COMPUTER SCIENCE PROJECT- OPTICAL RECOGNISERS

2020-21

SUBJECT CODE: 083 KENDRIYA VIDYALAYA NO.2 TAMBARAM CHENNAI GROUP MEMBERS: A SATYANARAYANA M RITVIK PRASAD SUMIT KATIYAR

2020

A SATYANARAYANA

3/20/2020

ACKNOWLEDGEMENT

I Express my gratitude and sincere thanks to the **Principal I/C Mr. A J Rajendran** for his encouragement and all the facilities he has provided for the project work.

I sincerely appreciated his management by taking me into his fold for which I shall remain indebted to him. I extend my hearty thanks to **Ms Dhanashree Sontakke, PGT(CS)** who guided me to the successful completion of this project. I take this opportunity to express my deep sense of gratitude for her invaluable guidance constant encouragement immense motivation which has sustained my efforts at all the stages of this project work. I heartily thank my parents, my friends who have helped me and supported me in completing this project work successfully.

TABLE OF CONTENTS

|  |  |  |
| --- | --- | --- |
| **S. No.** | **Topic** | **Page No.** |
| 1 | Introduction | 4 |
| 2 | Software and Hardware requirements | 5 |
| 3 | Why Python? | 6 |
| 4 | Design Work | 8 |
| 5 | Algorithm | 9 |
| 6 | Code | 14 |
| 7 | Output | 35 |
| 8 | Advantages of Project | 39 |
| 9 | Further Development Areas | 41 |
| 10 | Bibliography | 43 |

**INTERNAL EXAMINER EXTERNAL EXAMINER**

**PRINCIPAL**

INTRODUCTION

 In our project , we have written programs on Face detection , Mask detection , Motion detection and Vehicle motion detection.

The purpose of our first program **Face detection program** is to detect a face which can be modified for security purposes like as passwords , or can be used for attendances in school , colleges , offices etc. The aim of our second program **Mask detection program** is to observe whether a person is wearing mask or not .If a person is wearing mask , it prints “Mask” on the screen and if not , it prints “No Mask” .We can use this program to ensure that people wear mask in crowded places like schools , offices , hospitals , malls , shops etc. The aim of our third program  **Motion detector** is to observe movements of a person which is very helpful to detect any kind of movement is restricted areas where normal people are not allowed to enter . It can also be used for security purposes and can be very helpful in night vision. The purpose of our fourth program **Vehicle motion detector** is to detect the motion of vehicles and also to count the total number of vehicles passing from a particular area . This program has many application like to check whether the vehicle in moving within the speed limit or not , or to check the acceleration or deceleration of vehicle. It can also be used to calculate the number of vehicles passing from an area in a particular time period and can be used to decide which road should be renovated .

For this project , we used  Python version 3.6(minimum) with modules like tensorflow , tkinter , cv2 , numpy , os , sys , datetime and keras and the hardware required for the execution of programs are cpu with atleast 35hz refresh rate , gpu(optional) , intel core i3(minimum) , camera ( built in or external) . For design work , we used the module tkinter and with functions like ImageTk and Image.

SOFTWARE AND HARDWARE REQUIREMENTS

* Hardware requirements:

1. CPU WITH ATLEAST 35Hz REFRESH RATE
2. GPU(OPTIONAL)
3. INTEL CORE i3(MINIMUM)
4. CAMERA ( BUILT IN OR EXTERNAL)

* Software requirements:

1. PYTHON 3.6(MINIMUM)
2. PYTHON MODULES:
   * 1. CV2
     2. NUMPY
     3. OS
     4. SYS
     5. DATETIME
     6. KERAS
     7. TENSORFLOW

WHY PYTHON?

**1) Readable and Maintainable Code**

While writing a software application, you must focus on the quality of its source code to simplify maintenance and updates. The syntax rules of Python allow you to express concepts without writing additional code. At the same time, Python, unlike other programming languages, emphasizes on code readability, and allows you to use English keywords instead of punctuations. Hence, you can use Python to build custom applications without writing additional code. The readable and clean code base will help you to maintain and update the software without putting extra time and effort.

**2) Compatible with Major Platforms and Systems**

At present, Python is supports many operating systems. You can even use Python interpreters to run the code on specific platforms and tools. Also, Python is an interpreted programming language. It allows you to you to run the same code on multiple platforms without recompilation. Hence, you are not required to recompile the code after making any alteration. You can run the modified application code without recompiling and check the impact of changes made to the code immediately. The feature makes it easier for you to make changes to the code without increasing development time.

**3) Robust Standard Library**

Its large and robust standard library makes Python score over other programming languages. The standard library allows you to choose from a wide range of modules according to your precise needs. Each module further enables you to add functionality to the Python application without writing additional code..

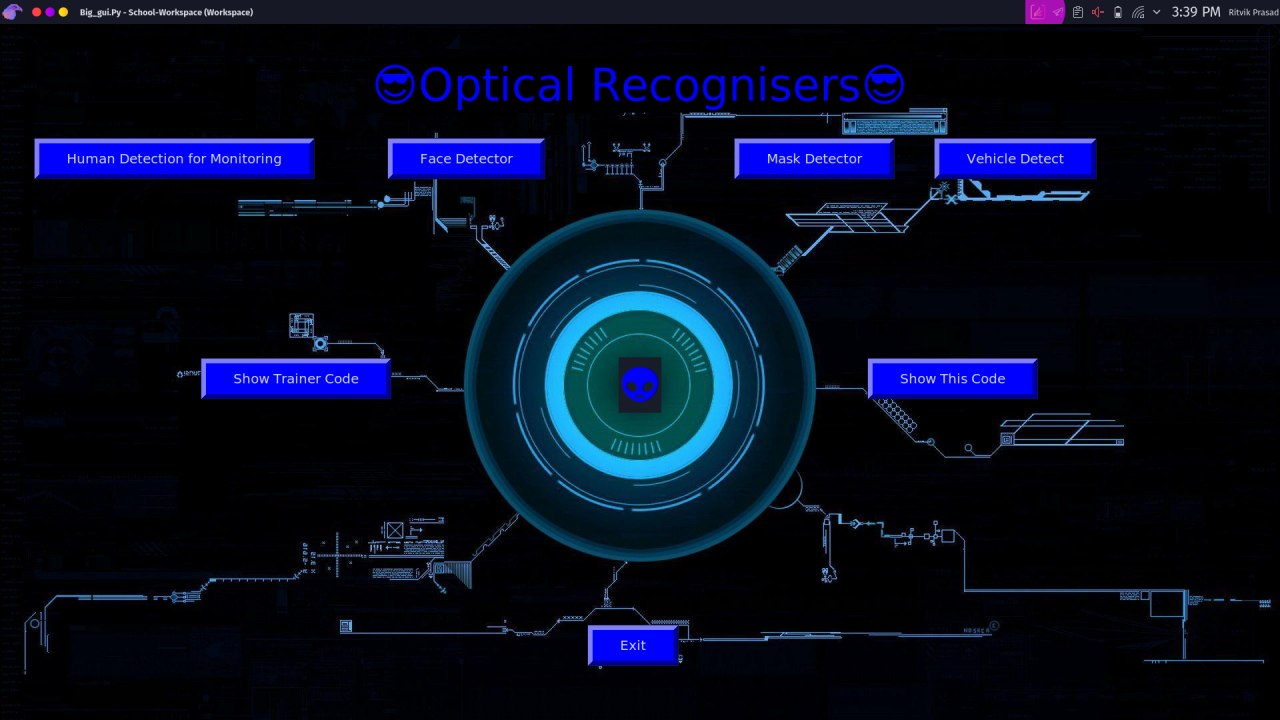
**4) Many Open Source Frameworks and Tools**

As an open source programming language, Python helps you to curtail software development cost significantly. You can even use several open source Python frameworks, libraries and development tools to curtail development time without increasing development cost. You even have option to choose from a wide range of open source Python frameworks and development tools according to your precise needs. For instance, you can simplify and speedup web application development by using robust Python web frameworks like Django, Flask, Pyramid, Bottle and Cherrypy. Likewise, you can accelerate desktop GUI application development using[**Python GUI frameworks**](http://www.allaboutweb.biz/python-gui-frameworks-usage/)and toolkits like PyQT, PyJs, PyGUI, Kivy, PyGTK and WxPython.

**5) Simplify Complex Software Development**

Python is a general purpose programming language. Hence, you can use the programming language for developing both desktop and web applications. Also, you can use Python for developing complex scientific and numeric applications. Python is designed with features to facilitate data analysis and visualization. You can take advantage of the data analysis features of Python to create custom big data solutions without putting extra time and effort. At the same time, the data visualization libraries and APIs provided by Python help you to visualize and present data in a more appealing and effective way. Many [**Python developers**](http://www.mindfiresolutions.com/python-development.htm) even use Python to accomplish artificial intelligence (AI) and natural language processing tasks.

DESIGN WORK



Tkinter is used as frontend.

ALGORITHM

1. import module tkinter as tk

2. define function facedetect as

2.1 import module cv2

2.2 from module cv2 import cv2

2.3 declare the cascade file as face cascade

2.4 set the frame parameters

2.5 capture each frame as cap

2.6 while frame is being captured

2.6.1 read the frame as frame

2.6.2 convert the frame to grayscale mode

2.6.3 detect the faces in the grayscale mode

2.6.4 for points in faces:

draw a rectange around each face using the coordinates

put text "face" above each rectangle

2.6.5 show the detected faces in the frame

2.6.6 if escape key is pressed

break

2.7 release the frames

2.8 destroyAllWindows

3 define function vehicle detect as

3.1 import module cv2

3.2 from cv2 import cv2

3.3 import numpy as np

3.4 from time import sleep

3.5 min1=80

3.6 min2=80

3.7 offset=6

3.8 position1=550

3.9 delay= 60

3.10 detection1 = []

3.11 number\_of\_veh= 0

3.12 define a function\_to\_find\_pos(x, y, w, h):

x1 = int(w / 2)

y1 = int(h / 2)

cx = x + x1

cy = y + y1

return cx,cy

3.13 declare the video to be used as cap

3.14 create a background seperation variable

3.15 while True:

read the frame as frame1

tempo = float(1/delay)

sleep(tempo)

convert the frame to grayscale

blur the grayscale using GaussianBlur

apply contour detection

dialate the frame using numpy arrays

use morphological ellpitical transformation on kernels of size 5

dialate the kernel

dialate the kernel data

find contours in the processed frame

declare a equation of line to monitor the vehicle detection

for countour in contours

draw a rectangle around each contour

if contour is not valid

continue

find the center of this rectangle

append this point to a list

use a pointer such as circle in the center of the rectangle

for each center point crossing the line equation

number\_of\_veh+=1

remove the rectangle

put text on the original frame indicating number of vehicles

show the frame

3.17 if escape key is pressed

break

3.18 release the frames

3.19 destroyAllWindows

4 define a function show\_this\_code as

4.1 text=text in this code

4.2 create a tkinter window

4.3 create a text window

4.4 insert this text in text window

4.5 pack this window

4.6 run this tkinter mainloop

5 def a function cctv capture as

5.1 import module cv2

5.2 from module cv2 import cv2

5.3 import module numpy as np

5.4 cam = vedio to detect or camera index

5.5 declare frame parameters

5.6 capture the frames in cam

5.7 read first frame as frame1

5.8 read second frame as frame2

5.9 decare a vediowriter variable if necessary

5.10 while frame is being captured

compute the difference between frame1 and frame2 and declare it as diff

convert the difference frame into grayscale

apply GaussianBluron the grayscale frame

apply a threshold of threshbinary

dilate this thresholded frame with three iterations

find thd contours in this frame

for contour in contours

draw a rectangle using bounding rect fucntion

if area of the contour is less than 500

continue

put rectangle around this contour

put text "DETECTED movement" at origin

put text "human detected" on top of each rectangle

5.11 show the frame1

5.12 frame1 = frame2

5.13 read frame2

5.14 if escape key is pressed

break

5.15 release the frames

5.16 destroyAllWindows

6 define a function mask\_detect as

6.1 import cv2

6.2 from cv2 import cv2

6.3 import numpy as np

6.4 from keras.models import load\_model

6.5 use the tensorflow trained model as model

6.6 results={0:'without mask',1:'mask'}

6.7 GR\_dict={0:(0,0,255),1:(0,255,0)}

6.8 rect\_size = 4

6.9 cap = cv2.VideoCapture(cam)

6.10 declare the cascade file as face cascade

6.11 set the frame parameters

6.12 while True:

capture each frame as cap

detect face using haarcascade detectmultiscale

for f in faces:

(x, y, w, h) = [v \* rect\_size for v in f]

face\_img = im[y:y+h, x:x+w]

rerect\_sized=cv2.resize(face\_img,(150,150))

normalized=rerect\_sized/255.0

reshaped=np.reshape(normalized,(1,150,150,3))

reshaped = np.vstack([reshaped])

result=model.predict(reshaped)

label=np.argmax(result,axis=1)[0]

cv2.rectangle(im,(x,y),(x+w,y+h),GR\_dict[label],2)

cv2.rectangle(im,(x,y-40),(x+w,y),GR\_dict[label],-1)

cv2.putText(im, results[label], (x, y-10),cv2.FONT\_HERSHEY\_SIMPLEX,0.8,(255,255,255),2)

show the frame

if escape key is pressed

break

6.13 release the frames

6.14 destroyAllWindows

7 define time1 as

7.1 import time

7.2 tim = time.strftime("%H:%M:%S %p")

7.3 tim = str(tim)

7.4 return tim

8 define function showcode\_model as

8.1 text=text in the tensorflow model training code

8.2 create a tkinter window

8.3 create a text window

8.4 insert this text in text window

8.5 pack this window

8.6 run this tkinter mainloop

#creating frontend using tkinter

9 create a tkinter window

10 give this window a title as python project

11 specify the geometry of the window as 1920\*1080

12 declare a variable to the image assigned as background to the window

13 create a label with the background

14 place this label at origin of the window

15

16 put text "😎Optical Recognisers😎" as the heading

17 my\_text.pack(pady=50)

18 create a button that would call the functio face\_detect on clicking

19 place this button at x=580,y=170

20 create a button that would call the functio cctv\_capture on clicking

21 place this button at x=50,y=170

22 my\_button4 = tk.Button(my\_label, text="Vehicle Detect" ,command=vehicledetect ,bg="blue", font=("Algerian" ,15), padx=40 , pady=10 , borderwidth=7)

23 place this button at x=1400,y=170

24 create a button that would call the functio mask\_detect on clicking

25 place this button at x=1100,y=170

26 create a button that would end the program on clicking

27 place this button at x=880,y=900

28 create a button that would call the functio showcode\_model on clicking

29 place this button at x=300,y=500

30 create a button that would call the functio show\_this\_code on clicking

31 place this button at x=1300,y=500

32 create a label to display at the center

33 run this as mainloop

34 Exit

CODE

import tkinter as tk

import cv2

from cv2 import cv2

def facedetect():

import cv2

from cv2 import cv2

face\_Cascade = cv2.CascadeClassifier("/home/ritvik/.workspace/opencv-master/data/haarcascades\_cuda/haarcascade\_frontalface\_default.xml")

#eye\_Cascade = cv2.CascadeClassifier("/home/ritvik/.workspace/opencv-master/data/haarcascades\_cuda/haarcascade\_eye.xml")

#body\_Cascade = cv2.CascadeClassifier("/home/ritvik/.workspace/opencv-master/data/haarcascades\_cuda/haarcascade\_lowerbody.xml")

#smile\_Cascade = cv2.CascadeClassifier("/home/ritvik/.workspace/opencv-master/data/haarcascades\_cuda/haarcascade\_smile.xml")

cap = cv2.VideoCapture(0)

cap.set(10, 100)

while cap.isOpened():

\_ret, frame = cap.read()

gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

faces = face\_Cascade.detectMultiScale(gray, 1.1, 4)

for (x,y,w,h) in faces:

#roiface\_gray = gray[y:y+h, x:x+h]

cv2.rectangle(frame, (x,y), (x+w, y+h), (0,255,0), 2)

cv2.putText(frame, "FACE", (x,y), cv2.FONT\_HERSHEY\_COMPLEX, 1, (0,255,0),2, 3)

cv2.imshow("face detect", frame)

if cv2.waitKey(40) == 27:

break

cap.release()

cv2.destroyAllWindows()

def vehicledetect():

import cv2

from cv2 import cv2

import numpy as np

from time import sleep

min1=80

min2=80

offset=6

position1=550

delay= 60

detection1 = []

var\_bla= 0

def function\_to\_find\_pos(x, y, w, h):

x1 = int(w / 2)

y1 = int(h / 2)

cx = x + x1

cy = y + y1

return cx,cy

cap = cv2.VideoCapture('/home/ritvik/.workspace/proj/project\_final/video.mp4')

subtracao = cv2.bgsegm.createBackgroundSubtractorMOG()

while True:

\_ret , frame1 = cap.read()

tempo = float(1/delay)

sleep(tempo)

grey = cv2.cvtColor(frame1,cv2.COLOR\_BGR2GRAY)

blur = cv2.GaussianBlur(grey,(3,3),5)

img\_sub = subtracao.apply(blur)

dilat = cv2.dilate(img\_sub,np.ones((5,5)))

kernel = cv2.getStructuringElement(cv2.MORPH\_ELLIPSE, (5, 5))

dilatada = cv2.morphologyEx (dilat, cv2. MORPH\_CLOSE , kernel)

dilatada = cv2.morphologyEx (dilatada, cv2. MORPH\_CLOSE , kernel)

contorno,h=cv2.findContours(dilatada,cv2.RETR\_TREE,cv2.CHAIN\_APPROX\_SIMPLE)

cv2.line(frame1, (25, position1), (1200, position1), (255,127,0), 3)

for(\_i,c) in enumerate(contorno):

(x,y,w,h) = cv2.boundingRect(c)

validar\_contorno = (w >= min1) and (h >= min2)

if not validar\_contorno:

continue

cv2.rectangle(frame1,(x,y),(x+w,y+h),(0,255,0),2)

centro = function\_to\_find\_pos(x, y, w, h)

detection1.append(centro)

cv2.circle(frame1, centro, 4, (0, 0,255), -1)

for (x,y) in detection1:

if y<(position1+offset) and y>(position1-offset):

var\_bla+=1

cv2.line(frame1, (25, position1), (1200, position1), (0,127,255), 3)

detection1.remove((x,y))

cv2.putText(frame1, "VEHICLE COUNT : "+str(var\_bla), (450, 70), cv2.FONT\_HERSHEY\_SIMPLEX, 2, (0, 0, 255),5)

cv2.imshow("Video Original" , frame1)

if cv2.waitKey(1) == 27:

break

cv2.destroyAllWindows()

cap.release()

def show\_this\_code():

text="""

import tkinter as tk

def facedetect():

import cv2

from cv2 import cv2

face\_Cascade = cv2.CascadeClassifier("/home/ritvik/.workspace/opencv-master/data/haarcascades\_cuda/haarcascade\_frontalface\_default.xml")

#eye\_Cascade = cv2.CascadeClassifier("/home/ritvik/.workspace/opencv-master/data/haarcascades\_cuda/haarcascade\_eye.xml")

#body\_Cascade = cv2.CascadeClassifier("/home/ritvik/.workspace/opencv-master/data/haarcascades\_cuda/haarcascade\_lowerbody.xml")

#smile\_Cascade = cv2.CascadeClassifier("/home/ritvik/.workspace/opencv-master/data/haarcascades\_cuda/haarcascade\_smile.xml")

cap = cv2.VideoCapture(0)

cap.set(10, 100)

while cap.isOpened():

\_ret, frame = cap.read()

gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

faces = face\_Cascade.detectMultiScale(gray, 1.1, 4)

for (x,y,w,h) in faces:

#roiface\_gray = gray[y:y+h, x:x+h]

cv2.rectangle(frame, (x,y), (x+w, y+h), (0,255,0), 2)

cv2.putText(frame, "FACE", (x,y), cv2.FONT\_HERSHEY\_COMPLEX, 1, (0,255,0),2, 3)

cv2.imshow("face detect", frame)

if cv2.waitKey(40) == 27:

break

cap.release()

cv2.destroyAllWindows()

def vehicledetect():

import cv2

from cv2 import cv2

import numpy as np

from time import sleep

min1=80

min2=80

offset=6

position1=550

delay= 60

detection1 = []

var\_bla= 0

def function\_to\_find\_pos(x, y, w, h):

x1 = int(w / 2)

y1 = int(h / 2)

cx = x + x1

cy = y + y1

return cx,cy

cap =cv2.VideoCapture('/home/ritvik/.workspace/proj/project\_final/video.mp4')

subtracao = cv2.bgsegm.createBackgroundSubtractorMOG()

while True:

\_ret , frame1 = cap.read()

tempo = float(1/delay)

sleep(tempo)

grey = cv2.cvtColor(frame1,cv2.COLOR\_BGR2GRAY)

blur = cv2.GaussianBlur(grey,(3,3),5)

img\_sub = subtracao.apply(blur)

dilat = cv2.dilate(img\_sub,np.ones((5,5)))

kernel = cv2.getStructuringElement(cv2.MORPH\_ELLIPSE, (5, 5))

dilatada = cv2.morphologyEx (dilat, cv2. MORPH\_CLOSE , kernel)

dilatada = cv2.morphologyEx (dilatada, cv2. MORPH\_CLOSE , kernel)

contorno,h=cv2.findContours(dilatada,cv2.RETR\_TREE,cv2.CHAIN\_APPROX\_SIMPLE)

cv2.line(frame1, (25, position1), (1200, position1), (255,127,0), 3)

for(\_i,c) in enumerate(contorno):

(x,y,w,h) = cv2.boundingRect(c)

validar\_contorno = (w >= min1) and (h >= min2)

if not validar\_contorno:

continue

cv2.rectangle(frame1,(x,y),(x+w,y+h),(0,255,0),2)

centro = function\_to\_find\_pos(x, y, w, h)

detection1.append(centro)

cv2.circle(frame1, centro, 4, (0, 0,255), -1)

for (x,y) in detection1:

if y<(position1+offset) and y>(position1-offset):

var\_bla+=1

cv2.line(frame1, (25, position1), (1200, position1), (0,127,255), 3)

detection1.remove((x,y))

cv2.putText(frame1, "VEHICLE COUNT : "+str(var\_bla), (450, 70), cv2.FONT\_HERSHEY\_SIMPLEX, 2, (0, 0, 255),5)

cv2.imshow("Video Original" , frame1)

if cv2.waitKey(1) == 27:

break

cv2.destroyAllWindows()

cap.release()

def show\_this\_code():

text=""

root = tk.Tk()

T = tk.Text(root, height=100, width=150)

T.pack()

T.insert(tk.END, text)

tk.mainloop()

def cctv\_capture():

import cv2

from cv2 import cv2

import numpy as np

cam = "/home/ritvik/.workspace/opencv-master/samples/data/vtest.avi"

cap = cv2.VideoCapture(cam)

cap.set(10, 20)

\_ret, frame1 = cap.read()

\_ret, frame2 = cap.read()

fourcc = cv2.VideoWriter\_fourcc(\*"XVID")

out = cv2.VideoWriter("/home/ritvik/workspace/detect.avi", fourcc, 60.0, (720,720))

while cap.isOpened():

#ret, frame = cap.read()

diff = cv2.absdiff(frame1, frame2)

gray = cv2.cvtColor(diff, cv2.COLOR\_BGR2GRAY)

blur = cv2.GaussianBlur(gray, (5,5), 0)

\_,thresh = cv2.threshold(blur, 20, 255, cv2.THRESH\_BINARY)

dilated = cv2.dilate(thresh, None, iterations=3)

contours, \_ = cv2.findContours(dilated, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)

for contour in contours:

#(x1 ,y1, w1, h1) = cv2.boundingRect(contour-1)

(x ,y, w, h) = cv2.boundingRect(contour)

#area = str(cv2.contourArea(contour))\

#print(area)

if cv2.contourArea(contour) < 500:

continue

#if abs(x-x1) < 20000:

# cv2.putText(frame1, "violation", (x,y), cv2.FONT\_HERSHEY\_COMPLEX\_SMALL, 1, (0,0,255),1)

cv2.rectangle(frame1, (x,y), (x+w, y+h), (0,255.0), 2)

cv2.putText(frame1, "Status: {}".format("Detected Moment !!!"), (10,20), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (0,0,255), 2)

cv2.putText(frame1, "HUMAN DETECTED", (x,y), cv2.FONT\_HERSHEY\_PLAIN, 1, (0,0,0), 2)

#if cv2.contourArea(contour) > 250 and cv2.contourArea(contour) < 600:

# cv2.putText(frame1,"VEHICLE",(x,y) ,cv2.FONT\_HERSHEY\_SIMPLEX ,0.5, (0,255,0), 2)

#cv2.drawContours(frame1, contours, -1, (0,255,0), 2)

out.write(frame2)

cv2.imshow("feed", frame1)

frame1 = frame2

\_ret, frame2 = cap.read()

if cv2.waitKey(40) == 27:

break

cap.release()

cv2.destroyAllWindows()

import time

tim = time.strftime("%H:%M:%S %p")

def mask\_detect(cam):

import cv2

from cv2 import cv2

import numpy as np

from keras.models import load\_model

model=load\_model("/home/ritvik/.workspace/proj/project\_final/mask.model")

#model=load\_model("/home/ritvik/.workspace/model2-003.model")

results={0:'without mask',1:'mask'}

GR\_dict={0:(0,0,255),1:(0,255,0)}

rect\_size = 4

cap = cv2.VideoCapture(cam)

haarcascade = cv2.CascadeClassifier("/home/ritvik/.workspace/opencv-master/data/haarcascades\_cuda/haarcascade\_frontalface\_default.xml")

while True:

(\_ret, im) = cap.read()

rerect\_size = cv2.resize(im, (im.shape[1] // rect\_size, im.shape[0] // rect\_size))

faces = haarcascade.detectMultiScale(rerect\_size)

for f in faces:

(x, y, w, h) = [v \* rect\_size for v in f]

face\_img = im[y:y+h, x:x+w]

rerect\_sized=cv2.resize(face\_img,(150,150))

normalized=rerect\_sized/255.0

reshaped=np.reshape(normalized,(1,150,150,3))

reshaped = np.vstack([reshaped])

result=model.predict(reshaped)

label=np.argmax(result,axis=1)[0]

cv2.rectangle(im,(x,y),(x+w,y+h),GR\_dict[label],2)

cv2.rectangle(im,(x,y-40),(x+w,y),GR\_dict[label],-1)

cv2.putText(im, results[label], (x, y-10),cv2.FONT\_HERSHEY\_SIMPLEX,0.8,(255,255,255),2)

cv2.putText(im, "PLEASE WEAR BLUE MASK FOR BEST RESULT", (4,73), cv2.FONT\_HERSHEY\_DUPLEX, 1, (0,0,255), 2)

#cv2.imshow('LIVE1', blur)

cv2.imshow('LIVE', im)

key = cv2.waitKey(10)

if key == 27:

break

cap.release()

cv2.destroyAllWindows()

def time1():

import time

global tim

tim = time.strftime("%H:%M:%S %p")

tim = str(tim)

return tim

def nothing():

pass

def showcode\_model():

text=""

from keras.optimizers import RMSprop

from keras.preprocessing.image import ImageDataGenerator

import cv2

from keras.models import Sequential

from keras.layers import Conv2D, Input, ZeroPadding2D, BatchNormalization, Activation, MaxPooling2D, Flatten, Dense,Dropout

from keras.models import Model, load\_model

from keras.callbacks import TensorBoard, ModelCheckpoint

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import f1\_score

from sklearn.utils import shuffle

import tensorflow as tf

import imutils

import numpy as np

model = Sequential([

Conv2D(100, (3,3), activation='relu', input\_shape=(150, 150, 3)),

MaxPooling2D(2,2),

Conv2D(100, (3,3), activation='relu'),

MaxPooling2D(2,2),

Flatten(),

Dropout(0.5),

Dense(50, activation='relu'),

Dense(2, activation='softmax')

])

model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['acc'])

TRAINING\_DIR = "/home/ritvik/.workspace/proj/project\_final/face-mask-dataset/Dataset/train"

train\_datagen = ImageDataGenerator(rescale=1.0/255,

rotation\_range=40,

width\_shift\_range=0.2,

height\_shift\_range=0.2,

shear\_range=0.2,

zoom\_range=0.2,

horizontal\_flip=True,

fill\_mode='nearest')

train\_generator = train\_datagen.flow\_from\_directory(TRAINING\_DIR,

batch\_size=10,

target\_size=(150, 150))

VALIDATION\_DIR = "/home/ritvik/.workspace/proj/project\_final/face-mask-dataset/Dataset/test"

validation\_datagen = ImageDataGenerator(rescale=1.0/255)

validation\_generator = validation\_datagen.flow\_from\_directory(VALIDATION\_DIR,

batch\_size=10,

target\_size=(150, 150))

checkpoint = ModelCheckpoint('model2-{epoch:03d}.model',monitor='val\_loss',verbose=0,save\_best\_only=True,mode='auto')

history = model.fit\_generator(train\_generator,

epochs=10,

validation\_data=validation\_generator,

callbacks=[checkpoint])

""

root = tk.Tk()

T = tk.Text(root, height=100, width=150)

T.pack()

T.insert(tk.END, text)

#my\_button5 = tk.Button(my\_label, text="Exit",command=root.quit, bg="blue",font=("Algerian" ,15), padx=40 , pady=10 , borderwidth=7)

#my\_button5.grid(row=0, column=3, padx=10 , pady=250 )

#my\_button5.place(x=880,y=900)

tk.mainloop()

root = tk.Tk()

root.title("PYTHON PROJECT")

root.geometry("1920x1080")

bg = tk.PhotoImage(file="/home/ritvik/Pictures/wp1913251.png")

my\_label = tk.Label(root, image=bg)

my\_label.place(x=0, y=0, relwidth=1, relheight=1)

my\_text = tk.Label(root, text="😎Optical Recognisers😎", font=("Jocker", 50), fg="blue",bg="BLACK")

my\_text.pack(pady=50)

my\_button2 = tk.Button(my\_label, text="Face Detector" ,command=facedetect , bg="blue", font=("Algerian" ,15) , padx=40 , pady=10 , borderwidth=7)

my\_button2.grid(row=0, column=1, padx=10 , pady=250)

my\_button2.place(x=580,y=170)

my\_button3 = tk.Button(my\_label, text="Human Detection for Monitoring" ,command=cctv\_capture ,bg="blue", font=("Algerian" ,15), padx=40 , pady=10 , borderwidth=7)

my\_button3.grid(row=0,column=5, padx=10 , pady=250)

my\_button3.place(x=50,y=170)

my\_button4 = tk.Button(my\_label, text="Vehicle Detect" ,command=vehicledetect ,bg="blue", font=("Algerian" ,15), padx=40 , pady=10 , borderwidth=7)

my\_button4.grid(row=0,column=5, padx=10 , pady=250)

my\_button4.place(x=1400,y=170)

my\_button4 = tk.Button(my\_label, text="Mask Detector" ,command=nothing ,bg="blue", font=("Algerian" ,15), padx=40 , pady=10 , borderwidth=7)

my\_button4.grid(row=0,column=5, padx=10 , pady=250)

my\_button4.place(x=1100,y=170)

my\_button5 = tk.Button(my\_label, text="Exit",command=root.quit, bg="blue",font=("Algerian" ,15), padx=40 , pady=10 , borderwidth=7)

my\_button5.grid(row=0, column=3, padx=10 , pady=250 )

my\_button5.place(x=880,y=900)

my\_button6 = tk.Button(my\_label, text="Show Trainer Code",command=showcode\_model, bg="blue",font=("Algerian" ,15), padx=40 , pady=10 , borderwidth=7)

my\_button6.grid(row=0, column=3, padx=10 , pady=250 )

my\_button6.place(x=300,y=500)

my\_button7 = tk.Button(my\_label, text="Show This Code",command=showcode\_model, bg="blue",font=("Algerian" ,15), padx=40 , pady=10 , borderwidth=7)

my\_button7.grid(row=0, column=3, padx=10 , pady=250 )

my\_button7.place(x=1300,y=500)

my\_text2 = tk.Label(root, text='👽', font=("Jocker", 50), fg="blue")

my\_text2.place(x=928, y=500)

#my\_text1 = tk.Label(root, text='HELLO WORLD', font=("Jocker", 50), fg="blue")

#my\_text1.place(x=600, y=600)

root.mainloop()

"""

root = tk.Tk()

T = tk.Text(root, height=100, width=150)

T.pack()

T.insert(tk.END, text)

tk.mainloop()

exit()

def cctv\_capture():

import cv2

from cv2 import cv2

import numpy as np

cam = "/home/ritvik/.workspace/opencv-master/samples/data/vtest.avi"

cap = cv2.VideoCapture(cam)

cap.set(10, 20)

\_ret, frame1 = cap.read()

\_ret, frame2 = cap.read()

fourcc = cv2.VideoWriter\_fourcc(\*"XVID")

out = cv2.VideoWriter("/home/ritvik/workspace/detect.avi", fourcc, 60.0, (720,720))

while cap.isOpened():

#ret, frame = cap.read()

diff = cv2.absdiff(frame1, frame2)

gray = cv2.cvtColor(diff, cv2.COLOR\_BGR2GRAY)

blur = cv2.GaussianBlur(gray, (5,5), 0)

\_,thresh = cv2.threshold(blur, 20, 255, cv2.THRESH\_BINARY)

dilated = cv2.dilate(thresh, None, iterations=3)

contours, \_ = cv2.findContours(dilated, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)

for contour in contours:

#(x1 ,y1, w1, h1) = cv2.boundingRect(contour-1)

(x ,y, w, h) = cv2.boundingRect(contour)

#area = str(cv2.contourArea(contour))\

#print(area)

if cv2.contourArea(contour) < 500:

continue

#if abs(x-x1) < 20000:

# cv2.putText(frame1, "violation", (x,y), cv2.FONT\_HERSHEY\_COMPLEX\_SMALL, 1, (0,0,255),1)

cv2.rectangle(frame1, (x,y), (x+w, y+h), (0,255.0), 2)

cv2.putText(frame1, "Status: {}".format("Detected Moment !!!"), (10,20), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (0,0,255), 2)

cv2.putText(frame1, "HUMAN DETECTED", (x,y), cv2.FONT\_HERSHEY\_PLAIN, 1, (0,0,0), 2)

#if cv2.contourArea(contour) > 250 and cv2.contourArea(contour) < 600:

# cv2.putText(frame1,"VEHICLE",(x,y) ,cv2.FONT\_HERSHEY\_SIMPLEX ,0.5, (0,255,0), 2)

#cv2.drawContours(frame1, contours, -1, (0,255,0), 2)

out.write(frame2)

cv2.imshow("feed", frame1)

frame1 = frame2

\_ret, frame2 = cap.read()

if cv2.waitKey(40) == 27:

break

cap.release()

cv2.destroyAllWindows()

import time

tim = time.strftime("%H:%M:%S %p")

def mask\_detect(cam):

import cv2

from cv2 import cv2

import numpy as np

from keras.models import load\_model

model=load\_model("/home/ritvik/.workspace/proj/project\_final/mask.model")

#model=load\_model("/home/ritvik/.workspace/model2-003.model")

results={0:'without mask',1:'mask'}

GR\_dict={0:(0,0,255),1:(0,255,0)}

rect\_size = 4

cap = cv2.VideoCapture(cam)

haarcascade = cv2.CascadeClassifier("/home/ritvik/.workspace/opencv-master/data/haarcascades\_cuda/haarcascade\_frontalface\_default.xml")

while True:

(\_ret, im) = cap.read()

rerect\_size = cv2.resize(im, (im.shape[1] // rect\_size, im.shape[0] // rect\_size))

faces = haarcascade.detectMultiScale(rerect\_size)

for f in faces:

(x, y, w, h) = [v \* rect\_size for v in f]

face\_img = im[y:y+h, x:x+w]

rerect\_sized=cv2.resize(face\_img,(150,150))

normalized=rerect\_sized/255.0

reshaped=np.reshape(normalized,(1,150,150,3))

reshaped = np.vstack([reshaped])

result=model.predict(reshaped)

label=np.argmax(result,axis=1)[0]

cv2.rectangle(im,(x,y),(x+w,y+h),GR\_dict[label],2)

cv2.rectangle(im,(x,y-40),(x+w,y),GR\_dict[label],-1)

cv2.putText(im, results[label], (x, y-10),cv2.FONT\_HERSHEY\_SIMPLEX,0.8,(255,255,255),2)

cv2.putText(im, "PLEASE WEAR BLUE MASK FOR BEST RESULT", (4,73), cv2.FONT\_HERSHEY\_DUPLEX, 1, (0,0,255), 2)

#cv2.imshow('LIVE1', blur)

cv2.imshow('LIVE', im)

key = cv2.waitKey(10)

if key == 27:

break

cap.release()

cv2.destroyAllWindows()

def time1():

import time

global tim

tim = time.strftime("%H:%M:%S %p")

tim = str(tim)

return tim

def nothing():

pass

def showcode\_model():

text="""

from keras.optimizers import RMSprop

from keras.preprocessing.image import ImageDataGenerator

import cv2

from keras.models import Sequential

from keras.layers import Conv2D, Input, ZeroPadding2D, BatchNormalization, Activation, MaxPooling2D, Flatten, Dense,Dropout

from keras.models import Model, load\_model

from keras.callbacks import TensorBoard, ModelCheckpoint

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import f1\_score

from sklearn.utils import shuffle

import tensorflow as tf

import imutils

import numpy as np

model = Sequential([

Conv2D(100, (3,3), activation='relu', input\_shape=(150, 150, 3)),

MaxPooling2D(2,2),

Conv2D(100, (3,3), activation='relu'),

MaxPooling2D(2,2),

Flatten(),

Dropout(0.5),

Dense(50, activation='relu'),

Dense(2, activation='softmax')

])

model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['acc'])

TRAINING\_DIR = "/home/ritvik/.workspace/proj/project\_final/face-mask-dataset/Dataset/train"

train\_datagen = ImageDataGenerator(rescale=1.0/255,

rotation\_range=40,

width\_shift\_range=0.2,

height\_shift\_range=0.2,

shear\_range=0.2,

zoom\_range=0.2,

horizontal\_flip=True,

fill\_mode='nearest')

train\_generator = train\_datagen.flow\_from\_directory(TRAINING\_DIR,

batch\_size=10,

target\_size=(150, 150))

VALIDATION\_DIR = "/home/ritvik/.workspace/proj/project\_final/face-mask-dataset/Dataset/test"

validation\_datagen = ImageDataGenerator(rescale=1.0/255)

validation\_generator = validation\_datagen.flow\_from\_directory(VALIDATION\_DIR,

batch\_size=10,

target\_size=(150, 150))

checkpoint = ModelCheckpoint('model2-{epoch:03d}.model',monitor='val\_loss',verbose=0,save\_best\_only=True,mode='auto')

history = model.fit\_generator(train\_generator,

epochs=10,

validation\_data=validation\_generator,

callbacks=[checkpoint])

"""

root = tk.Tk()

T = tk.Text(root, height=100, width=150)

T.pack()

T.insert(tk.END, text)

#my\_button5 = tk.Button(my\_label, text="Exit",command=root.quit, bg="blue",font=("Algerian" ,15), padx=40 , pady=10 , borderwidth=7)

#my\_button5.grid(row=0, column=3, padx=10 , pady=250 )

#my\_button5.place(x=880,y=900)

tk.mainloop()

exit()

root = tk.Tk()

root.title("PYTHON PROJECT")

root.geometry("1920x1080")

bg = tk.PhotoImage(file="/home/ritvik/Pictures/wp1913251.png")

my\_label = tk.Label(root, image=bg)

my\_label.place(x=0, y=0, relwidth=1, relheight=1)

my\_text = tk.Label(root, text="😎Optical Recognisers😎", font=("Jocker", 50), fg="blue",bg="BLACK")

my\_text.pack(pady=50)

my\_button2 = tk.Button(my\_label, text="Face Detector" ,command=facedetect , bg="blue", font=("Algerian" ,15) , padx=40 , pady=10 , borderwidth=7)

my\_button2.grid(row=0, column=1, padx=10 , pady=250)

my\_button2.place(x=580,y=170)

my\_button3 = tk.Button(my\_label, text="Human Detection for Monitoring" ,command=cctv\_capture ,bg="blue", font=("Algerian" ,15), padx=40 , pady=10 , borderwidth=7)

my\_button3.grid(row=0,column=5, padx=10 , pady=250)

my\_button3.place(x=50,y=170)

my\_button4 = tk.Button(my\_label, text="Vehicle Detect" ,command=vehicledetect ,bg="blue", font=("Algerian" ,15), padx=40 , pady=10 , borderwidth=7)

my\_button4.grid(row=0,column=5, padx=10 , pady=250)

my\_button4.place(x=1400,y=170)

my\_button4 = tk.Button(my\_label, text="Mask Detector" ,command=nothing ,bg="blue", font=("Algerian" ,15), padx=40 , pady=10 , borderwidth=7)

my\_button4.grid(row=0,column=5, padx=10 , pady=250)

my\_button4.place(x=1100,y=170)

my\_button5 = tk.Button(my\_label, text="Exit",command=root.quit, bg="blue",font=("Algerian" ,15), padx=40 , pady=10 , borderwidth=7)

my\_button5.grid(row=0, column=3, padx=10 , pady=250 )

my\_button5.place(x=880,y=900)

my\_button6 = tk.Button(my\_label, text="Show Trainer Code",command=showcode\_model, bg="blue",font=("Algerian" ,15), padx=40 , pady=10 , borderwidth=7)

my\_button6.grid(row=0, column=3, padx=10 , pady=250 )

my\_button6.place(x=300,y=500)

my\_button7 = tk.Button(my\_label, text="Show This Code",command=show\_this\_code, bg="blue",font=("Algerian" ,15), padx=40 , pady=10 , borderwidth=7)

my\_button7.grid(row=0, column=3, padx=10 , pady=250 )

my\_button7.place(x=1300,y=500)

my\_text2 = tk.Label(root, text='👽', font=("Jocker", 50), fg="blue")

my\_text2.place(x=928, y=500)

#my\_text1 = tk.Label(root, text='HELLO WORLD', font=("Jocker", 50), fg="blue")

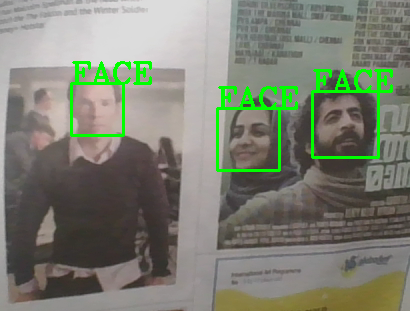
#my\_text1.place(x=600, y=600)

root.mainloop()

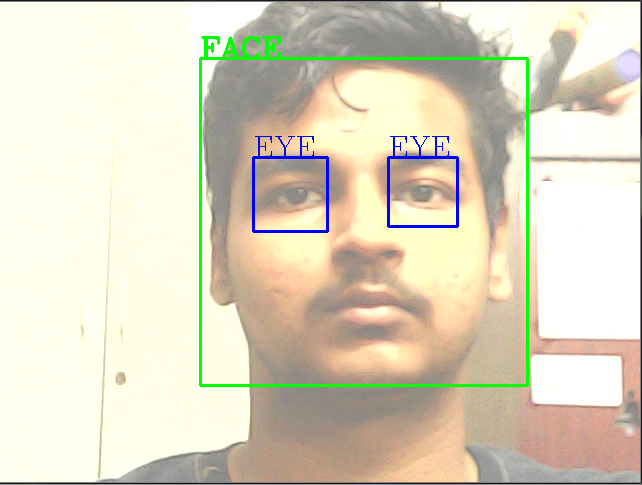
exit()

OUTPUT

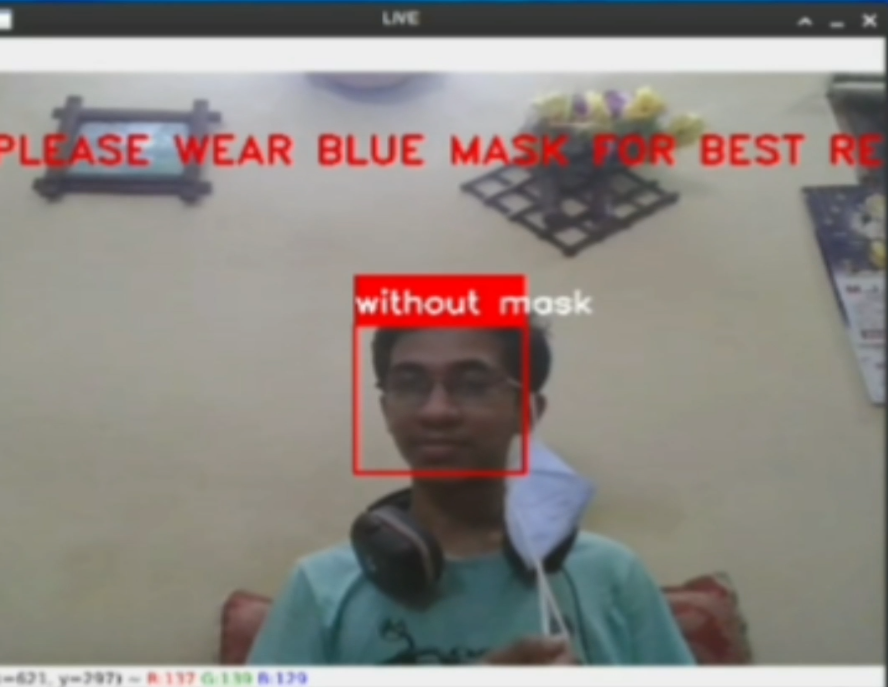
1. Face detector:







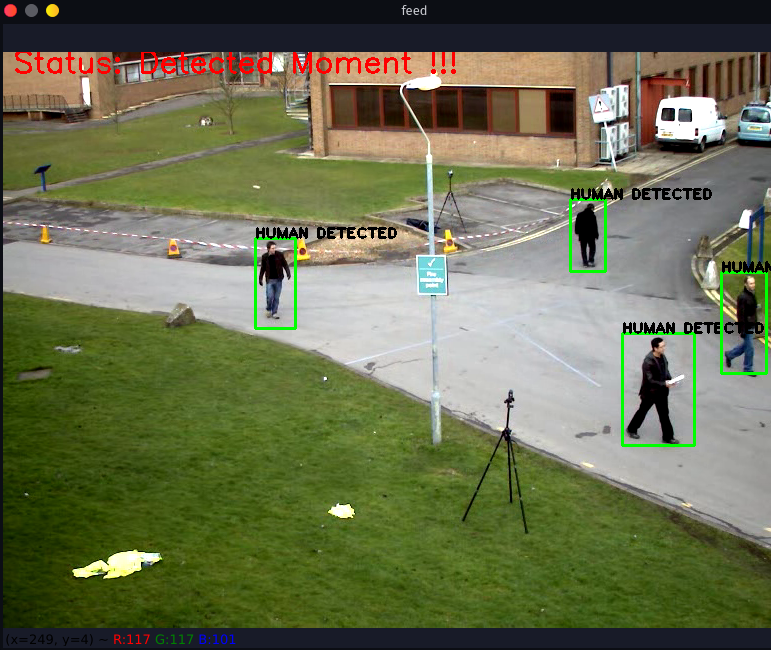
2. Mask detector:



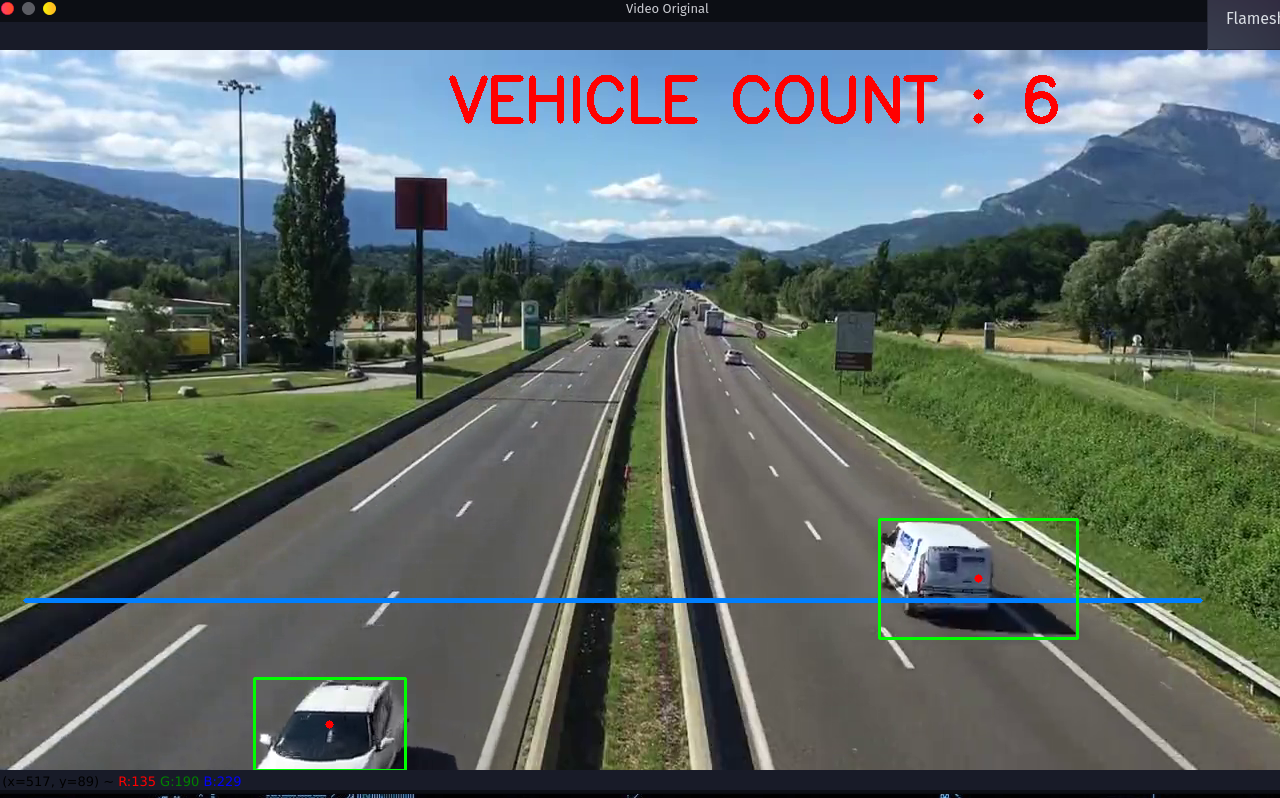
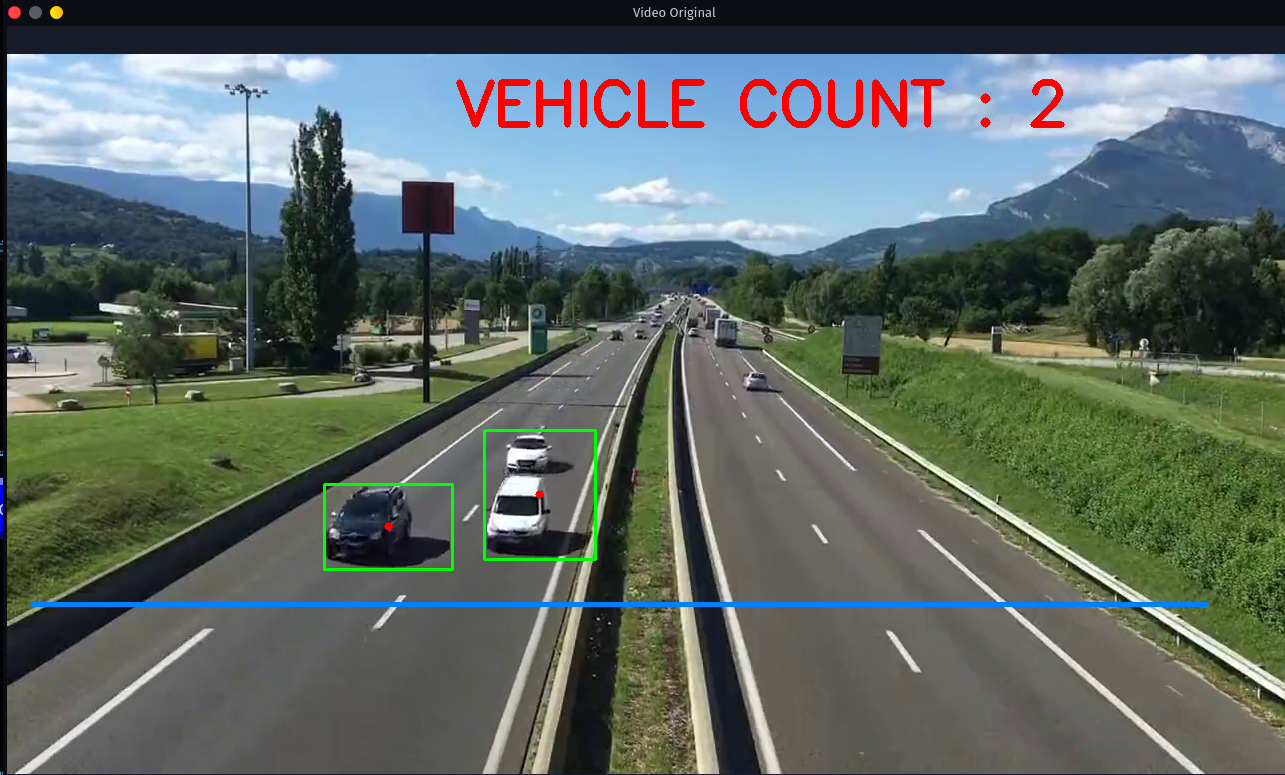


3. Motion detector:





4. Vehicle Motion detector:



ADVANTAGES OF PROJECT

**1. Face detector:**

i)The program Face detector can be used for many purposes such as for attendance in offices , schools etc.

ii)It can be used as a password which is nowadays a very famous and is available in every new model of smartphones and now even in laptops.

iii)Facebook uses face recognition technology to automatically recognize when Facebook members appear in photos. This makes it easier for people to find photos they are in and can suggest when particular people should be tagged in photos.

iv) Eye tracking has also been introduced to the human-computer interaction and gaming industry which now enables for instance game designers to get a better understanding of the game experience so that it is somewhat possible to control the experience and create features that push the boundaries of reality even more.In the time to come, it will most likely even be possible to personalize the game’s development in regard to [pupil dilation](https://imotions.com/blog/pupillometry-101/) of the player and the gamer will be able to control the game with eye movements.

2. Mask detector:

i)Mask detector can be used to recognize if an individual is wearing mask or not. In this time of new viruses and bacteria , wearing mask is a must.

ii) The Face Mask Detection System can be used at airports to detect travelers without masks. If a traveler is found to be without a face mask, their picture is sent to the airport authorities so that they could take quick action.

iii) Using Face Mask Detection System, Hospitals can monitor if their staff is wearing masks during their shift or not. If any health worker is found without a mask, they will receive a notification with a reminder to wear a mask.

iv) The Face Mask Detection System can be used at office premises to detect if employees are maintaining safety standards at work.

3. Motion detector:

i)Motion detector can be used as an alarm for illegal entry in any restricted area .Any irregular movement may be detected by the program and the alarm will warn the authorities about the illegal entry.

ii) Motion detector sensors also have a wide application in home and personal security. It is however, impossible to have a security system without a [motion sensor](https://www.eurovigil.in/home-automation-system/wireless-pir/).

iii) Motion sensors are also used to detect vehicles if an unauthorized vehicle enters your driveway.

4. Vehicle Motion detector:

i)Vehicle motion detector has a wide application in field of traffic and safety of passengers of vehicle. It can be used to detect lane changing , turning, and speed measurement.

ii)It can also be used for keeping a safe distance from another vehicles either in front or behind one’s own vehicle for safety and can reduce the chances of road accidents.

FURTHER DEVELOPMENT AREAS

**1. Face detector:**

1. Smarter advertising

Face recognition has the ability to make advertising more targeted by making educated guesses at people’s age and gender. Companies like Tesco are already planning on installing screens at gas stations with face recognition built in. It’s only a matter of time before face-recognition becomes an omni-present advertising technology.

## 2. Diagnose diseases

Face recognition can be used to diagnose diseases that cause detectable changes in appearance. As an example, the National Human Genome Institute Research Institute, uses face recognition to detect a rare disease called DiGeorge syndrome, in which there is a portion of the 22nd chromosome missing. Face recognition has helped diagnose the disease in 96% of cases. As algorithms get even more sophisticated, face recognition will become an invaluable diagnostic tool for all sorts of conditions.

## 3. Validate identity at ATMs

It seems likely that face scans will eventually replace ATM cards completely. But in the meantime, face recognition can be used to make sure that individuals using ATMs cards are who they say they are.  Face recognition is currently being used at ATMs in Macau to protect peoples’ identities.

**2. Mask detector:**

1.In future , we may see a mask detector which may identify a person’s face with mask on without any error .

In upcoming time , wearing mask may become a necessity for all of us to wear mask and the violator may have to face a severe punishment .So , mask detector may warn them in advance to wear mask

**3. Motion detector:**

In future, if implemented properly in highly prohibited areas, such as areas under military control, it can help to rule out unwanted invasions.

**4. Vehicle Motion detector:**

1. In future , vehicle motion detector can be used for collecting the information of the vehicles whose speed was above the limit or the information of those vehicles who violated the traffic rules by noting it’s number plate and the violator can be punished or fined easily.

2. Vehicle motion detector can also be used to calculate the density of cars travelling through a given area in a particular interval of time and can help for detecting the area which needs more maintenance.

3. It can also help in solving any accident case by giving the information regarding the speed , acceleration or direction of any vehicle before any accident.

BIBLIOGRAPHY

1. www.stackoverflow.com
2. www.geeksforgeeks.org
3. www.github.com
4. www.youtube.com

CD ATTACHED TO

RITVIK’S

PROJECT REPORT