**Source Code**

**from** **tkinter** **import** \*

**from** **tkinter** **import** messagebox

top = Tk()

Declare a variables and initializethe 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, widt, 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** #importing libraries

**import** **cv2** **as** **cv**

confidenceThreshold = **0.0**

NMSThreshold = **0.6**

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.

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()

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()