If the biker wears helmet the output must be given as helmet detected.

Test case 4: If the biker did not wear helmet the output must be given as no helmet.

RESULT: All the test cases mentioned above passed successfully. No defects encountered.

****

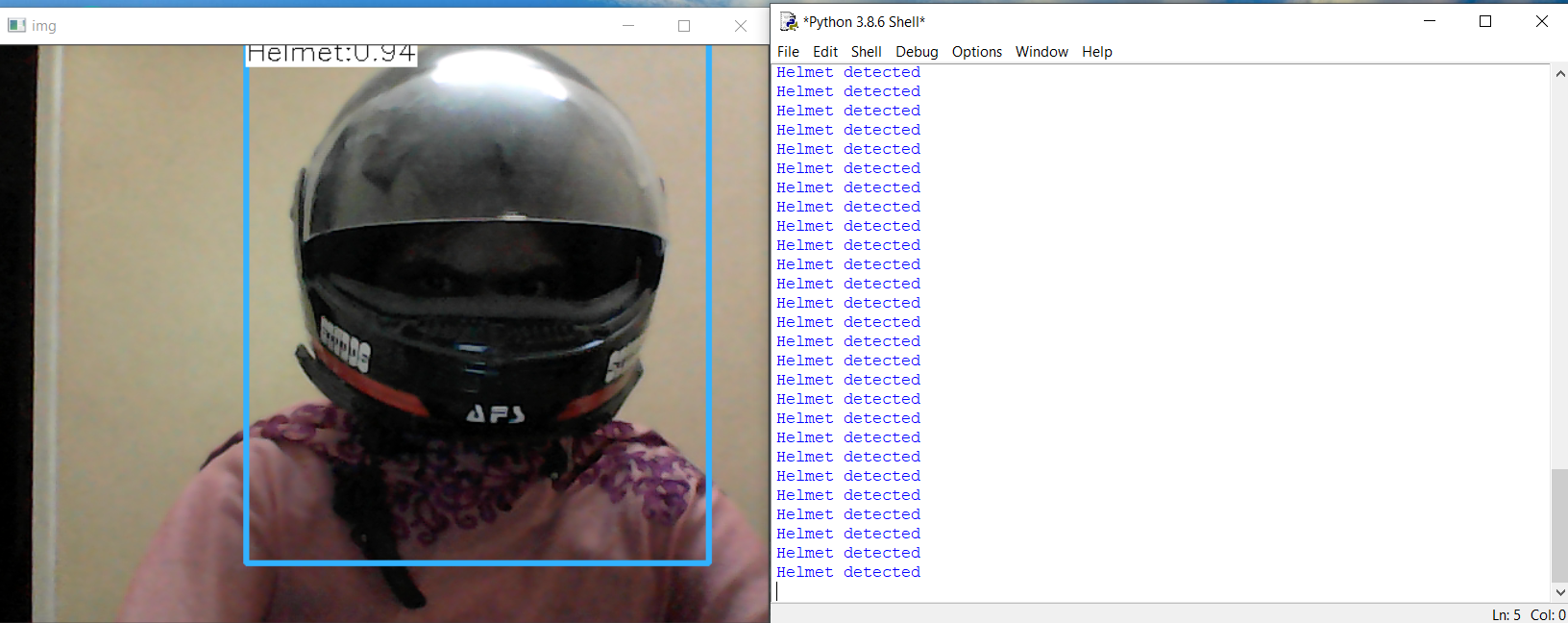
****

1. **CONCLUSION**
   1. **CONCLUSION AND FUTURE ENHANCEMENTS**

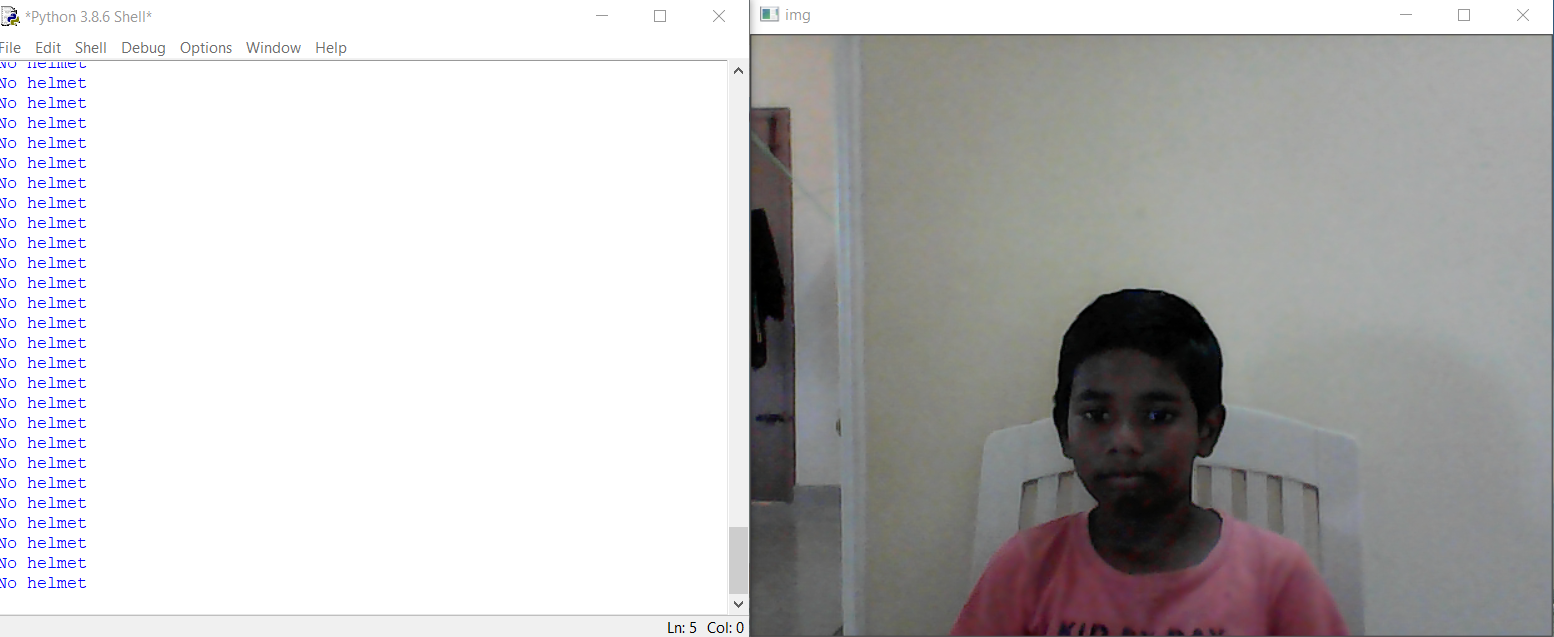
In this project we developed a Real time Helmet detection method by capturing live images .Four algorithms of image preprocessing, license plate location, license plate segmentation and character recognition are introduced in this paper. License plate location is the basis of image preprocessing. The location of license plate has a direct impact on the accuracy of character segmentation. In our project Helmet id detected and vehicle plate detection is obtained by using character recognization and region of interest. An emergency and accident alert system works well to combat the worst situations. In future the helmet can be enhanced by adding some other features like, alcohol detection and riders fatigue detection system to provide a better safety and security to the rider.According to the licence plate detcetion feature extraction process is important.During this detection we can detect the thief when he will taken the vehicle here we are using morphological filters for the addition of color to detect the features maximum we are using on roads and car parking areas for the future purpose we can use deep neural network algorithm using this process we can detect more vehicles at a time.These can be use at traffic,parking ares,airport etc..for the future purpose we can avoid roberries.

**APPENDICES**

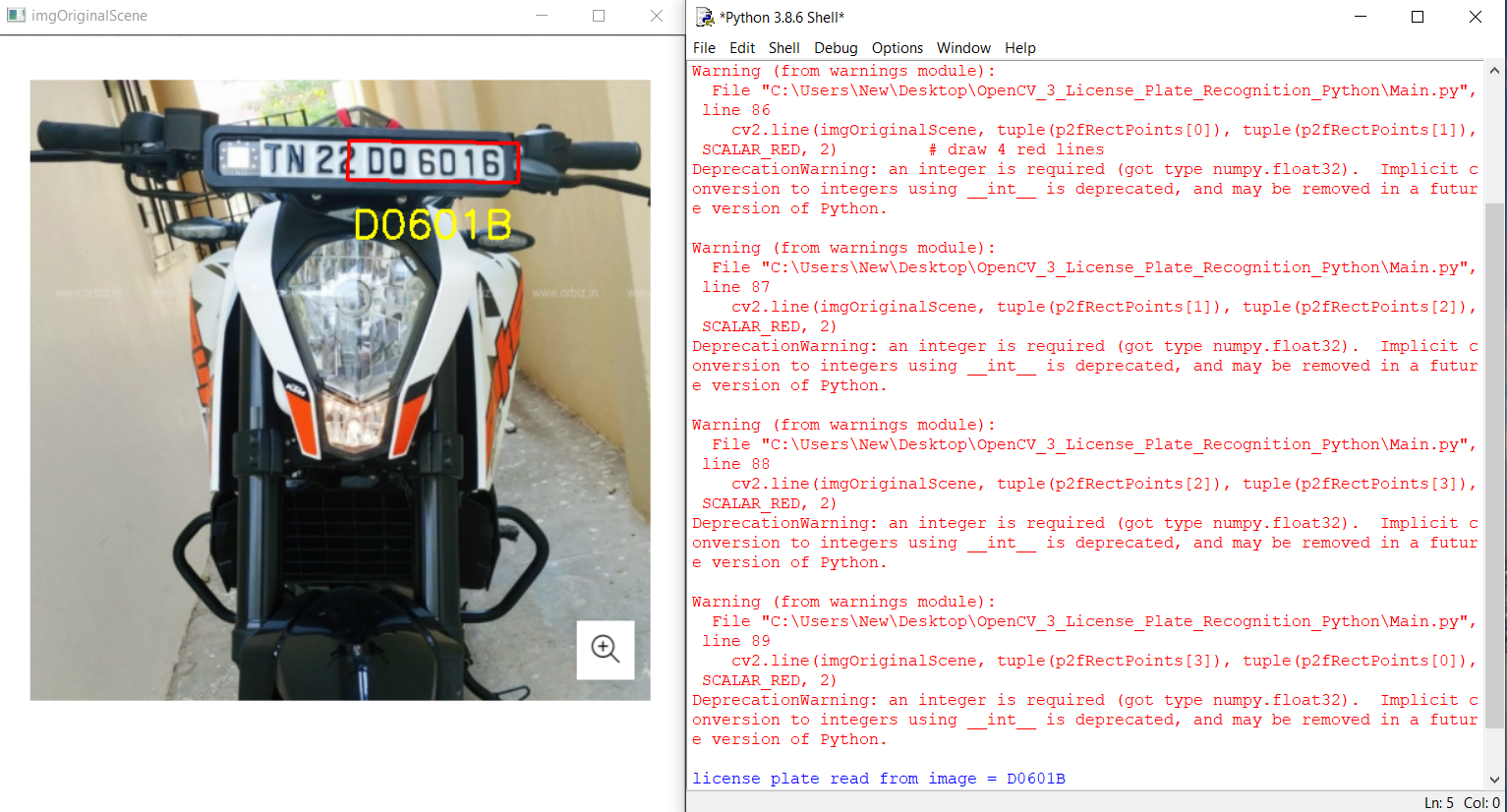
**A.1 SAMPLE SCREENS**

****

A.1.1 Screenshot for helmet detected

****

A.1.2 Screenshot for helmet not detected

****

A.1.3 Screenshot for license plate detection

**A.2 PUBLICATION**

This project was published as a paper in international journal of creative research thoughts on 29.03.2021 by Bangalore Harika, Brightlin Selvamary A, Maddina Sreelekha.

**Published issue**: Vol 9 issue 3, march 2021

**DOI Details**: <http://doi.one/10.1729/Journal.26363>

**REFERENCES:**

[1] A. Adam, E. Rivlin, I. Shimshoni, and D. Reinitz, “Robust real-time unusual event detection using multiple fixed-location monitors,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 30, no. 3, pp. 555–560, March 2008.

[2] Archana D “ Mission on! Innovations in bike systems to provide a safe ride based on IOT”,2017.

[3] B. Duan, W. Liu, P. Fu, C. Yang, X. Wen, and H. Yuan, “Real-time onroad vehicle and motorcycle detection using a single camera,” in Procs. of the IEEE Int. Conf. on Industrial Technology (ICIT), pp. 1–6, 10-13 Feb 2009.

[4] C.-C. Chiu, M.-Y. Ku, and H.-T. Chen, “Motorcycle detection and tracking system with occlusion segmentation,” in Int. Workshop on Image Analysis for Multimedia Interactive Services, Santorini, pp. 32–32 June 2007.

[5] C. Stauffer and W. Grimson, “Adaptive background mixture models for real-time tracking,” in Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), vol. 2,1999.

[6] Dinesh Singh “Visual big data analytics for traffic monitoring in smart city”,2016.

[7] J. Chiverton, “Helmet presence classification with motorcycle detection and tracking,” Intelligent Transport sytems (IET), vol. 6, no. 3, pp. 259–269, September 2012.

[8] Kunal Dahiya “Automatic detection of bike-riders without helmet using surveillance videos in real-time”,2016.

[9] Rongbao Chen “ An improved license plate location method based on edge ”,2012.

[10] R. Rodrigues Veloso e Silva, K. Teixeira Aires, and R. De Melo Souza Veras, “Helmet detection on motorcyclists using image descriptors and classifiers,” in Procs. of the Graphics, Patterns and Images (SIBGRAPI), pp. 141–148,Aug 2014.

[11] R. Silva, K. Aires, T. Santos, K. Abdala, R. Veras, and A. Soares, “Automatic detection of motorcyclists without helmet,” in Computing Conf. (CLEI), XXXIX Latin American, pp. 1–7, Oct 2013.

[12] Sabjit Kaur “ An automatic number plate recognition system under image procesing”,2017.

[13] W. Hu, T. Tan, L. Wang, and S. Maybank, “A survey on visual surveillance of object motion and behaviors,” IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, vol. 34, no. 3, pp. 334S–352, Aug 2004.

[14] Z. Chen, T. Ellis, and S. Velastin, “Vehicle detection, tracking and classification in urban traffic,” in Procs. of the IEEE Int. Conf. on Intelligent Transportation Systems (ITS), Anchorage, AK pp. 951–956, Sept 2012.

[15] Z. Zivkovic, “Improved adaptive gaussian mixture model for background subtraction,” in Proc. of the Int. Conf. on Pattern Recognition (ICPR), vol. 2, pp. 28–31, Aug.23-26 2004.