

Visvesvaraya Technological University



PROJECT WORK REPORT ON **“DETECTION OF VIOLATION OF TRAFFIC RULES** **USING DEEP LEARNING”**

Submitted by,

Name of the students

USN

Kavya T N

4JN17TE017

Pooja H

4JN17TE029

Sneha Mohan Hebbar

4JN17TE044

Sushma S

4JN17TE047

Under the guidance of:
Mrs. Rashmi M Hullamani
Assistant Professor
Dept of TCE



Department of Telecommunication Engineering
JAWAHARLAL NEHRU NATIONAL COLLEGE OF
ENGINEERING
SHIVAMOGGA – 577204
2020-21

**JAWAHARLAL NEHRU NATIONAL COLLEGE OF ENGINEERING
SHIVAMOGGA – 577204**



Department of Telecommunication Engineering
CERTIFICATE

This is to certify that the project work entitled

**“DETECTION OF VIOLATION OF TRAFFIC RULES
USING DEEP LEARNING”**

Is a bonafide work carried out jointly by

KAVYA T N	4JN17TE017
POOJA H	4JN17TE029
SNEHA MOHAN HEBBAR	4JN17TE044
SUSHMA S	4JN17TE047

These students of 8th semester B.E., Telecommunication Engineering under our supervision and guidance towards the partial fulfillment of the requirements for the award of the degree of Bachelor of Engineering in Telecommunication Engineering as per the university regulations during the year 2020-21.

.....
Signature of the Guide
Mrs. Rashmi M Hullamani
Asst. Prof., T.C.E.

.....
Signature of the HOD
Dr. M. B. Ushadevi
HOD, T.C.E.

.....
Signature of Principal
Dr. PManjunatha
Principal, JNNCE

Name of Examiners:

Signature of Examiners with date:

1.

.....

2.

.....

ABSTRACT

We usually prefer motorbikes over other vehicles as it is significantly less expensive to run, less demanding to park and adaptable in rush hour gridlock. In India, in excess of 118 million individuals are utilizing bikes. Since lot of bikes travel along with other vehicles wearing headgear is critical to decrease the danger of injuries. we propose an approach where we identify the motorcycle riders without headgear and who are triple riding utilizing surveillance videos in real-time. Our approach also proposes a system where a message will be sent to the concerned authority about the vehicle details of the person who violates the above-mentioned rules with the proof. The proposed approach initially recognizes motorcycle riders utilizing OCR and object segmentation. The vertical projection of binary image is utilized for counting number of riders.

ACKNOWLEDGEMENT

Firstly, we would like to express our sincere gratitude to the almighty for his solemn presence throughout the project work.

We express our sincere gratitude to **Dr. M. B. Usha Devi, Professor**, Head of the Telecommunication Engineering Department for providing us with adequate facilities, ways and means by which we are able to complete this project work.

We are deeply indebted to our project coordinators **Mrs. Aparna**, Assistant Professor, Department of Telecommunication Engineering and **Mr. Benak Patel M P**, Asst. Professor, TCE department for providing us with valuable advice and guidance during the course of the study. Without their wise counsel and able guidance, it would have been impossible to complete our project work in this manner.

We would like to express a deep sense of gratitude and thanks profusely to our project guide **Mrs. Rashmi M Hullamani**, designation, Department of Telecommunication engineering for her/his proper guidance and valuable suggestions. Without her wise counsel and able guidance, it would have been impossible to complete the project work in this manner.

We would also like to express our special thanks to the principal **Dr. Manjunath P.** for providing an opportunity to carry out this project work.

We would like to extend our heartfelt gratitude to the teaching and non-teaching staff of **Department of Electronics and Telecommunication Engineering** for their constructive support and co-operation at each and every juncture of the project work.

Finally, we would also like to express our gratitude to **Jawaharlal Nehru New College of Engineering** for providing us with all the required facilities without which the project work would not have been possible.

Project associates,

Name	USN
KAVYA T N	4JN17TE017
POOJA H	4JN17TE029
SNEHA MOHAN HEBBAR	4JN17TE044
SUSHMA S	4JN17TE047

CONTENT

Chapter No			Title	Page No
			Abstract	i
			Acknowledgement	ii
			Table of contents	iii
			List of figures	iv
1			Introduction	
	1.1		General introduction	1
	1.2		Literature survey	2-4
	1.3		Problem statement	5
	1.4		Methodology	5
	1.5		Scope of the project	6-7
	1.6		Limitations	8
	1.7		Organization of the report	8
2			Theoretical Background	
	2.1		Block diagram	9
	2.2		Block description	9-13
3			Design and Implementation	
	3.1		Introduction	14
	3.2		Software code	14-23
	3.3		Implementation	24
	3.4		Software details	24
		3.4.2	Flow chart of the proposed system	32
		3.4.3	Algorithm	33-35
	3.5		Summary	36
4			Results and Discussions	37-41
5			Conclusions and Future Scope	
	5.1		Conclusions	42
	5.2		Future Scope	42
			References	43
			Appendix	

LIST OF FIGURES

- Fig. 1.4.1: Riding bike without helmet
- Fig. 1.4.2: Riding bike without helmet
- Fig. 1.4.3: Triple or more than triple riding
- Fig. 1.4.4: Using mobile phones while riding
- Fig.2.1 Block diagram of the proposed system
- Fig 2.2.1 Graphical User Interface
- Fig 2.2.2 The YOLO Model
- Fig 2.2.3 Blob Image
- Fig 2.2.4 Non-max suppression
- Fig 3.3.1.1a) Python Installing
- Fig 3.3.1.2 b) OpenCV
- Fig 3.3.1.2 c) OpenCV Detection
- Fig 3.3.1.2 d) Installing OpenCV libraries in command prompt
- Fig 3.3.1.3 e) Installing Pandas libraries in command prompt
- Fig 3.3.1.4 f) Installing Requests libraries in command prompt
- Fig 3.3.1.5 g) Installing Pillow libraries in command prompt
- Fig 3.3.1.6 h) Datasets
- Fig 3.3.1.7 i) Input and Output window
- Fig 3.4.2 a) Flow diagram process of detecting the image
- Fig 3.4.3 a) Grayscale Technique
- Fig 3.4.3 b) YOLO Algorithm
- Fig 3.4.3 c) Intersection over Union
- Fig 3.4.3 d) Non-max suppression
- Fig 4.1.1 Helmet Input Video
- Fig 4.1.2 Detect persons
- Fig 4.1.3 Detect Helmet
- Fig 4.1.4 TripleRide input Video
- Fig 4.1.5 Detect Persons
- Fig 4.1.6 Detect Tripleride video
- Fig 4.1.7 Detect Helmet video
- Fig 4.1.8 Traffic Detection
- Fig 4.1.9 Detect NumberPlate
- Fig 4.1.10 Detect Text NumberPlate

Chapter 1

Introduction

1.1 General Introduction

Bike is an extremely mainstream method of transportation in India. However, there is a high risk involved due to lack of protection. To decrease the involved risk, it is highly desirable for motorcycle riders to use helmet. That's why the government has made it a punishable offense to ride a bike without helmet. The drawback of the current method where human intervention is required can be solved by our proposed method. Automation of this procedure is exceptionally attractive for vigorous observing of these infringements and additionally it likewise altogether lessens the measure of human resource required. Also, many countries are adopting systems involving surveillance cameras at public places. So, the solution for detecting violators using the existing infrastructure is also cost-effective. However, in order to adopt such automatic solutions certain challenges, need to be addressed.

1.2 Literature Survey

1. "Automatic Helmet Detection on Public Roads"

Done by - Maharsh Desai , Shubham Khandelwal , Lokneesh Singh , Prof. Shilpa Gite.

Histogram of Oriented Gradients Descriptor: provides better performance than other existing feature sets. It is used to extract human feature from visible spectrum images. It has been determined that when LBP combined with HOG descriptor it improves, detection, performance considerably on some data sets. RESULTS To ensure bike rider's safety, we have designed this project. Many projects have been designed so far but they all are concentrated more on four wheelers. Very less importance was given to motorbikes. Today accidents caused by motorbikes are more than cars. Thus, in this project safety of bike rider is major concern.

Helmet Authentication to ensure that the bike rider is wearing a helmet. Alcohol detection to ensure that the bike rider has not consumed any type of alcohol. Fall detection in case of accident, to inform bike rider's family about the accident. In future we intend to use more advanced safety measures like to check alcohol consumption, lane change detection, collision detection, traffic information, e-toll collection, license renewal etc. We also think of applying deep neural network techniques & make transportation more intelligent.

2. “Helmet detection on two-wheeler riders using machine learning”

Done by - AMANDEEP RATHEE, KRISHNANGINI KALITA, MAHIMA SINGH DEO
MACHINE LEARNING APPROACHES, A.

Random Forest, Gradient Boosted Trees, Support Vector Machine, all of the five classifiers are trained on about 2000 images. The training images contain both, the front view and the back view of the vehicle. The surveying of various algorithms was done in R [10]. However, the final end-to-end system was implemented in MATLAB using the best classifier. All algorithms were tested separately for vehicle classification and helmet detection task. Raw pixel values are used as features in vehicle classification as well as helmet detection. The metric that's chosen is accuracy since all the classes are balanced, that is, each class has almost the same number of images in both the tasks. 20% of the entire data that was collected wasn't used to train the classifiers. Instead, it was used to test the classifiers. The results are shown in the following tables. A deep neural network is expected to perform better than a random forest in image recognition, but due to lack of data, it does not perform as expected. As stated earlier, deep learning algorithms shine when there is a lot of training data. In the future, the system can be improved by scrutinizing its drawbacks. There are a couple of drawbacks.

3. “The Real-Time Detection of Traffic Participants Using YOLO Algorithm”

Aleksa Ćorović, Velibor Ilić, Siniša Đurić, Mališa Marijan, and Bogdan Pavković

YOLO v3 algorithm consists of fully CNN and an algorithm for post-processing outputs from neural network. CNNs are special architecture of neural networks suitable for processing grid-like data topology. The distinctive feature of CNNs which bears importance in object detection is parameter sharing. Unlike feedforward neural networks, where each weight parameter is used once, in CNN architecture each member of the kernel is used at every position of the input, which means learning one set of parameters for every location instead a separate set of parameters. This feature plays important role in capturing whole scene on the road. In this paper, we have presented application of YOLOv3 algorithm for real time detection of traffic participants. The weights of the neural network were initialized using a pretrained model trained on the COCO dataset. The neural network was further trained on the Berkley Deep Drive dataset to specialize in the detection of five classes of traffic participants. False detections were investigated on the custom dataset made from images representing different

traffic situations in Novi Sad. The application of YOLO algorithm presented in this paper provides a solid base for the object detection module as a part of the ADAS. In the future work, precision of the algorithm could be improved with training on the bigger and more diverse datasets that cover different weather and lighting conditions. Also, this algorithm could be used in fusion with other sensor's readings to reduce the number of false detections.

3. **“YOLO-LITE: A Real-Time Object Detection Algorithm Optimized for Non-GPU Computers”**

Done by - Rachel Huang, Jonathan Pedoeem, Cuixian Chen

YOLO v3 algorithm consists of fully CNN and an algorithm for post-processing outputs from neural network. CNNs are special architecture of neural networks suitable for processing grid-like data topology. The distinctive feature of CNNs which bears importance in object detection is parameter sharing. Unlike feedforward neural networks, where each weight parameter is used once, in CNN architecture each member of the kernel is used at every position of the input, which means learning one set of parameters for every location instead a separate set of parameters. This feature plays important role in capturing whole scene on the road. In this paper, we have presented application of YOLOv3 algorithm for real time detection of traffic participants. The weights of the neural network were initialized using a pretrained model trained on the COCO dataset. The neural network was further trained on the Berkley Deep Drive dataset to specialize in the detection of five classes of traffic participants. False detections were investigated on the custom dataset made from images representing different traffic situations in Novi Sad. The application of YOLO algorithm presented in this paper provides a solid base for the object detection module as a part of the ADAS. In the future work, precision of the algorithm could be improved with training on the bigger and more diverse datasets that cover different weather and lighting conditions. Also, this algorithm could be used in fusion with other sensor's readings to reduce the number of false detections.

4. **“YOLO-LITE: A Real-Time Object Detection Algorithm Optimized for Non-GPU Computers”**

Done by - Rachel Huang, Jonathan Pedoeem, Cuixian Chen

YOLO-LITE ARCHITECTURE Our goal with YOLO-LITE is to develop an architecture that can run at a minimum of ~ 10 frames per second (FPS) on a non-GPU powered computer

with a mAP of 30% on PASCAL VOC. This goal is determined from looking at the state-of-the-art and creating a reasonable benchmark to reach. YOLO-LITE offers two main contributions to the field of object detection: Demonstrates the capability of shallow networks with fast non-GPU object detection applications. Suggests that batch normalization is not necessary for shallow networks and, in fact, slows down the overall speed of the network. While some works [15], [16], [17] focused on creating an original convolution layer or pruning methods in order to shrink the size of the network, YOLO-LITE focuses on taking what already existed and pushing.

Additionally, YOLO-LITE focuses on speed and not overall physical size of the network and weights. YOLO-LITE achieved its goal of bringing object detection to non-GPU computers. In addition, YOLO-LITE offers several contributions to the field of object detection. First, YOLO-LITE shows that shallow networks have immense potential for lightweight realtime object detection networks. Running at 21 FPS on a non-GPU computer is very promising for such a small system. Second, YOLO-LITE shows that the use of batch normalization should be questioned when it comes to smaller shallow networks. Movement in this area of lightweight real-time object detection is the last frontier in making object detection usable in everyday instances. Lightweight architectures in general have a significant drop off in accuracy from the original YOLO architecture. YOLOv2 has a map of 48.1% with a decrease to 23.7% in YOLOv2-Tiny. The YOLO-LITE architecture has a map decrease down to 12.16%. There is a constant tradeoff for speed in lightweight models and accuracy in a larger model. Although YOLO-LITE achieves the fastest map compared to state of the art, the accuracy prevents the model from succeeding in real applications such as an autonomous vehicle. Future work may include techniques to increase the map for both COCO and VOC models.

5. “Detection of Non-Helmet Riders and Extraction of License Plate Number using Yolo v2 and OCR Method”

Done by - Prajwal M. J., Tejas K. B., Varshad V., Mahesh Madivalappa Murgod, Shashidhar R

Manual surveillance of traffic using CCTV is an existing methodology. But here so many iterations have to be performed to attain the objective and it demands a lot of human resource. Therefore, cities with millions of populations having so many vehicles running on the roads cannot afford this inadequate manual method of helmet detection. So here we propose a methodology for full helmet detection and license plate. A Non-Helmet Rider Detection

system is developed where a video file is taken as input.

If the motorcycle rider in the video footage is not wearing helmet while riding the motorcycle, then the license plate number of that motorcycle is extracted and displayed. Object detection principle with YOLO architecture is used for motorcycle, person, helmet and license plate detection. OCR is used for license plate number extraction if rider is not wearing helmet. Extracting the license number will allow the system to automatically send a ticket to the registered owner of the two-wheeler, in case they are not wearing a helmet. If the above issues are addressed, and a lot of training data is gathered from surveillance cameras, the system can become much more robust and reliable than it is now.

1.3 Problem Statement

- There is no proper method to avoid the persons who are violate the traffic rules Manual or visual method takes more time to detect the person.
- More accidents are occurred by the violation of traffic rules So we are planned a method OCR(optical character recognition)that helps recognising texts or number plates of vehicles.
- The main aim of the project is to provide effective and efficient technology to detect problems in the road especially persons who are not wearing helmet, use of mobile phones, triple riding.

1.4 Methodology

- * **STEP1** - A real time input video is fed to the model.
- * **STEP2** - The model is previously trained with a dataset. Then the real time input is compared with dataset.
- * **STEP3** - If the two-wheeler is detected in the input the process continues else the image is discarded.
- * **STEP4** - After two-wheeler is detected, the pretrained model detects the rider wearing helmet and triple riding. If the rules are violated it goes to the next step else the image is discarded.
- * **STEP5** - When the rules are violated an SMS alert will be sent to the concerned authority.

1.4.1 Objectives of the paper

1. Detection of persons on the motorbikes who are not wearing the helmet while crossing the signals and junctions.



Fig. 1.4.1: Riding bike without helmet



Fig. 1.4.2: Riding bike without helmet

2. Detection of people who are triple riding and using mobile phones on the motorbikes while crossing the signals and junctions.



Fig. 1.4.3: Triple or more than triple riding



Fig. 1.4.4: Using mobile phones while riding

3. Sending an SMS to the concerned authority about the vehicle details of the person who violates the above-mentioned rules.

1.5 Scope of the project

- Safety drive
- Reduces Accidents
- Continuous monitoring of traffic rules 24x7
- No need of human intervention
- An automated system, if employed to detect and penalize motorcycle riders without a helmet, will naturally motivate the two-wheeler riders to wear helmets.

1.6 Limitations

1. The detection has a poor performance when the images are not very clear, the safety helmets are too small and obscure, and the background is too complex.
2. Some images of the dataset are less in quantity.
3. The processing operations of the images are confined to rotation, cutting, and zooming.
4. The manual labelling is not comprehensive and may miss some objects.

1.7 Organization of the report

This report has been organized into five chapters. The first chapter gives in depth of introduction of the project. In the second chapter we are discussing about theoretical background of the task and related in detail. The third chapter is the head of the project, which gives the details about the design and implementation of the Road sign recognition system in the smart car. The fourth chapter deals with the information of result obtained and the fifth chapter deals with the conclusion and future scope of the project.

Chapter 2

THEORETICAL BACKGROUND

2.1 BLOCK DIAGRAM

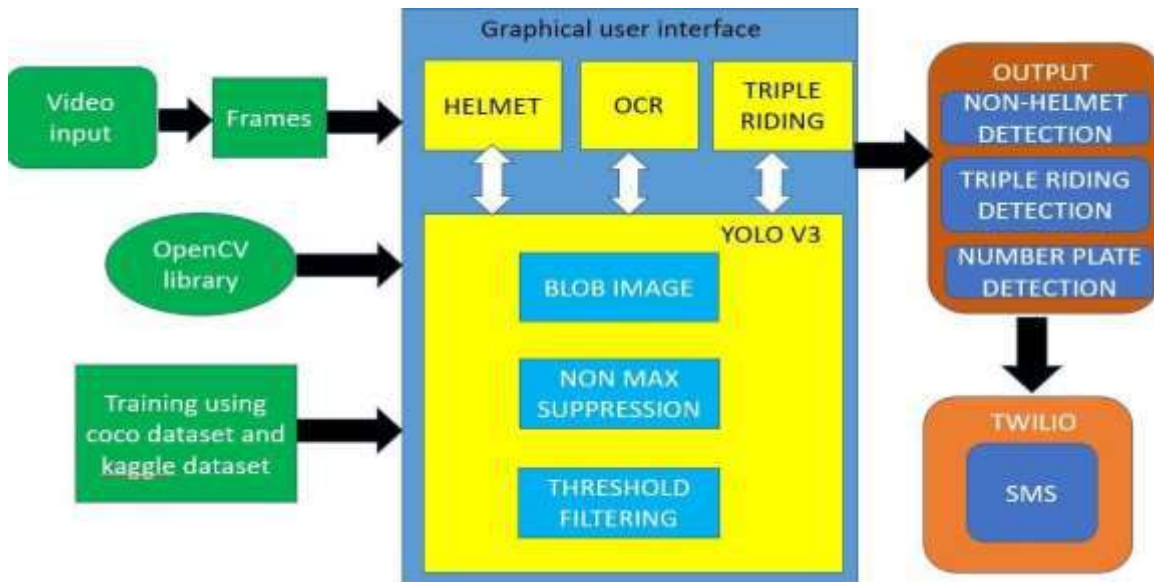


Fig.2.1 Block diagram of the proposed system

Block diagram Here is simple block diagram of proposed system trained by using coco dataset and Kaggle dataset and also with OpenCV libraries. Then giving the input with correct framed video to the GUI that is having three working objectives like helmet detection and triple riding by doing Blob image, Non max these suppression, Threshold filtering with YOLO V3. Then give the output of vehicles who are violating the rules and save them in it then by using GSM message will sent to respective authorities.

2.2 BLOCK DIAGRAM DESCRIPTION

2.2.1 Graphical User Interface

The graphical user interface, developed in the late 1970s by the Xerox Palo Alto research laboratory and deployed commercially in Apple's Macintosh and Microsoft's Windows operating systems, was designed as a response to the problem of inefficient usability in early, text-based command-line interfaces for the average user.

Graphical user interfaces would become the standard of user-centered design in software application programming, providing users the capability to intuitively operate computers and

other electronic devices through the direct manipulation of graphical icons such as buttons, scroll bars, windows, tabs, menus, cursors, and the mouse pointing device. Many modern graphical user interfaces feature touchscreen and voice-command interaction capabilities.

- A GUI (graphical user interface) is a system of interactive visual components for computer software.
- A GUI displays objects that convey information, and represent actions that can be taken by the user.
- **Tkinter** -It is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications.
- Hence we use Tkinter in our project For user interface for different applications.
- We create buttons in the GUI page for each applications.



Fig 2.2.1 Graphical User Interface

2.2.2 The Yolo Model

- YOLO calculates the probabilities and bounding boxes for objects in a single forward pass. The way this is done is splitting the image into grids and detecting objects for each of the boxes.
- However, this is not done sequentially, and is actually implemented as a convolution operation.
- For example: if
- Input image size is (448,448,3)
- Number of classes to predict = 25 and Grid size = (7,7)

- Then, each label vector will be of length 30 (Pc, Bx, By, Bw, Bh, +25 classes).

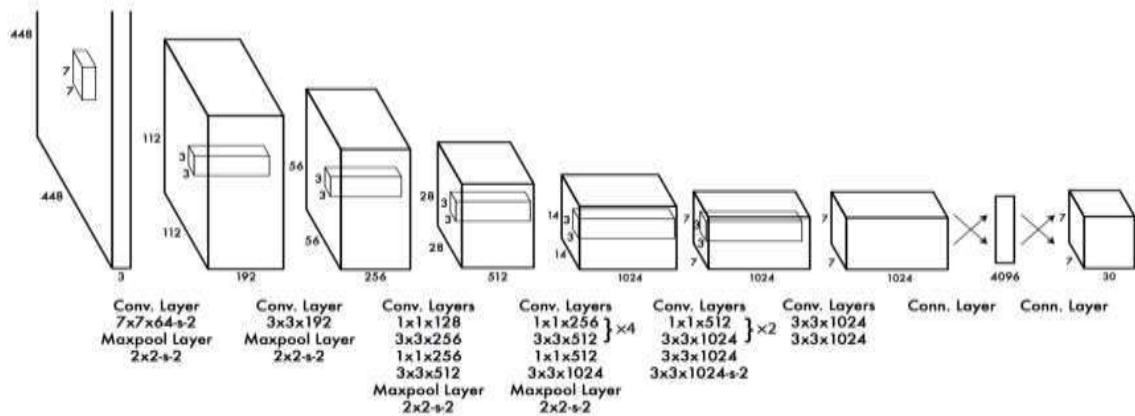


Fig 2.2.2 The YOLO Model

2.2.3 COCO (Common Objects in Context)

The COCO dataset is labeled, providing data to train supervised computer vision models that are able to identify the common objects in the dataset. Of course, these models are still far from perfect, so the COCO dataset provides a benchmark for evaluating the periodic improvement of these models through computer vision research.

- This program uses the COCO (Common Objects in Context) class list, which has 25 object categories. For reference, these classes are given in the file “coco_names”.
- When the detected images matches the objects list in coco class, the object number gets incremented by one. Hence when the person number reaches three we can say that triple riding detection is achieved.

The first few entries are:

- 1.person
- 2.bicycle
- 3.car
- 4.motorbike
- 5.aeroplane
- 6.bus
- 7.train
- 8.truck

Applications

- Can be used in all traffic signals.
- Can be used in any type streets.

2.2.4 BLOB IMAGE

In computer vision, blob detection methods are aimed at detecting regions in a digital image that differ in properties, such as brightness or color, compared to surrounding regions. Informally, a blob is a region of an image in which some properties are constant or approximately constant; all the points in a blob can be considered in some sense to be similar to each other. The most common method for blob detection is convolution.

Given some property of interest expressed as a function of position on the image, there are two main classes of blob detectors:

- (i) differential methods, which are based on derivatives of the function with respect to position, and
- (ii) Methods based on local extrema, which are based on finding the local maxima and minima of the function. With the more recent terminology used in the field, these detectors can also be referred to as interest point operators, or alternatively interest region operators (see also interest point detection and corner detection).

There are several motivations for studying and developing blob detectors. One main reason is to provide complementary information about regions, which is not obtained from edge detectors or corner detectors. In early work in the area, blob detection was used to obtain regions of interest for further processing. These regions could signal the presence of objects or parts of objects in the image domain with application to object recognition and/or object tracking. In other domains, such as histogram analysis, blob descriptors can also be used for peak detection with application to segmentation. Another common use of blob descriptors is as main primitives for texture analysis and texture recognition. In more recent work, blob descriptors have found increasingly popular use as interest points for wide baseline stereo matching and to signal the presence of informative image features for appearance-based object recognition based on local image statistics. There is also the related notion of ridge detection to signal the presence of elongated objects.

- Input Video will be converted into frames and the number of frames per second depends on the processor.
- YOLO architecture needs blob images and hence normal images are converted into blob images for processing.

- When compared to the phase 1 now we are able to get better efficiency because we used YOLO V3 algorithm.

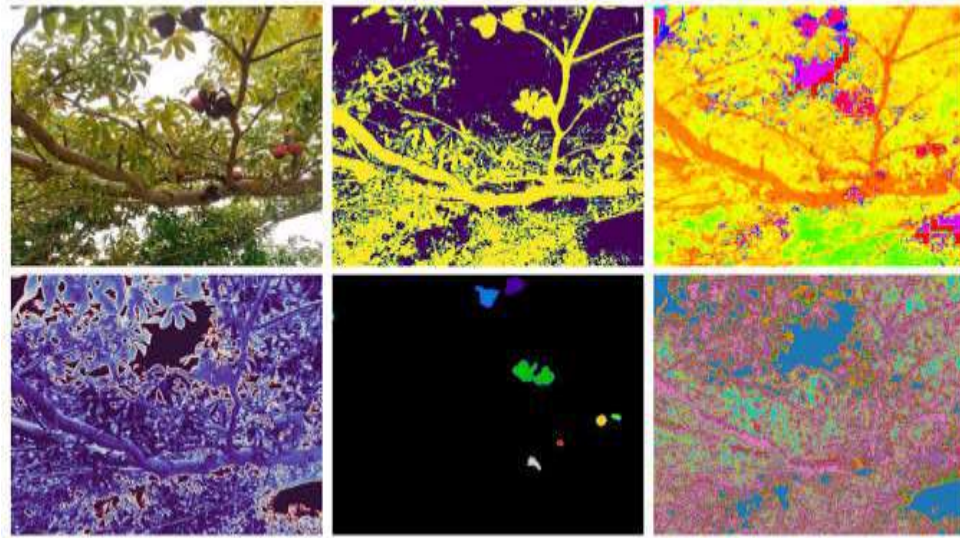


Fig 2.2.3 Blob Image

2.2.5 OCR (Optical Character Recognition)

Optical character recognition or optical character reader (OCR) is the electronic or mechanical conversion of images of typed, handwritten or printed text into machine-encoded text, whether from a scanned document, a photo of a document, a scene-photo or from subtitle text superimposed on an image.

2.2.6 Non-max suppression and threshold filtering

- The first step, quite naturally, is to get rid of all the boxes which have a low probability of an object being detected.
- Non-max suppression makes use of a concept called “intersection over union” or IoU.
- It takes as input two boxes, and as the name implies, calculates the ratio of the intersection and union of the two boxes.

The simplest thresholding methods replace each pixel in an image with a black pixel if the image intensity is less than some fixed constant T or a white pixel if the image intensity is greater than that constant. In the example image on the right, this results in the dark tree becoming completely black, and the white snow becoming completely white.

Chapter 3

DESIGN AND IMPLEMENTATION

3.1 Introduction

This project uses OpenCV and Python language for coding. All the inputs are given by using datasets and output will be obtained by running python file. This project also uses Graphical User Interface(GUI).The frames from the system input video are processed in the open CV platform. And using the downloaded datasets as inputs it will detect the vehicles which are violating the rules and send the messages to respective authorities.

Sensing with Python

Now that we've hooked our Ultrasonic Sensor up to our Pi, we need to program a Python script to detect distance!

The Ultrasonic sensor output (ECHO) will always output low (0V) unless it's been triggered in which case it will output 5V (3.3V with our voltage divider!). We therefore need to set one GPIO pin as an output, to trigger the sensor, and one as an input to detect the ECHO voltage change.

First, import the all the messages from tkinter

```
from tkinter import *
from tkinter import messagebox
top = Tk()
```

Declare a variables and initialize the color, height, width and size

```
C = Canvas(top, bg="blue", height=250, width=300)
filename = PhotoImage(file = "resized_test.png")
background_label = Label(top, image=filename)
background_label.place(x=0, y=0, relwidth=1, relheight=1)
```

pack() will put a widget inside a frame

```
C.pack()
```

Import window from tkinter and assign the title, geometry and configuration.

```

from tkinter import *
window=Top
root=window
window.title("project1")
window.geometry("3500x900")
window.configure(background="#E4287C")

```

Labelling will be done by declaring a variable called lbl and labelling the required name with color, size, width, height.

```

lbl=Label(window, text="Triple Ride and Helmet Detection" ,fg="black" ,width=25
,height=1,font=('times', 30, 'italic bold underline'))
lbl.place(x=400,y=10)

```

Then import os and define time(), start python files that are in os

```

import os
def time():
    os.startfile("yolo_detection_webcam1.py")
def time1():
    os.startfile("Helmet_detection_YOLOV3.py")
def time2():
    os.startfile("yolo_detection_images.py")
def time3():
    os.startfile("yolo_detection_webcam.py")    #start python file
def time4():
    os.startfile("helmet.py")
def time5():
    os.startfile("live1.py")
def time6():
    os.startfile("yolo_detection_images4.py")
def time8():
    try:

```

Import the Python libraries, and declare threshold filtering and NMS (Non Maxima Suppression) values

```

import numpy as np
import cv2
import cv2 as cv

confidenceThreshold = 0.0
NMSThreshold = 0.6

```

#importing libraries

Take the dataset inputs from coco and other dataset files

```
modelConfiguration = 'cfg/yolov3.cfg'
modelWeights = 'yolov3.weights'

labelsPath = 'coco.names'
labels = open(labelsPath).read().strip().split("\n")

np.random.seed(10)
COLORS = np.random.randint(0, 255, size=(len(labels), 3), dtype="uint8")

net = cv2.dnn.readNetFromDarknet(modelConfiguration, modelWeights)

outputLayer = net.getLayerNames()
outputLayer = [outputLayer[i][0] - 1] for i in net.getUnconnectedOutLayers()
```

Then VideoCapture() will help to capture the input video

```
video_capture = cv2.VideoCapture("om1.mp4")

(W, H) = (None, None)
count = 0
```

writing the while loop and if loop for repeating the process

```
while True:
    ret, frame = video_capture.read()
    fram1=frame
    #frame = cv2.flip(frame, 1)
    if W is None or H is None:
        (H,W) = frame.shape[:2]
```

Blobing of image will be done in this section of code

```
blob = cv2.dnn.blobFromImage(frame, 1 / 255.0, (416, 416), swapRB = True, crop =
False)
net.setInput(blob)
layersOutputs = net.forward(outputLayer)

boxes = []

confidences = []
classIDs = []
```

for and if loop will be used here for this particular section of the code.

```
for output in layersOutputs:
    for detection in output:
        scores = detection[5:]
```

```

classID = np.argmax(scores)
confidence = scores[classID]
if confidence > confidenceThreshold:
    box = detection[0:4] * np.array([W, H, W, H])
    (centerX, centerY, width, height) = box.astype('int')
    x = int(centerX - (width/2))
    y = int(centerY - (height/2))

    boxes.append([x, y, int(width), int(height)])
    confidences.append(float(confidence))
    classIDs.append(classID)

```

Non Maxima Suppression will be done in this part of the code.

```

#Apply Non Maxima Suppression

detectionNMS = cv2.dnn.NMSBoxes(boxes, confidences, confidenceThreshold,
NMSThreshold)
if len(detectionNMS) > 0:
    for i in detectionNMS.flatten():
        (x, y) = (boxes[i][0], boxes[i][1])
        (w, h) = (boxes[i][2], boxes[i][3])

        color = [int(c) for c in COLORS[classIDs[i]]]

        print(labels[classIDs[i]])
        if labels[classIDs[i]]=="motorbike": #since my detector only has 1 class
            cv2.imwrite("tripleride//frame%d.jpg" % count, frame[y-200:y+h, x-20:x+w])
            count = count + 1
cv2.imshow('Output', frame)
if (cv.waitKey(1) & 0xFF == ord('q')):
    break

```

Finally when video capture is over, release the video capture and destroyAllWindows

```

video_capture.release()
cv2.destroyAllWindows()

```

Further declarations and assigning values are required for the video and image input and output frames.

```

except:
    print("completed video")

def time9():
    os.startfile("yolo_detection_images6.py")
def time7():

```

```

import requests
import base64
import json
from glob import glob
import pandas as pd
import time
import os
def ocr(IMAGE_PATH):
    SECRET_KEY = 'sk_fa7d3dcec0363bdfb6ac3e06'
    with open(IMAGE_PATH, 'rb') as image_file:
        img_base64 = base64.b64encode(image_file.read())
    url =
'https://api.openalpr.com/v2/recognize_bytes?recognize_vehicle=1&country=ind&secret_key
=%s' % (SECRET_KEY) #Replace 'ind' with your country code
    r = requests.post(url, data = img_base64)
    try:
        return(r.json()['results'][0]['plate'])
    except:
        print("No number plate found")
l=[]
c=0
for fn in glob('LP/*.jpg'):
    print("processing",c)
    c+=1
    l.append(ocr(fn))
    if(c==3):
        break

l=set(l)
print(l)
for text in l:
    raw_data = {'date':[time.asctime( time.localtime(time.time()))],":[text]}
    #raw_data = [time.asctime( time.localtime(time.time()))],[text]
    df = pd.DataFrame(raw_data)
    df.to_csv('data.csv',mode='a')
    os.startfile('data.csv')

btn1=Button(window, text="Helmet input Video", command=time ,fg="blue" ,bg="orange"
,width=30 ,height=3 ,activebackground = "white" ,font=('times', 18, ' bold '))
btn1.place(x=0,y=150)

btn1=Button(window, text="Detect Persons", command=time2 ,fg="blue" ,bg="orange"
,width=30 ,height=3 ,activebackground = "white" ,font=('times', 18, ' bold '))
btn1.place(x=500,y=150)

btn1=Button(window, text="Detect Helmet", command=time1 ,fg="blue" ,bg="orange"
,width=30 ,height=3 ,activebackground = "white" ,font=('times', 18, ' bold '))
btn1.place(x=980,y=150)

```



```
btn1=Button(window, text="TripleRide input Video", command=time8 ,fg="blue"
,bg="White" ,width=30 ,height=3 ,activebackground = "white" ,font=('times', 18, ' bold '))
btn1.place(x=150,y=280)
```

```
btn1=Button(window, text="Detect Persons", command=time9 ,fg="blue" ,bg="White"
,width=30 ,height=3 ,activebackground = "white" ,font=('times', 18, ' bold '))
btn1.place(x=900,y=280)
```

```
btn1=Button(window, text="Detect NumberPlate", command=time6 ,fg="blue" ,bg="White"
,width=30 ,height=3 ,activebackground = "white" ,font=('times', 18, ' bold '))
btn1.place(x=150,y=450)
```

```
btn1=Button(window, text="Detect Text NumberPlate", command=time7 ,fg="blue"
,bg="White" ,width=30 ,height=3 ,activebackground = "white" ,font=('times', 18, ' bold '))
btn1.place(x=900,y=450)
```

```
btn1=Button(window, text="Detect Tripleride video", command=time3 ,fg="blue"
,bg="#00ff00" ,width=30 ,height=3 ,activebackground = "white" ,font=('times', 18, ' bold '))
btn1.place(x=0,y=580)
```

```
btn1=Button(window, text="Detect helmet video", command=time4 ,fg="blue"
,bg="#00ff00" ,width=30 ,height=3 ,activebackground = "white" ,font=('times', 18, ' bold '))
btn1.place(x=500,y=580)
```

```
btn1=Button(window, text="traffic Detection", command=time5 ,fg="blue" ,bg="#00ff00"
,width=30 ,height=3 ,activebackground = "white" ,font=('times', 18, ' bold '))
btn1.place(x=980,y=580)
```

```
window.mainloop()
```

3.2 Software Code

```
from tkinter import *
from tkinter import messagebox
top = Tk()

C = Canvas(top, bg="blue", height=250, width=300)
filename = PhotoImage(file = "resized_test.png")
background_label = Label(top, image=filename)
background_label.place(x=0, y=0, relwidth=1, relheight=1)

C.pack()

from tkinter import *
window=top
root=window
```

```

window.title("project1")
window.geometry("3500x900")
window.configure(background="#E4287C")

lbl=Label(window, text="Triple Ride and Helmet Detection" ,fg="black" ,width=25
,height=1,font=('times', 30, 'italic bold underline'))
lbl.place(x=400,y=10)

import os
def time():
    os.startfile("yolo_detection_webcam1.py")
def time1():
    os.startfile("Helmet_detection_YOLOV3.py")
def time2():
    os.startfile("yolo_detection_images.py")
def time3():
    os.startfile("yolo_detection_webcam.py")    #start python file
def time4():
    os.startfile("helmet.py")
def time5():
    os.startfile("live1.py")
def time6():
    os.startfile("yolo_detection_images4.py")
def time8():
    try:
        import numpy as np
        import cv2                                #importing libraries
        import cv2 as cv

        confidenceThreshold = 0.0
        NMSThreshold = 0.6

        modelConfiguration = 'cfg/yolov3.cfg'
        modelWeights = 'yolov3.weights'

        labelsPath = 'coco.names'
        labels = open(labelsPath).read().strip().split('\n')

        np.random.seed(10)
        COLORS = np.random.randint(0, 255, size=(len(labels), 3), dtype="uint8")

        net = cv2.dnn.readNetFromDarknet(modelConfiguration, modelWeights)

        outputLayer = net.getLayerNames()

        outputLayer = [outputLayer[i[0] - 1] for i in net.getUnconnectedOutLayers()]

```

```

video_capture = cv2.VideoCapture("om1.mp4")

(W, H) = (None, None)
count = 0
while True:
    ret, frame = video_capture.read()
    fram1=frame
    #frame = cv2.flip(frame, 1)
    if W is None or H is None:
        (H,W) = frame.shape[:2]

    blob = cv2.dnn.blobFromImage(frame, 1 / 255.0, (416, 416), swapRB = True, crop =
False)
    net.setInput(blob)
    layersOutputs = net.forward(outputLayer)

    boxes = []
    confidences = []
    classIDs = []

    for output in layersOutputs:
        for detection in output:
            scores = detection[5:]
            classID = np.argmax(scores)
            confidence = scores[classID]
            if confidence > confidenceThreshold:
                box = detection[0:4] * np.array([W, H, W, H])
                (centerX, centerY, width, height) = box.astype('int')
                x = int(centerX - (width/2))
                y = int(centerY - (height/2))

                boxes.append([x, y, int(width), int(height)])
                confidences.append(float(confidence))
                classIDs.append(classID)

    #Apply Non Maxima Suppression

    detectionNMS = cv2.dnn.NMSBoxes(boxes, confidences, confidenceThreshold,
NMSThreshold)

    if(len(detectionNMS) > 0):
        for i in detectionNMS.flatten():
            (x, y) = (boxes[i][0], boxes[i][1])
            (w, h) = (boxes[i][2], boxes[i][3])

            color = [int(c) for c in COLORS[classIDs[i]]]

            print(labels[classIDs[i]])
            if labels[classIDs[i]]=="motorbike": #since my detector only has 1 class
                cv2.imwrite("tripleride//framet%d.jpg" % count, frame[y-200:y+h, x-20:x+w])

```

```

        count = count + 1
    cv2.imshow('Output', frame)
    if(cv.waitKey(1) & 0xFF == ord('q')):
        break

#Finally when video capture is over, release the video capture and destroyAllWindows

video_capture.release()
cv2.destroyAllWindows()
except:
    print("completed video")

def time9():
    os.startfile("yolo_detection_images6.py")
def time7():
    import requests
    import base64
    import json
    from glob import glob
    import pandas as pd
    import time
    import os
    def ocr(IMAGE_PATH):
        SECRET_KEY = 'sk_fa7d3dcec0363bdfb6ac3e06'
        with open(IMAGE_PATH, 'rb') as image_file:
            img_base64 = base64.b64encode(image_file.read())
            url =
'https://api.openalpr.com/v2/recognize_bytes?recognize_vehicle=1&country=ind&secret_key
=%s' % (SECRET_KEY) #Replace 'ind' with your country code

        r = requests.post(url, data = img_base64)
        try:
            return(r.json()['results'][0]['plate'])
        except:
            print("No number plate found")
    l=[]
    c=0
    for fn in glob('LP/*.jpg'):
        print("processing",c)
        c+=1
        l.append(ocr(fn))
        if(c==3):
            break
    l=set(l)
    print(l)
    for text in l:
        raw_data = {'date':[time.asctime( time.localtime(time.time()))],":[text]}
        #raw_data = [time.asctime( time.localtime(time.time()))],[text]
        df = pd.DataFrame(raw_data)
        df.to_csv('data.csv',mode='a')

```

```
os.startfile('data.csv')
```

```
btn1=Button(window, text="Helmet input Video", command=time ,fg="blue" ,bg="orange"
,width=30 ,height=3 ,activebackground = "white" ,font=('times', 18, ' bold '))
btn1.place(x=0,y=150)
```

```
btn1=Button(window, text="Detect Persons", command=time2 ,fg="blue" ,bg="orange"
,width=30 ,height=3 ,activebackground = "white" ,font=('times', 18, ' bold '))
btn1.place(x=500,y=150)
```

```
btn1=Button(window, text="Detect Helmet", command=time1 ,fg="blue" ,bg="orange"
,width=30 ,height=3 ,activebackground = "white" ,font=('times', 18, ' bold '))
btn1.place(x=980,y=150)
```

```
btn1=Button(window, text="TripleRide input Video", command=time8 ,fg="blue"
,bg="White" ,width=30 ,height=3 ,activebackground = "white" ,font=('times', 18, ' bold '))
btn1.place(x=150,y=280)
```

```
btn1=Button(window, text="Detect Persons", command=time9 ,fg="blue" ,bg="White"
,width=30 ,height=3 ,activebackground = "white" ,font=('times', 18, ' bold '))
btn1.place(x=900,y=280)
```

```
btn1=Button(window, text="Detect NumberPlate", command=time6 ,fg="blue" ,bg="White"
,width=30 ,height=3 ,activebackground = "white" ,font=('times', 18, ' bold '))
btn1.place(x=150,y=450)
```

```
btn1=Button(window, text="Detect Text NumberPlate", command=time7 ,fg="blue"
,bg="White" ,width=30 ,height=3 ,activebackground = "white" ,font=('times', 18, ' bold '))
btn1.place(x=900,y=450)
```

```
btn1=Button(window, text="Detect Triplride video", command=time3 ,fg="blue"
,bg="#00ff00" ,width=30 ,height=3 ,activebackground = "white" ,font=('times', 18, ' bold '))
btn1.place(x=0,y=580)
```

```
btn1=Button(window, text="Detect helmet video", command=time4 ,fg="blue"
,bg="#00ff00" ,width=30 ,height=3 ,activebackground = "white" ,font=('times', 18, ' bold '))
btn1.place(x=500,y=580)
```

```
btn1=Button(window, text="traffic Detection", command=time5 ,fg="blue" ,bg="#00ff00"
,width=30 ,height=3 ,activebackground = "white" ,font=('times', 18, ' bold '))
btn1.place(x=980,y=580)
```

```
window.mainloop()
```

3.3 Implementation

3.3.1 Software details

3.3.1.1 Installing Python

Python is a widely used general-purpose, high-level programming language. Its syntax allows the programmers to express concepts in fewer lines of code when compared with other languages like C, C++ or java.

To get started installing Python, visit Python.org and downloaded the version 3.8.3. windows x86-64 executable installer by setting correct path. Then install the libraries pandas, requests, pillow, and OpenCV in command prompt by giving the command `pip install`.

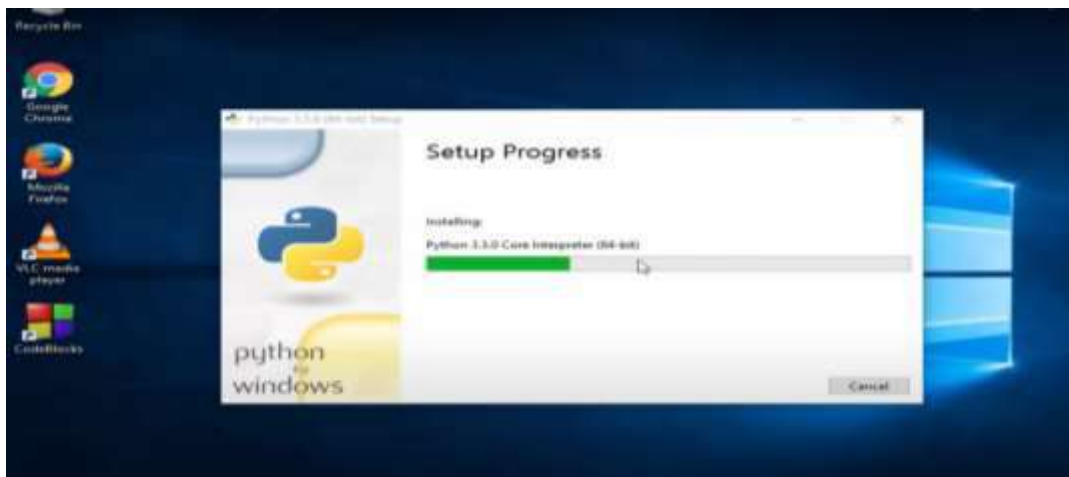


Fig 3.3.1.1a) Python Installing

3.3.1.2 OpenCV



Fig 3.3.1.2 b) OpenCV



Fig 3.3.1.2 c) OpenCV Detection

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

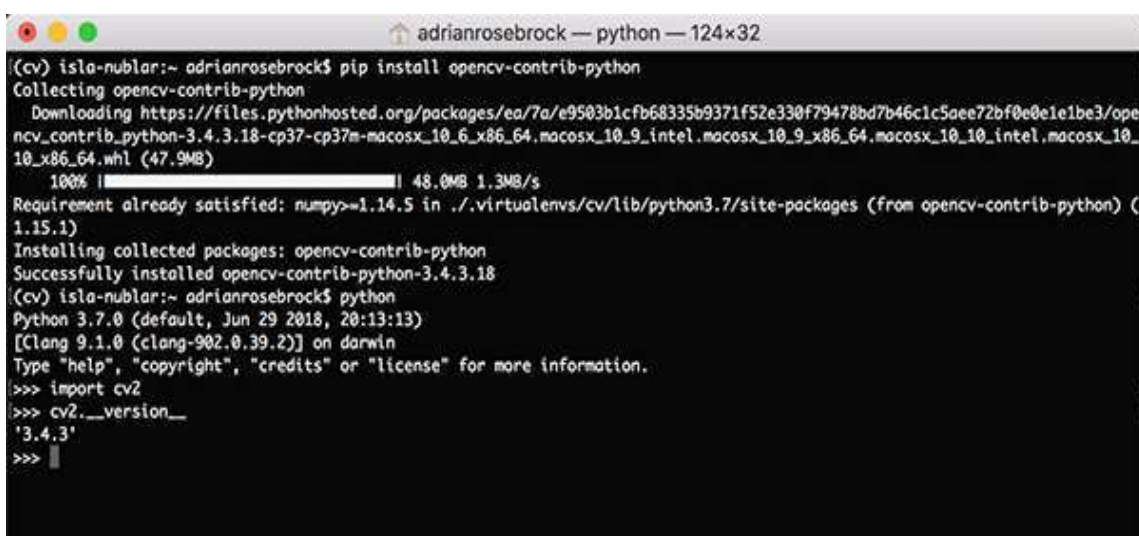
The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding [18 million](#). The library is used extensively in companies, research groups and by governmental bodies.

Along with well-established companies like Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota that employ the library, there are many startups such as Applied Minds, VideoSurf, and Zeitera, that make extensive use of OpenCV. OpenCV's deployed uses span

Detection of violation of traffic rules using Deep Learning the range from stitching streetview images together, detecting intrusions in surveillance video in Israel, monitoring mine equipment in China, helping robots navigate and pick up objects at Willow Garage, detection of swimming pool drowning accidents in Europe, running interactive art in Spain and New York, checking runways for debris in Turkey, inspecting labels on products in factories around the world on to rapid face detection in Japan.

It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, [Android](#) and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available. A full-featured [CUDA](#) and [OpenCL](#) interfaces are being actively developed right now. There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms. OpenCV is written natively in C++ and has a templated interface that works seamlessly with STL containers.

- Open CV is an image processing library created by Intel and later supported by Willow Garage and now maintained by Itseez.
- Available on Mac, Windows, Linux.
- Works in C, C++ and Python.
- Open Source and free.
- Easy to use and install.



```
(cv) isla-nublar:~ adrianrosebrock$ pip install opencv-contrib-python
Collecting opencv-contrib-python
  Downloading https://files.pythonhosted.org/packages/ea/7a/e9503b1cfb68335b9371f52e330f79478bd7b46c1c5aee72bf0e0e1e1be3/opencv_contrib_python-3.4.3.18-cp37-cp37m-macosx_10_6_x86_64.macosx_10_9_intel.macosx_10_9_x86_64.macosx_10_10_intel.macosx_10_10_x86_64.whl (47.9MB)
    100% |#####| 48.0MB 1.3MB/s
Requirement already satisfied: numpy>=1.14.5 in ./virtualenvs/cv/lib/python3.7/site-packages (from opencv-contrib-python) (1.15.1)
Installing collected packages: opencv-contrib-python
Successfully installed opencv-contrib-python-3.4.3.18
(cv) isla-nublar:~ adrianrosebrock$ python
Python 3.7.0 (default, Jun 29 2018, 20:13:13)
[Clang 9.1.0 (clang-902.0.39.2)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import cv2
>>> cv2.__version__
'3.4.3'
>>>
```

Fig 3.3.1.2 d) Installing OpenCV libraries in command prompt

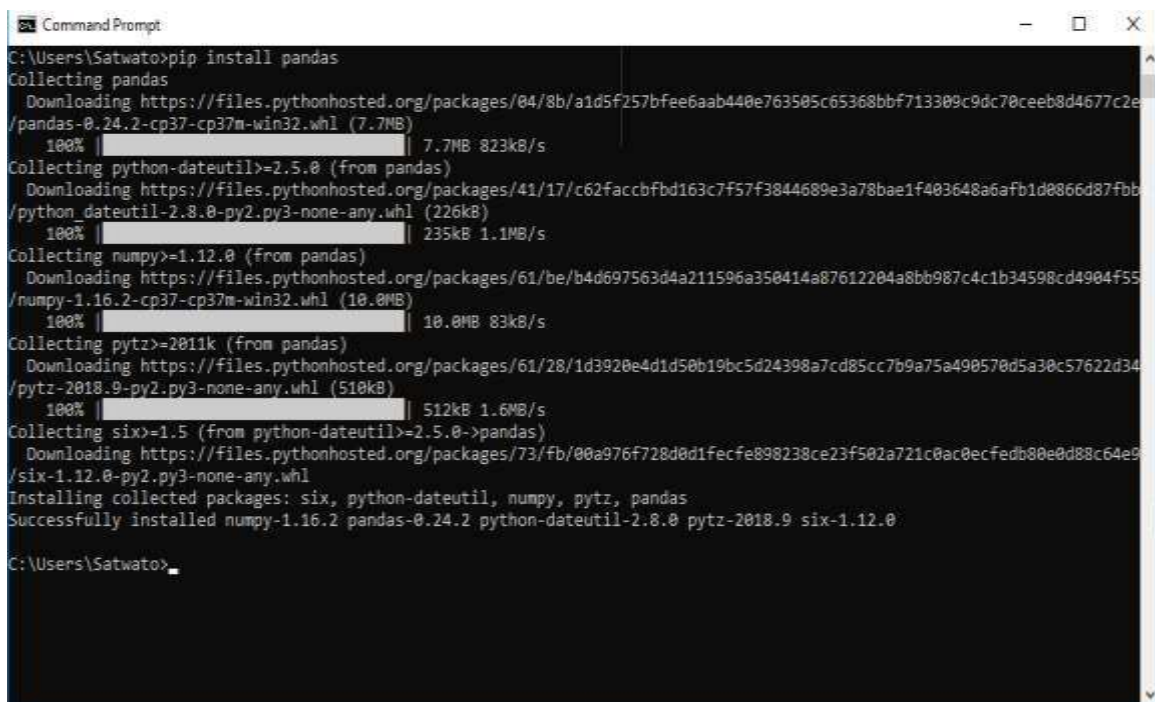
3.3.1.3 Pandas

Pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the [Python](#) programming language.

History of development

In 2008, *pandas* development began at [AQR Capital Management](#). By the end of 2009 it had been [open sourced](#), and is actively supported today by a community of like-minded individuals around the world who contribute their valuable time and energy to help make open source *pandas* possible. Thank you to [all of our contributors](#).

Since 2015, *pandas* is a [NumFOCUS sponsored project](#). This will help ensure the success of development of *pandas* as a world-class open-source project.



```

C:\Users\Satwato>pip install pandas
Collecting pandas
  Downloading https://files.pythonhosted.org/packages/04/8b/a1d5f257bfee6aab440e763505c65368bbf713309c9dc70ceeb8d4677c2e/pandas-0.24.2-cp37-cp37m-win32.whl (7.7MB)
    100% |#####| 7.7MB 823kB/s
Collecting python-dateutil<=2.5.0 (from pandas)
  Downloading https://files.pythonhosted.org/packages/41/17/c62facbfbfd163c7f57f38446089e3a78bae1f403648a6afb1d0866d87fbb/python_dateutil-2.8.0-py2.py3-none-any.whl (226kB)
    100% |#####| 235kB 1.1MB/s
Collecting numpy>=1.12.0 (from pandas)
  Downloading https://files.pythonhosted.org/packages/61/be/b4d697563d4a211596a350414a87612204a8bb987c4c1b34598cd4904f55/numpy-1.16.2-cp37-cp37m-win32.whl (10.0MB)
    100% |#####| 10.0MB 83kB/s
Collecting pytz>=2011k (from pandas)
  Downloading https://files.pythonhosted.org/packages/61/28/1d3920e4d1d50b19bc5d24398a7cd85cc7b9a75a490570d5a30c57622d34/pytz-2018.9-py2.py3-none-any.whl (510kB)
    100% |#####| 512kB 1.6MB/s
Collecting six>=1.5 (from python-dateutil<=2.5.0->pandas)
  Downloading https://files.pythonhosted.org/packages/73/fb/00a976f728d0d1fecfe898238ce23f502a721c0ac0ecfedb80e0d88c64e9/six-1.12.0-py2.py3-none-any.whl
Installing collected packages: six, python-dateutil, numpy, pytz, pandas
Successfully installed numpy-1.16.2 pandas-0.24.2 python-dateutil-2.8.0 pytz-2018.9 six-1.12.0
C:\Users\Satwato>
  
```

Fig 3.3.1.3 e) Installing Pandas libraries in command prompt

Timeline

- **2008:** Development of *pandas* started
- **2009:** *pandas* becomes open source
- **2012:** First edition of *Python for Data Analysis* is published
- **2015:** *pandas* becomes a [NumFOCUS sponsored project](#)
- **2018:** First in-person core developer sprint

Library Highlights

- A fast and efficient **DataFrame** object for data manipulation with integrated indexing;
- Tools for **reading and writing data** between in-memory data structures and different formats: CSV and text files, Microsoft Excel, SQL databases, and the fast HDF5 format;
- Intelligent **data alignment** and integrated handling of **missing data**: gain automatic label-based alignment in computations and easily manipulate messy data into an orderly form;
- Flexible **reshaping** and pivoting of data sets;
- Intelligent label-based **slicing, fancy indexing**, and **subsetting** of large data sets;
- Columns can be inserted and deleted from data structures for **size mutability**;
- Aggregating or transforming data with a powerful **group by** engine allowing split-apply-combine operations on data sets;
- High performance **merging and joining** of data sets;
- **Hierarchical axis indexing** provides an intuitive way of working with high-dimensional data in a lower-dimensional data structure;
- **Time series**-functionality: date range generation and frequency conversion, moving window statistics, date shifting and lagging. Even create domain-specific time offsets and join time series without losing data;
- Highly **optimized for performance**, with critical code paths written in [Cython](#) or C.
- Python with *pandas* is in use in a wide variety of **academic and commercial** domains, including Finance, Neuroscience, Economics, Statistics, Advertising, Web Analytics, and more.

3.3.1.4 Requests

- The requests library is the de facto standard for making HTTP requests in Python. It abstracts the complexities of making requests behind a beautiful, simple API so that you can focus on interacting with services and consuming data in your application.

```

(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\jeete>pip install requests
Defaulting to user installation because normal site-packages is not writeable
Collecting requests
  Using cached requests-2.24.0-py2.py3-none-any.whl (61 kB)
Collecting urllib3!=1.25.0,!1.25.1,<1.26,>=1.21.1
  Downloading urllib3-1.25.10-py2.py3-none-any.whl (127 kB)
    | 127 kB 409 kB/s
Collecting idna<3,>=2.5
  Using cached idna-2.10-py2.py3-none-any.whl (58 kB)
Collecting certifi>=2017.4.17
  Using cached certifi-2020.6.20-py2.py3-none-any.whl (156 kB)
Collecting chardet<4,>=3.0.2
  Using cached chardet-3.0.4-py2.py3-none-any.whl (133 kB)
Installing collected packages: urllib3, idna, certifi, chardet, requests
  WARNING: The script chardet.exe is installed in 'C:\Users\jeete\AppData\Roaming\Python\Python37\Scripts' which
  is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed certifi-2020.6.20 chardet-3.0.4 idna-2.10 requests-2.24.0 urllib3-1.25.10

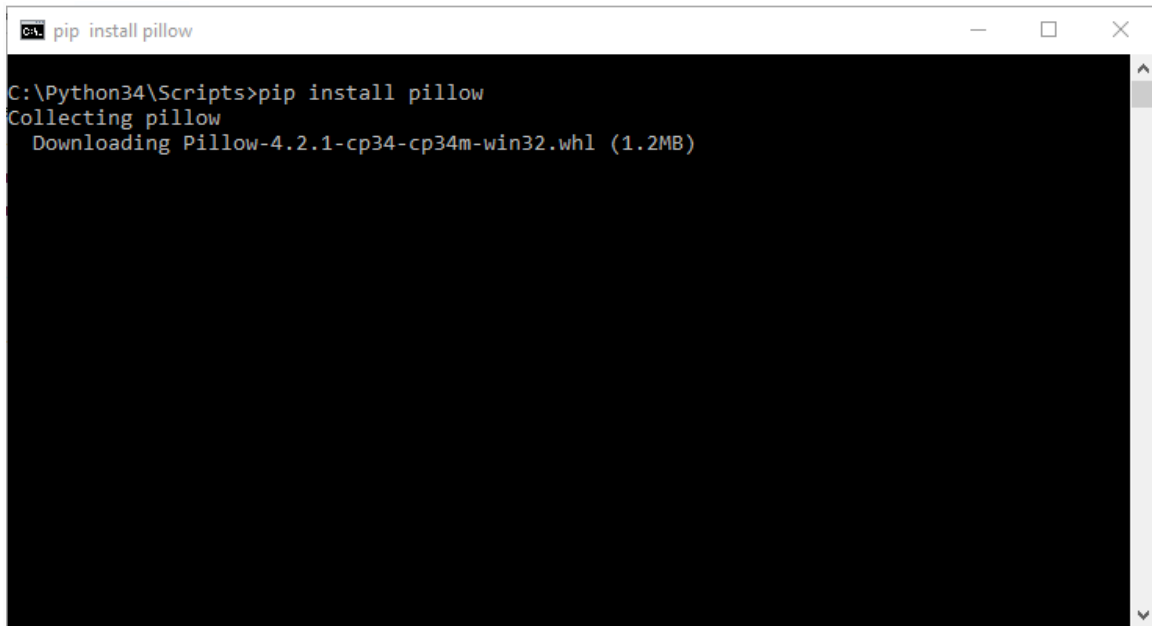
```

Fig 3.3.1.4 f) Installing Requests libraries in command prompt

3.3.1.5 Pillow

Pillow is the friendly PIL fork by [Alex Clark and Contributors](#). PIL is the Python Imaging Library by Fredrik Lundh and Contributors. The Python Imaging Library adds image processing capabilities to your Python interpreter.

This library provides extensive file format support, an efficient internal representation, and fairly powerful image processing capabilities. The core image library is designed for fast access to data stored in a few basic pixel formats. It should provide a solid foundation for a general image processing tool.



```
cmd - pip install pillow
C:\Python34\Scripts>pip install pillow
Collecting pillow
  Downloading Pillow-4.2.1-cp34-cp34m-win32.whl (1.2MB)
```

Fig 3.3.1.5 g) Installing Pillow libraries in command prompt

How to open and run the programs in Python

1.Initially install for the following

- OpenCV
- Pandas
- Requests
- Pillow

Step1 : Datasets

Datasets are downloaded first from the the Kaggle and make those datasets available for giving as input by adding them in correct directories.

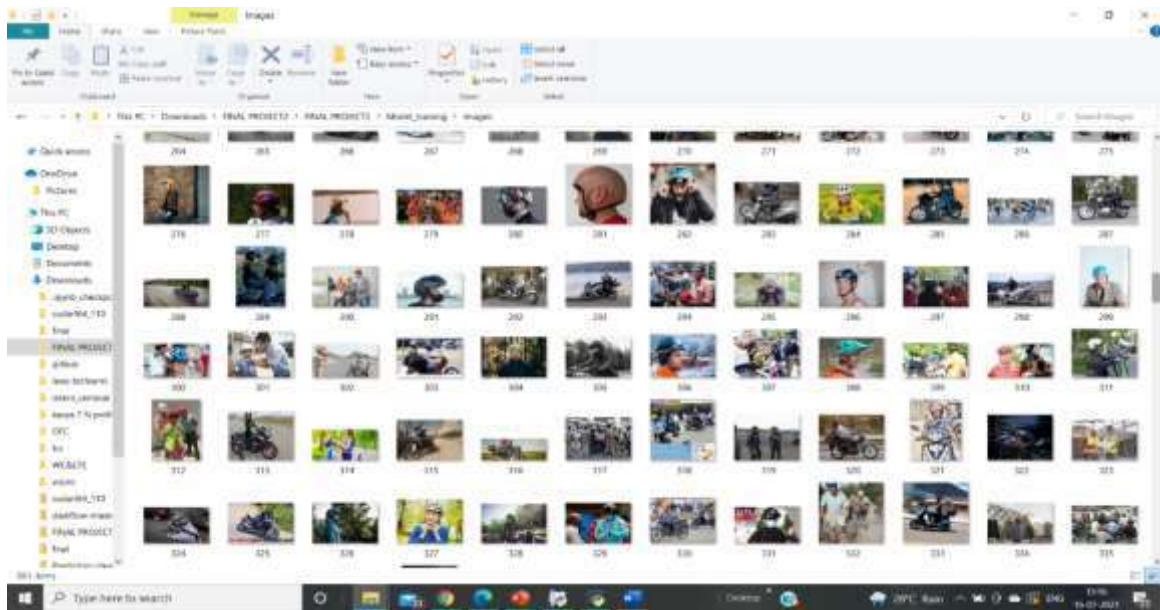


Fig 3.3.1.6 h) Datasets

Step2: Open the downloaded file which consists Python file in it

Open the python file that is downloaded, after opening the file we will get a new window as shown below.



Fig 3.3.1.7 i) Input and Output window

3.4.2 Flow Chart

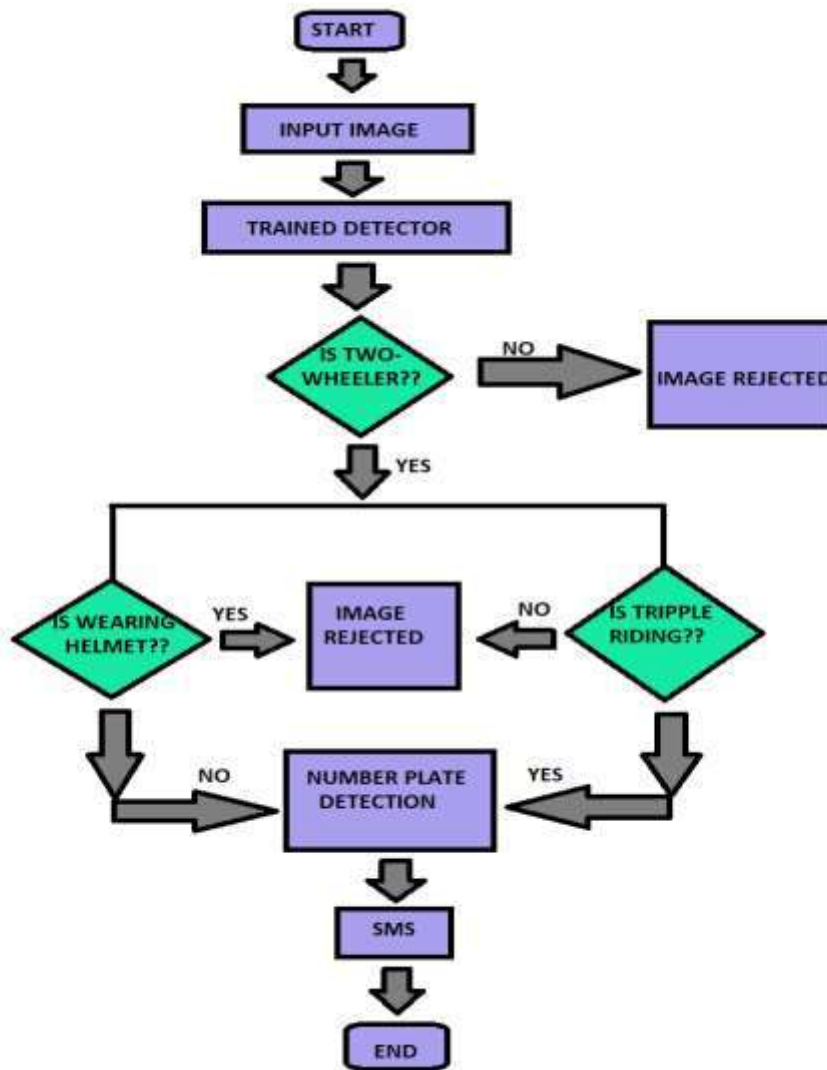


Fig 3.4.2 a) Flow diagram process of detecting the image

As shown in the flow diagram first we have to give image or video as input to the trained detector which is already trained by different inputs. And that detector will use the trained memory to detect the vehicle whether the vehicle is two wheeler or not. If the vehicle is two wheeler it will take two conditions, the first condition is detecting the person is wearing helmet or not and second condition is detecting the triple riding. If the person is wearing the helmet or not doing triple riding then the image will reject. If the person detected as not wearing helmet and doing triple riding it will automatically look in to the number plate of the vehicle and save it in the system with date, month, year and number in the number plate then authorities will look into it and take actions.

3.4.3 Algorithm

3.4.3 a) Grayscaleing

It is the process of converting an image from other color spaces e.g RGB, CMYK, HSV, etc. to shades of gray. It varies between complete black and complete white.

Importance of grayscaleing –

- Dimension reduction: For e.g. In RGB images there are three color channels and has three dimensions while grayscaled images are single dimensional.
- Reduces model complexity: neural network will need only 100 input node for grayscaled images.
- For other algorithms to work: There are many algorithms that are customized to work only on grayscaled images e.g. Canny edge detection function pre-implemented in OpenCV library works on Grayscaled images only.

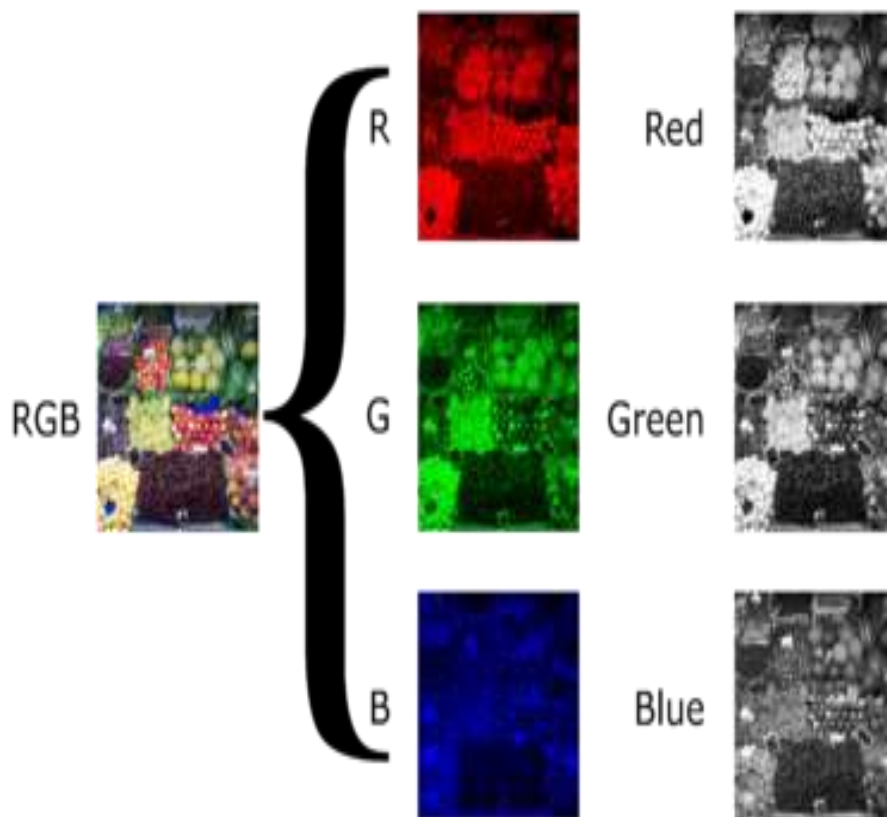


Fig 3.4.3 a) Grayscaleing Technique

3.4.3 b) YOLO Algorithm

YOLO algorithm is an algorithm based on regression, instead of selecting the interesting part of an Image, it predicts classes and bounding boxes for the whole image in one run of the Algorithm. Image classification is one of the many exciting applications of convolutional neural networks. Aside from simple image classification, there are plenty of fascinating problems in computer vision, with object detection being one of the most interesting. It is commonly associated with self-driving cars where systems blend computer vision, LIDAR and other technologies to generate a multidimensional representation of the road with all its participants. Object detection is also commonly used in video surveillance, especially in [crowd monitoring](#) to prevent terrorist attacks, count people for general statistics or analyze customer experience with walking paths within shopping centers.

- YOLO calculates the probabilities and bounding boxes for objects in a single forward pass.
- The way this is done is splitting the image into grids and detecting objects for each of the boxes.
- However, this is not done sequentially, and is actually implemented as a convolution operation.
- For example: if
- Input image size is (448,448,3)
- Number of classes to predict = 25 and Grid size = (7,7)

Then, each label vector will be of length 30 (Pc, Bx, By, Bw, Bh, +25 classes).

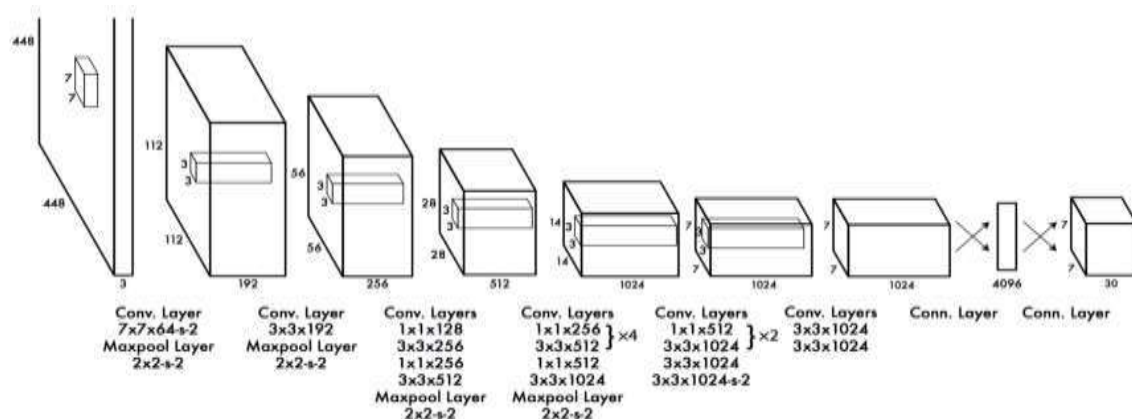


Fig 3.4.3 b) YOLO Algorithm

3.4.3 c) Non-max suppression Algorithm

Typical Object detection pipeline has one component for generating proposals for classification. Proposals are nothing but the candidate regions for the object of interest. Most of the approaches employ a sliding window over the feature map and assigns foreground/background scores depending on the features computed in that window. The neighbourhood windows have similar scores to some extent and are considered as candidate regions. This leads to hundreds of proposals. As the proposal generation method should have high recall, we keep loose constraints in this stage. However processing these many proposals all through the classification network is cumbersome. This leads to a technique which filters the proposals based on some criteria (which we will see soon) called Non-maximum Suppression.

NMS:

Input: A list of Proposal boxes B , corresponding confidence scores S and overlap threshold N .

Output: A list of filtered proposals D .

Algorithm:

1. Select the proposal with highest confidence score, remove it from B and add it to the final proposal list D . (Initially D is empty).
2. Now compare this proposal with all the proposals — calculate the IOU (Intersection over Union) of this proposal with every other proposal. If the IOU is greater than the threshold N , remove that proposal from B .
3. Again take the proposal with the highest confidence from the remaining proposals in B and remove it from B and add it to D .
4. Once again calculate the IOU of this proposal with all the proposals in B and eliminate the boxes which have high IOU than threshold.

5. This process is repeated until there are no more proposals left in B.

IOU calculation is actually used to measure the overlap between two proposals.

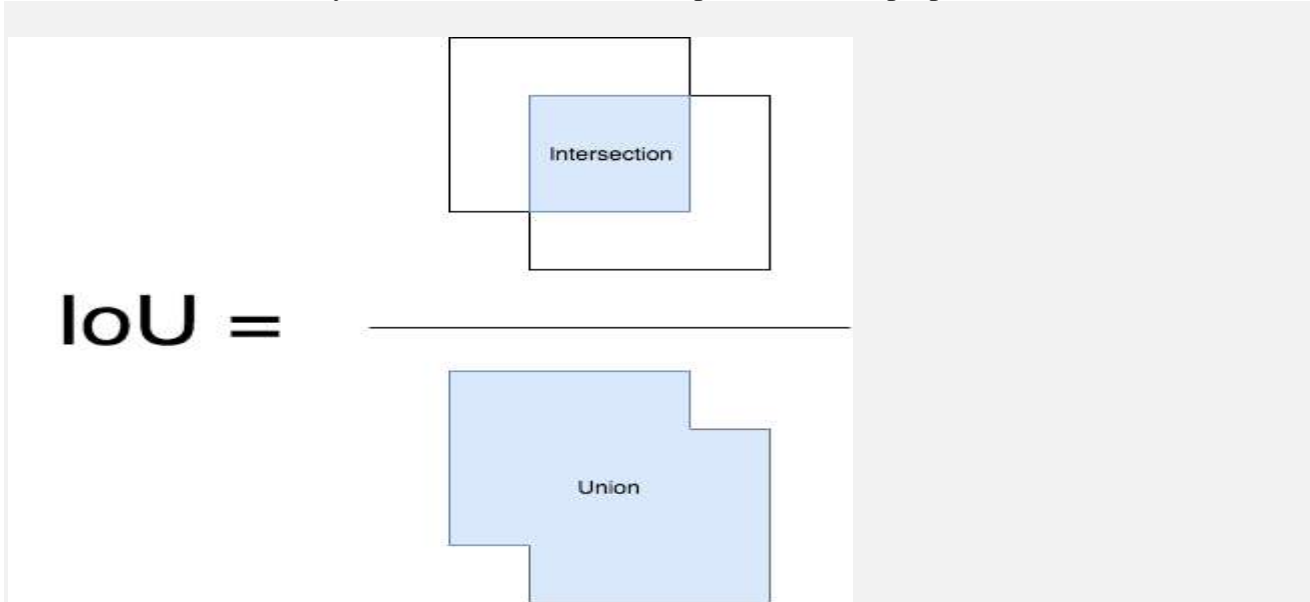


Fig 3.4.3 c) Intersection over Union



Fig 3.4.3 d) Non-max suppression

3.5 Summary

In this project, using Python. Coding for capturing the image and detection are done by python programming language. This project also uses Graphical User Interface(GUI). It enhanced the efficiency and ease of use for the underlying logical design of a stored program.

The frames from the system input video are processed in the openCV platform. And using the downloaded datasets as inputs it will detect the vehicles which are violating the rules and send the messages to respective authorities.

Chapter 4

RESULTS AND DISCUSSIONS

4.1.1 Helmet Input Video

If we Click on Helmet Input Video, it will analyse the video and it will give the output video and a window which is showing helmet detection in the video.

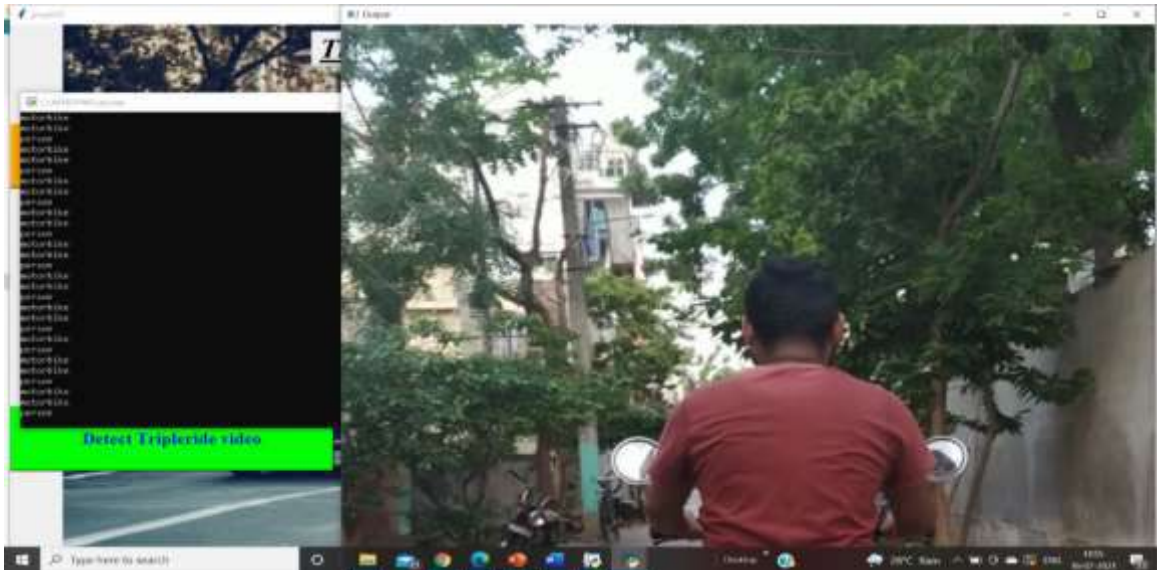


Fig 4.1.1 Helmet Input Video

4.1.2 Detect Persons

If we click on Detect person, it will detect the person in the vehicle and what vehicle he is riding after detecting it will give the output in the other window.

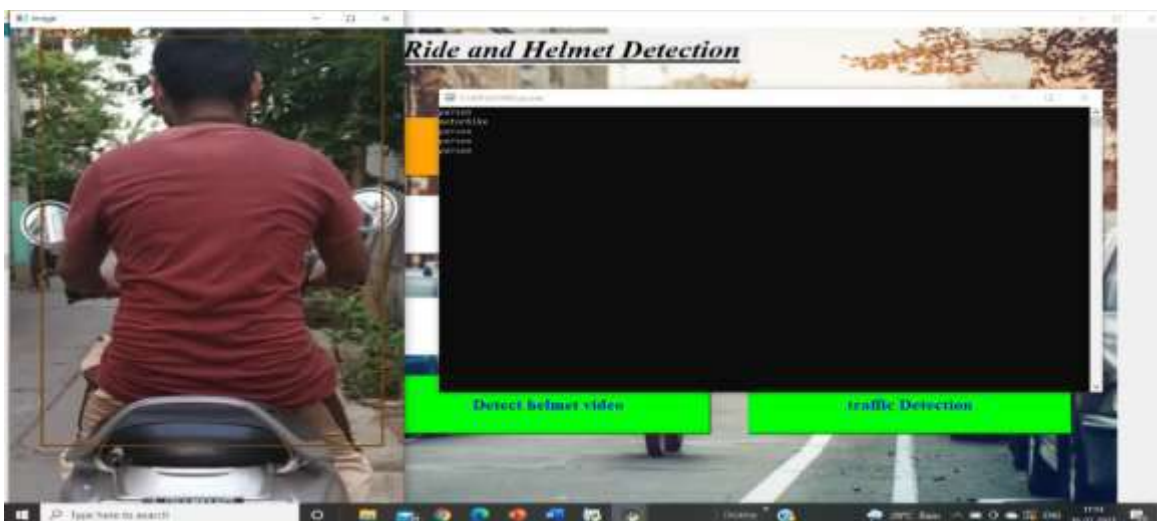


Fig 4.1.2 Detect persons

4.1.3 Detect Helmet

If we Click on Detect Helmet, it will analyse the video or image and it will give the output video and a window which is showing the output 0 that means there is no helmet detected in the video.

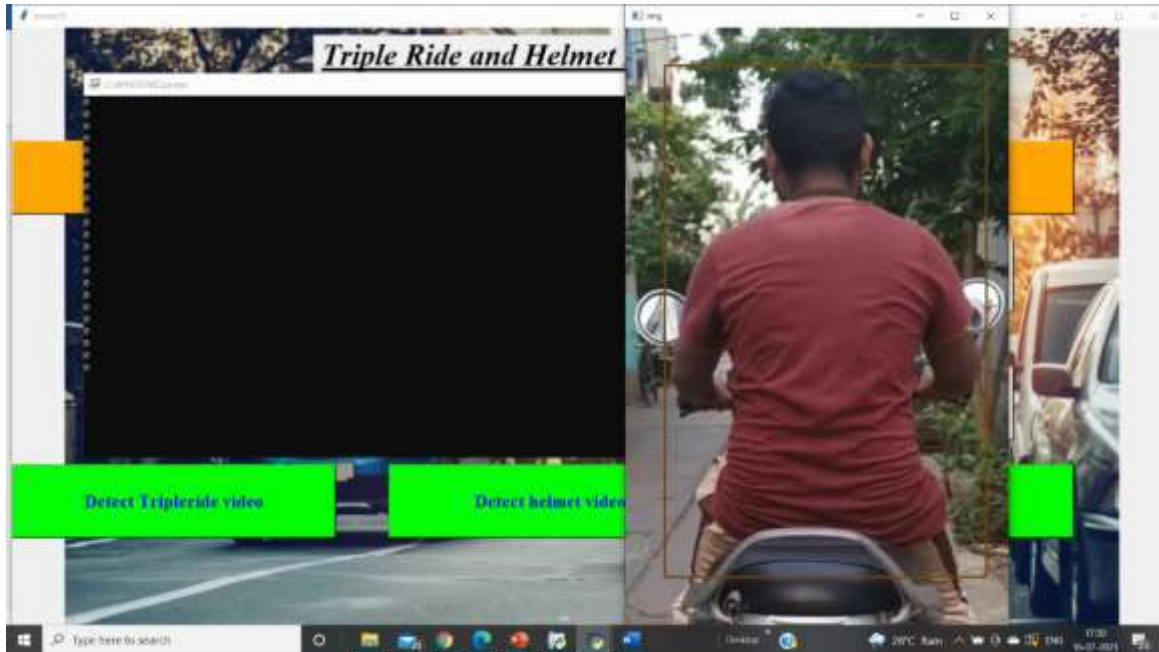


Fig 4.1.3 Detect Helmet

4.1.4 Triple Ride input Video

If we Click on Triple Ride input Video, it will analyse the video and it will give the output video and a window which is showing whether there is a single person riding the two wheeler or more than three are riding and what vehicle he is riding.

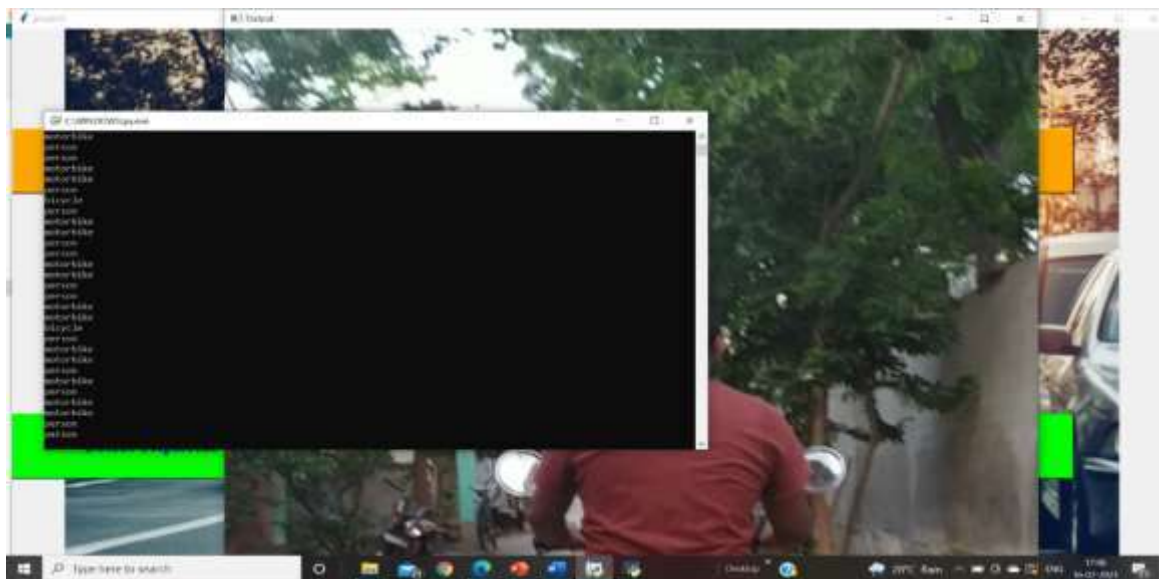


Fig 4.1.4 Triple Ride input Video

4.1.5 Detect Persons

It will detect the how many persons in which vehicle.

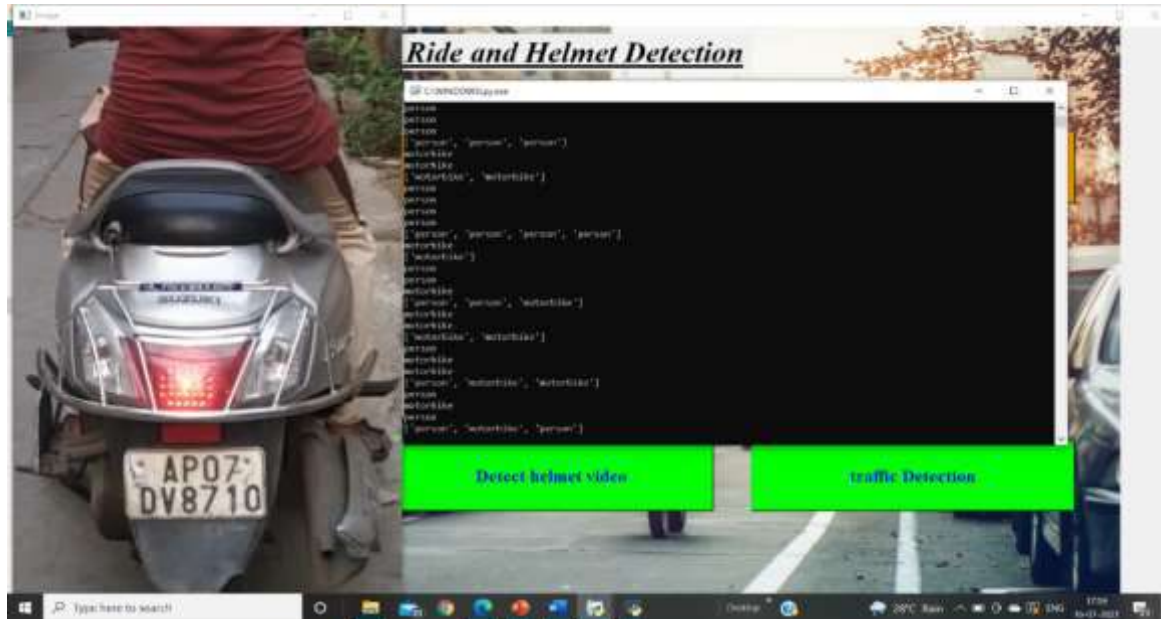


Fig 4.1.5 Detect Persons

4.1.6 Detect Triple ride video

If we Click on Detect Triple ride video, it will analyse the video and detect the number of persons in the video then it will give the output video and a window which is showing the output how many persons are riding the two wheeler and that means there is no helmet detected in the video.

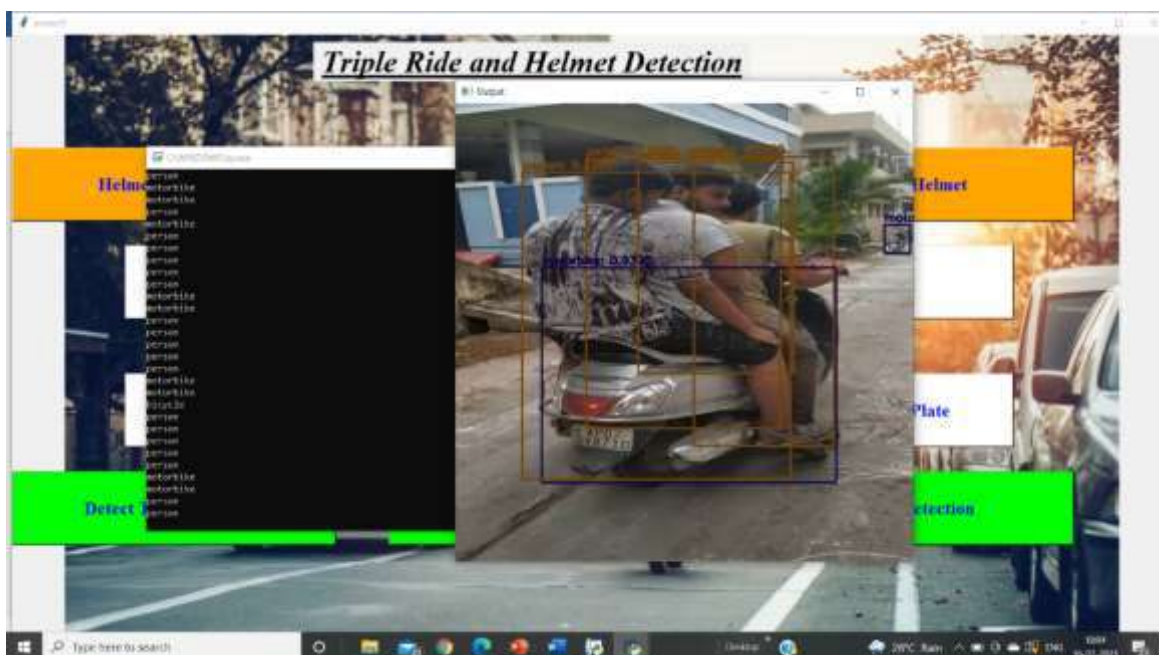


Fig 4.1.6 Detect Triple ride video

4.1.7 Detect Helmet video

Here we can see that how the system will detect the helmet in the video and give the output in the other window.

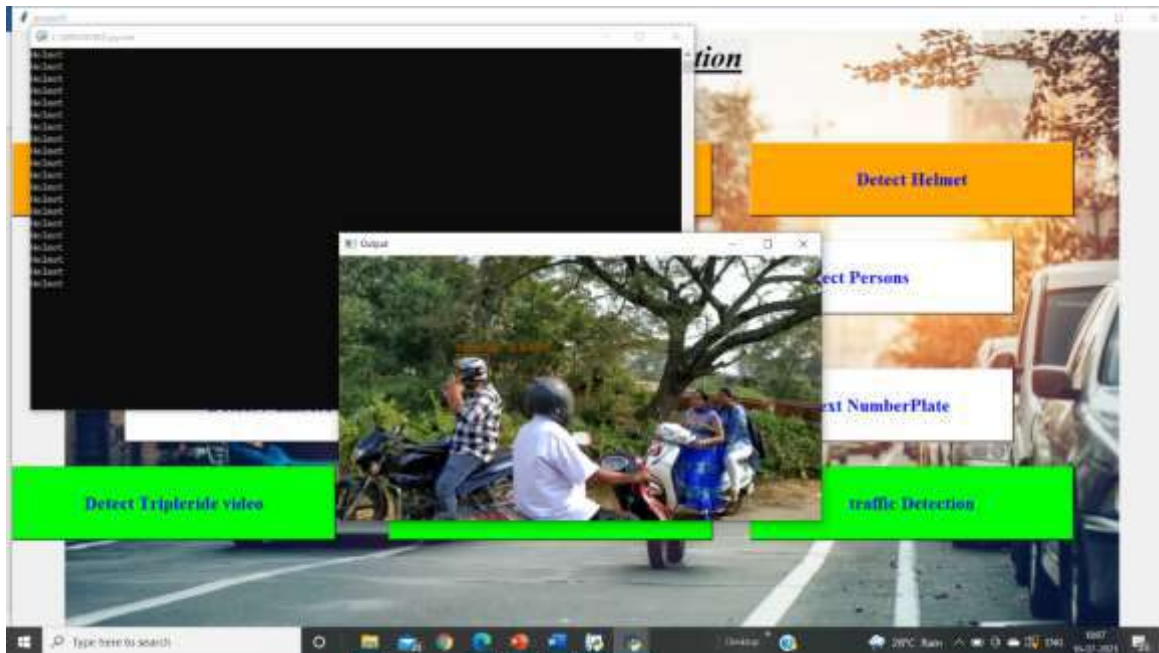


Fig 4.1.7 Detect Helmet video

4.1.8 Traffic Detection

It will detect the objects in the video and concentrates on two wheelers and the number plates of two wheelers, then save the data of two wheelers who are violating the rules.



Fig 4.1.8 Traffic Detection

Chapter 5

CONCLUSIONS AND FUTURE SCOPE

5.1 Conclusion

- As when compared to other algorithm YOLO is found be more advantageous and has higher efficiency and accuracy. Hence, we use the same approach to identify triple riding.
- Since we are implementing our project using YOLO algorithm which is a CNN based approach, results are obtained at fastest speed.

5.2 Future Scope

The objective of this project is to develop an application for enforcing helmet wearing using CCTV cameras. The developed application aims to help law enforcement by police, and eventually resulting in changing risk behaviours and consequently reducing the number of accidents and its severity. Conceptually, the application software implemented using Python language and OpenCV library uses two different angle of view CCTV cameras. Video frames recorded by the wide-angle CCTV camera are used to detect motorcyclists. If any motorcyclist without helmet is found, then the zoomed (narrow-angle) CCTV is activated to capture image of the violating motorcyclist and the motorcycle license plate in real time. Captured images are managed by database implemented using MySQL for ticket issuing.

REFERENCES

- Maharsh Desai , Shubham Khandelwal , Lokneesh Singh , Prof. Shilpa Gite **“Automatic Helmet Detection on Public Roads”**. IJETT 2016
- Amandeep Rathee, Krishnangini Kalita, Mahima Singh Deo **“Helmet detection on two-wheeler riders using machine learning”** 2017 International Conference.
- Aleksa Ćorović, Velibor Ilić, Siniša Đurić, Mališa Marijan, and Bogdan Pavković, **“The Real-Time Detection of Traffic Participants Using YOLO Algorithm”**2017 International Conference Paper, 2018 IEEE
- Rachel Huang, Jonathan Pedoeem, Cuixian Chen, **“YOLO-LITE: A Real-Time Object Detection Algorithm Optimized for Non-GPU Computers”**. Conference: 2018 IEEE International Conference on Big Data (Big Data).
- M. Prabu, Pooja Patil, Mallika Yadav, Nidhi Ranjan, **Detection of Non-Helmet Riders and Extraction of License Plate Number using Yolo v2 and OCR Method.**
- [Akanksha Soni](#), [Arun Pratap Singh](#) : 23 Feb. 2020 **“Automatic Motorcyclist Helmet Rule Violation Detection using Tensorflow & Keras in OpenCV”**.
- <https://ieeexplore.ieee.org/document/07960069>
- <https://www.sciencedirect.com/science/article/pii/S026322411830900X> ,2019
<https://patents.google.com/patent/US10339642B2/en> ,2019
- Pattern Recognition and Machine learning by Christopher Bishop in 2006
- Digital image processing by William K Pratt
- <https://www.youtube.com/playlist?list=PLS1QulWo1RIa7D1O6skqDQ-JZ1GGHKK-K>