

INTELLIGENT VIDEO MANAGEMENT



A PROJECT REPORT

Submitted By

| | |
|-----------------------------|-----------------------|
| R.NAVIN BALA | (912817104027) |
| N.D.NAVIN KUMAR | (912817104702) |
| R.SAKTHIVEL PRASANNA | (912817104034) |
| M.SELVA | (912817104036) |

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

**BACHELOR OF ENGINEERING IN COMPUTER
SCIENCE AND ENGINEERING**

**SYED AMMAL ENGINEERING COLLEGE
RAMANATHAPURAM**

ANNA UNIVERSITY::CHENNAI 600 025

MARCH 2021

ANNA UNIVERSITY::CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report “**INTELLIGENT VIDEO MANAGEMENT**” is the bonafide work of

R.NAVIN BALA (912817104027)

N.D.NAVIN KUMAR (912816104702)

R.SAKTHIVEL PRASANNA (912816104034)

M.SELVA (912816104036)

Who carried out the project work under my supervision.

SIGNATURE

Dr. B. MUTHU KUMAR, M.Tech., PhD.,

HEAD OF THE DEPARTMENT,

PROFESSOR,

Department of Computer Science and
Engineering

Syed Ammal Engineering College,

Ramanathapuram-623 502.

SIGNATURE

Mr.G.SARAVANA KUMAR, M.E.,

SUPERVISOR,

ASSISTANT PROFESSOR,

Department of Computer Science and
Engineering

Syed Ammal Engineering College,

Ramanathapuram-623 502.

Submitted to the viva voce held on.....

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We are very much grateful to the Almighty, without his blessings, we would not have achieved this today. We would like to thank our Honourable Secretary **Dr. CHINNADURAI ABDULLAH, M.D., D.M.R.D.**, who has provided the constant encouragement.

We take it as an immense pleasure to thank our adored Principal **Dr. M. PERIYSAMY, M.E., PhD.**, Syed Ammal Engineering College, for his constant support, encouragement and provided necessary facilities for our career growth.

We are very much eager to thank and honour our Head of the Department **Dr. B. MUTHU KUMAR, M. Tech., PhD.**, for his keen interest and continuous suggestions for completing this project as successful one.

Gratitude in the memory of heart, words cannot adequately express our thanks to our guide **Mr. G. SARAVANA KUMAR, M.E.**, for his valuable guidance, suggestions and encouragement to complete the project successfully.

We express our heartfelt thanks to our project supervisor and also our project coordinator **Mrs. G. SHARMILA, M.E., (PhD).**, who has ever been a great source of inspiration for our career.






We would like to extend our gratitude to all the Faculty Members who have helped directly and indirectly for the successful completion of our project.

Above all, we would like to thank our parents for their love and patience at all times.

R.NAVIN BALA
N.D.NAVIN KUMAR
R.SAKTHIVEL PRASANNA
M.SELVA

ABSTRACT

Intelligent Video Analytics system detects few events and object using camera. These algorithm modules coupled with Incident-Event-Action framework & Object detection helps in decreasing manual efforts as well as manual monitoring of video surveillance systems. In this project provides different analysis / analytics modules for various sectors. Some of them are:

-  Mask Detection
-  Face & Object Detection and Tracking
-  Watch If vehicles accident, if accident then inform to the hospital
-  Theft Detection, if theft detect the camera then inform to House owner
-  144 Watching, if group detect the camera then inform to Police

Each of the above Video Analytics / Analysis can run both in manual mode. On each Video Analytic Event and face detection, Intelligence Video management capture the data.

TABLE OF CONTENTS

| CHAPTER NO | TITLE | PAGE NO |
|---------------|-------------------------------|-------------|
| | ACKNOWLEDGEMENT | iii |
| | ABSTRACT | iv |
| | LIST OF FIGURES | vii |
| | LIST OF TABLES | viii |
| | LIST OF ABBREVIATIONS | ix |
| 1 | INTRODUCTION | 1 |
| | 1.1 VIDEO ANALYTICS | 1 |
| | 1.2 FACIAL RECOGNITION SYSTEM | 2 |
| | 1.3 OBJECT DETECTION | 2 |
| | 1.4 MOTION DETECTION | 3 |
| 2 | LITERATURE SURVEY | 4 |
| 3 | SYSTEM REQUIREMENTS | 9 |
| | 3.1 HARDWARE REQUIREMENTS | 9 |
| | 3.2 SOFTWARE REQUIREMENT | 9 |
| 4 | SYSTEM DESIGN | 10 |
| | 4.1 SYSTEM ARCHITECTURE | 10 |
| | 4.2 ER DIAGRAM | 10 |
| | 4.3 UML DIAGRAM | 11 |
| | 4.3.1 USE CASE DIAGRAM | 11 |
| | 4.3.2 CLASS DIAGRAM | 11 |
| | 4.3.3 ACTIVITY DIAGRAM | 12 |
| | 4.3.4 SEQUENCE DIAGRAM | 12 |
| 5 | SOFTWARE DESCRIPTION | 13 |
| | 5.1 OVERVIEW | 13 |
| | 5.2 WORKING OF PYTHON | 13 |
| | 5.3 OPENCV (CV2) | 15 |
| | 5.4 TENSORFLOW | 16 |
| | 5.5 NUMPY | 16 |

| | | |
|----------|--|-----------|
| | 5.6 IMUTILS | 18 |
| | 5.7 FACE RECOGNITION | 18 |
| 6 | MODULE DESCRIPTION | 19 |
| | 6.1 FACE MASK DETECTION | 19 |
| | 6.2 WORKING OF OPENCV | 20 |
| | 6.3 MOTION ANALYSIS | 21 |
| | 6.4 OBJECT DETECT AND TRACKING | 22 |
| 7 | SYSTEM STRATEGY | 23 |
| | 7.1 TESTING OVERVIEW | 23 |
| | 7.2 UNIT TESTING | 23 |
| | 7.3 INTEGRATION TESTING | 24 |
| | 7.4 VALIDATION TESTING | 25 |
| | 7.5 OUTPUT TESTING | 25 |
| 8 | CONCLUSION & FUTURE ENHANCEMENT | 27 |
| | 8.1 CONCLUSION | 27 |
| | 8.2 FUTURE ENHANCEMENT | 27 |
| | APPENDIX – 1 | 29 |
| | A.1 SOURCE CODE | 29 |
| | APPENDIX - 2 | 65 |
| | A.2 SCREEN SHOTS | 65 |
| | REFERENCES | 69 |

LIST OF TABLES

| TABLE NO | NAME OF THE TABLE | PAGE NO |
|-----------------|------------------------------------|----------------|
| 7.1 | Test Cases for Unit Testing | 18 |
| 7.2 | Test Cases for Integration Testing | 19 |
| 7.3 | Test Cases for Validation Testing | 19 |
| 7.4 | Test Cases for Output Testing | 20 |

LIST OF FIGURES

| FIGURE NO | NAME OF THE FIGURE | PAGE NO |
|------------------|---|----------------|
| 4.1 | System Architecture | 8 |
| 4.2 | ER Diagram | 8 |
| 4.3 | Use Case Diagram | 9 |
| 4.4 | Class Diagram | 9 |
| 4.5 | Activity Diagram | 10 |
| 4.6 | Sequence Diagram | 10 |
| 5.1 | Working of Python | 12 |
| 5.2 | Python IDLE Shell | 13 |
| 5.3 | OpenCV library | 14 |
| 5.4 | Numpy image to array conversion | 16 |
| 5.5 | Face Recognition | 16 |
| 6.1 | Face Mask Detection workflow | 18 |
| 6.2 | Working of opencv | 19 |
| 6.3 | Object Detect and tracking flow diagram | 20 |

LIST OF ABBREVIATIONS

| TERM | ABBREVIATIONS |
|-------------|---|
| Py | Python file Extension |
| Cv2 | ComputerVision2 (Python module) |
| ML | Machine Learning |
| AI | Artificial Intelligence |
| IDLE | Integrated Development and Learning Environment |

CHAPTER 1

INTRODUCTION

1.1 VIDEO ANALYTICS

Video analytics, also known as *video content analysis* or *intelligent video analytics*, has attracted **increasing interest** from both industry and the academic world. Thanks to the enormous advances made in deep learning, video analytics has introduced the **automation of tasks** that were once the exclusive purview of humans.

In this guide, you'll discover the basic **concept of video analytics**, how it's used **in the real world** to automate processes and gain valuable insights, and what you should consider when implementing an intelligent video analytics solutions in this project. The main goal of video analytics is to **automatically recognize temporal and spatial events in videos**. Some applications in the field of video analytics are widely known to the general public. One such example is **video surveillance**, a task that has existed for approximately 50 years. In principle, the idea is simple: install cameras strategically to allow human operators to control what happens in a room, area, or public space. Video analysis software can contribute in a major way by providing a means of accurately dealing with volumes of information. **Machine learning** and, in particular, the spectacular development of deep learning approaches, has **revolutionized video analytics**.

The use of Deep Neural Networks (DNNs) has made it possible to train video analysis systems that mimic human behaviour, resulting in a paradigm shift. It started with systems based on classic computer vision techniques (e.g. triggering an alert if the camera image gets too dark or changes drastically) and moved to **systems capable of identifying specific objects** in an image and tracking their path.

Crowd management is another key function of security systems. Cutting edge video analysis tools can make a big difference in places such as shopping malls, hospitals, stadiums, and airports. These tools can provide an estimated **crowd count** in real time and trigger alerts when a threshold is reached or surpassed. They can also analyse crowd flow to **detect movement in unwanted or prohibited directions**. Video analytics is a technology that processes a digital video signal using a special algorithm.

1.2 FACIAL RECOGNITION SYSTEM

A **facial recognition system** is a technology capable of matching a human face from a digital image or a video frame against a database of faces, typically employed to authenticate users through ID verification services, works by pinpointing and measuring facial features from a given image.

While initially a form of computer application, facial recognition systems have seen wider uses in recent times on smartphones and in other forms of technology, such as robotics. Because computerized facial recognition involves the measurement of a human's physiological characteristics facial recognition systems are categorised as biometrics. Although the accuracy of facial recognition systems as a biometric technology is lower than iris recognition and fingerprint recognition, it is widely adopted due to its contactless process.^[1] Facial recognition systems have been deployed in advanced human-computer interaction, video surveillance and automatic indexing of images.^[2] They are also used widely by law enforcement agencies.

1.3 OBJECT DETECTION

OpenCV is the huge open-source library for computer vision, machine learning, and image processing and now it plays a major role in real-time operation which is very important in today's systems. By using it, one can process images and videos to identify objects, faces, or even the handwriting of a human. This article focuses on detecting objects.

Note: For more information, refer to Introduction to OpenCV.

Object Detection

Object Detection is a computer technology related to computer vision, image processing, and deep learning that deals with detecting instances of objects in images and videos. We will do object detection in this article using something known as **haar cascades**.

Haar Cascades

Haar Cascade classifiers are an effective way for object detection. This method was proposed by Paul Viola and Michael Jones in their paper Rapid Object Detection using a Boosted

Cascade of Simple Features. Haar Cascade is a machine learning-based approach where a lot of positive and negative images are used to train the classifier.

- **Positive images** – These images contain the images which we want our classifier to identify.
- **Negative Images** – Images of everything else, which do not contain the object we want to detect.

Requirements.

1.4 MOTION DETECTION

I have always wanted a software based on the webcam that can detect movement and record in a video file only something is moving. It is now done. Indeed it is not really conceivable to record all along because the hard disk drive would be quickly filled if the software has to run a day for instance. Because I love OpenCV and due to lack of this kind of software on Linux I have decided to do it. As said before the program analyse the images taken from the webcam and intent to detect movement. If a movement is detected the program start recording the webcam in a video file 10 seconds. After that if a movement is again detected it still record until movements stops.

CHAPTER 2

LITERATURE SURVEY

1. Title: Face-Mask Detector

Author: Karan-Malik

Year: 2020

Abstract:

Face Mask Detection system built with OpenCV, Keras/TensorFlow using Deep Learning and Computer Vision concepts in order to detect face masks in static images as well as in real-time video streams. This project uses a Deep Neural Network, more specifically a Convolutional Neural Network, to differentiate between images of people with and without masks. The CNN manages to get an accuracy of **98.2% on the training set** and **97.3% on the test set**. Then the stored weights of this CNN are used to classify as mask or no mask, in real time, using OpenCV. With the webcam capturing the video, the frames are pre-processed and fed to the model to accomplish this task. The model works efficiently with no apparent lag time between wearing/removing mask and display of prediction.

Advantage:

- The camera with AI-based face mask detection monitoring can generate real-time alerts.
- Face mask detection feature uses visible stream from the camera combined with AI techniques to detect and generate an alert for people not wearing face masks.
- A user-friendly interface allows monitoring and review of alerts generated by the system.

Disadvantage:

- Start by the User.
- Need high configuration system.

2. Title: Detection-and-Tracking

Author: Ambakick

Year: 2020

Abstract:

The method Proposed here is divided into 2 main parts of Person Detection and Person Tracking. Person Detection - The person detection in Real-time is done with the help of Single Shot MultiBox Detector. SSD achieves 75.1% mAP, outperforming a comparable state of the art Faster R-CNN model. and the SSD model is available in the Tensorflow detection zoo. The seamless integration of SSD with tensorflow helps in further optimization and implementation of the algorithm. The SSD object detection composes of 2 parts: Extract feature maps, and Apply convolution filters to detect objects. Even though SSD is capable of detecting multiple objects in the frame, in this project I limited its detection to just human.

Person Tracking - Bounding box can be achieved around the object/person by running the Object Detection model in every frame, but this is computationally expensive. The tracking algorithm used here is Kalman Filtering . The Kalman Filter has long been regarded as the optimal solution to many tracking and data prediction tasks. Its use in the analysis of visual motion. The purpose of Filtering is to extract the required information from a signal, ignoring everything else. In this project the Kalman Filter is fed with the velocity, position and direction of the person which helps it to predict the future location of the Person based on his previous data.

Advantage:

- Easy to find specific object or person in video file or streaming live.

Disadvantage:

- Need more times for checking given input to find object in loaded video or live streaming.

3. Title: Detection-and-Tracking

Author: Manojpawars

Year: 2020

Abstract:

Dense Convolutional Network (DenseNet), connects each layer to every other layer in a feed-forward fashion. Whereas traditional convolutional networks with L layers have L connections - one between each layer and its subsequent layer - our network has $L(L+1)/2$ direct connections. For each layer, the feature-maps of all preceding layers are used as inputs, and its own feature-maps are used as inputs into all subsequent layers. DenseNets have several compelling advantages: they alleviate the vanishing-gradient problem, strengthen feature propagation, encourage feature reuse, and substantially reduce the number of parameters.

The 1-crop error rates on the imagenet dataset with the pretrained model are listed below.

Model structure Top-1 error Top-5 error

densenet121 : 25.35 : 7.83

densenet169 : 24.00 : 7.00

densenet201 : 22.80 : 6.43

densenet161 : 22.35 : 6.20

Advantage:

- Easy to find specific object or person in video file or streaming live.

Disadvantage:

- Need more times for checking given input to find object in loaded video or live streaming.

4. Title: Detection-and-Tracking

Author: jatinarora

Year: 2021

Abstract:

Theft is the most common crime committed across the world. According to the National Crime Records Bureau (NCRB), ~80% of the criminal cases are related to theft as shown in figure. Increasing theft rates cause people to suffer both financially and emotionally. Therefore, there is a need to develop a more deterrent surveillance system, which is convenient to use, free from false alarms, minimize human interference, and cost-effective. Machine Learning (ML) techniques prove to be fruitful in developing efficient surveillance systems. This paper aims to design a theft detection and monitoring system, which would be capable to detect theft using a motion-sensing camera using ML and alarm the owner with an alert message along with the captured image of that instance of motion.

The system consists of several levels of surveillance at each level the activity in each frame of the video will be monitored thoroughly using ML models, which are solely trained to perform their specific job. There are a total of six levels of surveillance and the system consists of two modes (i.e. day and night) and it will totally depend on the user which mode is required at the moment.

Advantage:

- Easy to find specific object or person in video file or streaming live.

Disadvantage:

- Need more times for checking given input to find object in loaded video or live streaming.

5. Title: Crowd-Detection-And-Analyzation

Author: jatinarora

Year: 2021

Abstract:

A Python Program to detect the maximum numbers of people present in the frame and show visualised analyse of the data. CrowdDetection.py First the program uses openCV to detect the average number of people present in the frame for a minute. It uses a specially trained algorithm known as the cascade of a program and this cascade uses different Objects. caffemodel Objects.prototxt.txt training model to learn and try to detect many objects and people. Since we just need our program to detect only the number of people in the given frame we enable only the people model to detect people. With the help of the ids of each detection, the program counts the number of people in the frame for one minute. This data is converted to an average of one minute. The time library is used to keep a track of time. Both the average and current time data is used for processing the further project.

Advantage:

- 3 or more than people in one place then, alert to the user.

Disadvantage:

- Need more CUDA graphics hardware in system for good performance.

CHAPTER 3

SOFTWARE REQUIREMENT CONFIGURATION

3.1 HARDWARE REQUIREMENTS

- System : Intel i5 10th Generation.
- Hard Disk : 256 GB
- Ram : 8 GB
- Graphics Card : 4GB HD Graphics Card
- HD Webcam.

3.2 SOFTWARE REQUIREMENTS

- OS : Windows 10
- Language : Python .

CHAPTER 4

SYSTEM DESIGN

4.1 SYSTEM ARCHITECTURE

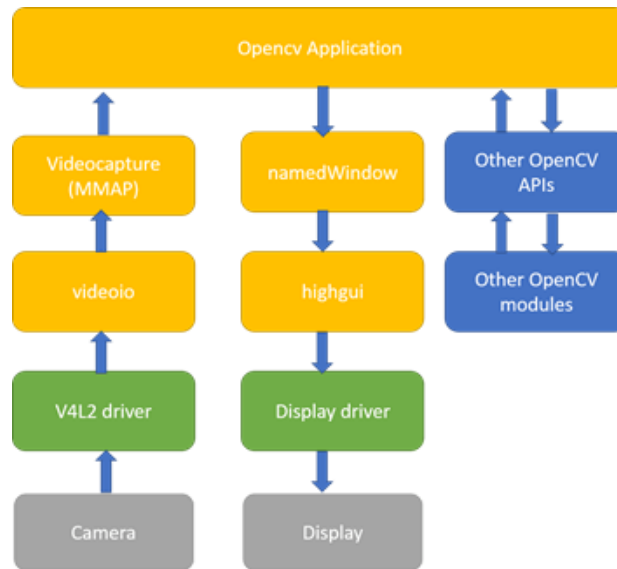


Fig 4.1 System Architecture

4.2 ER DIAGRAM

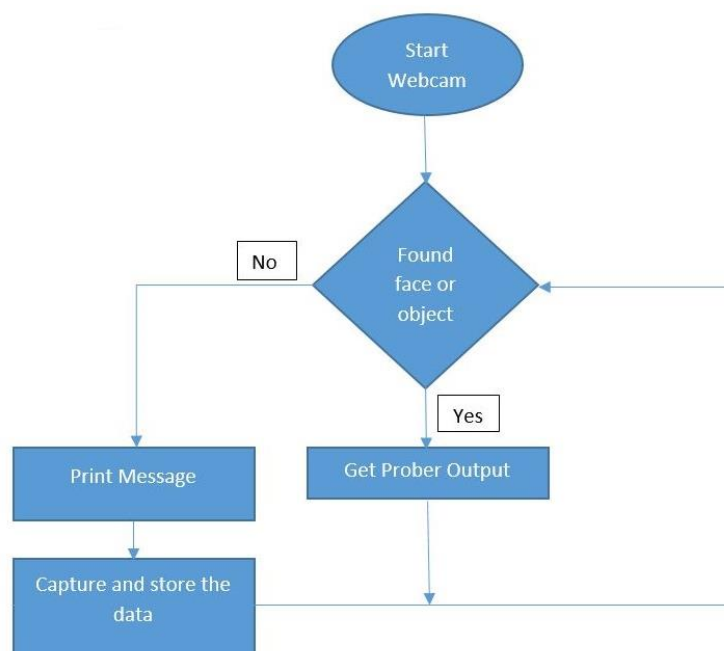


Fig 4.2 ER Diagram

UML DIAGRAM

4.3 USE CASE DIAGRAM

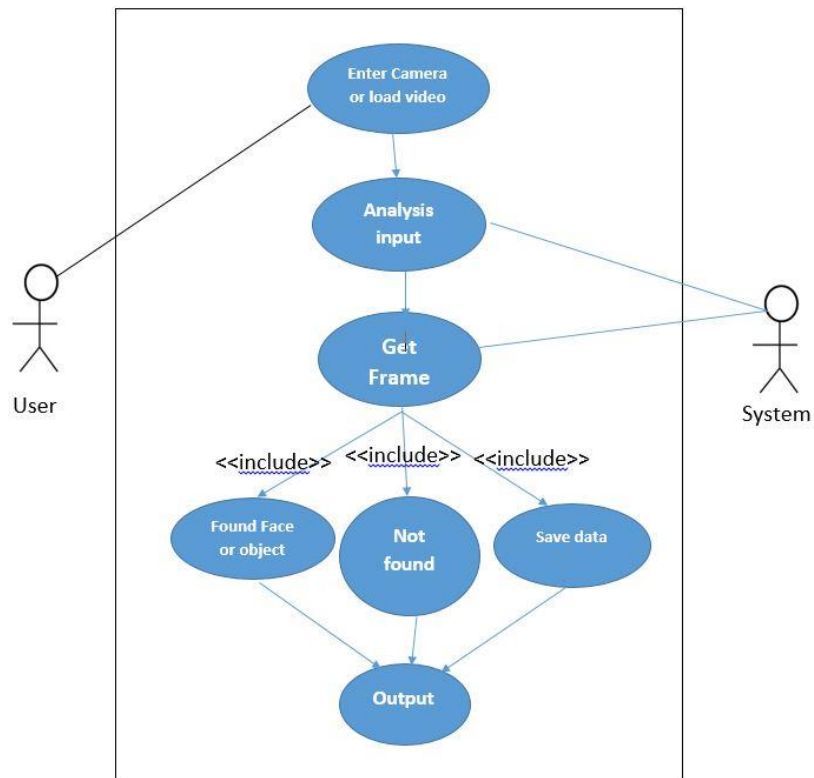


Fig 4.3 Use Case Diagram

4.4 CLASS DIAGRAM

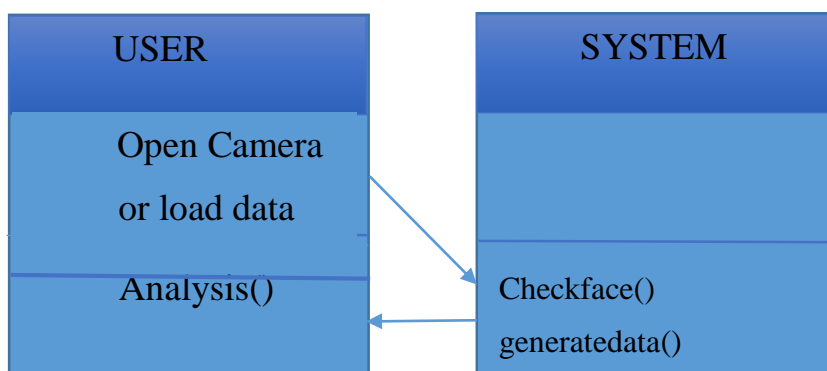


Fig 4.4 Class Diagram

4.5 ACTIVITY DIAGRAM

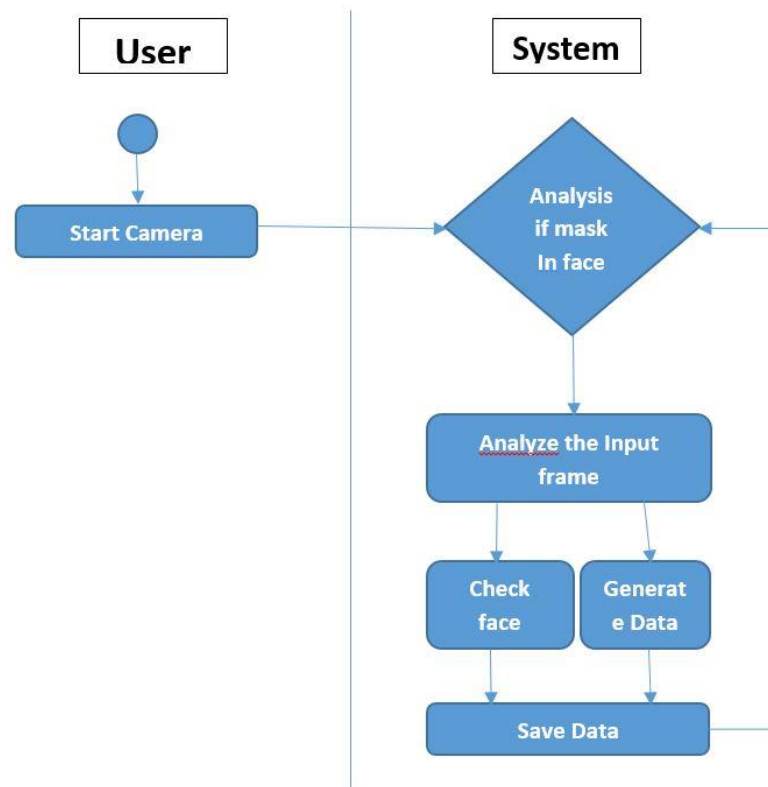


Fig 4.5 Activity Diagram

4.5 SEQUENCE DIAGRAM

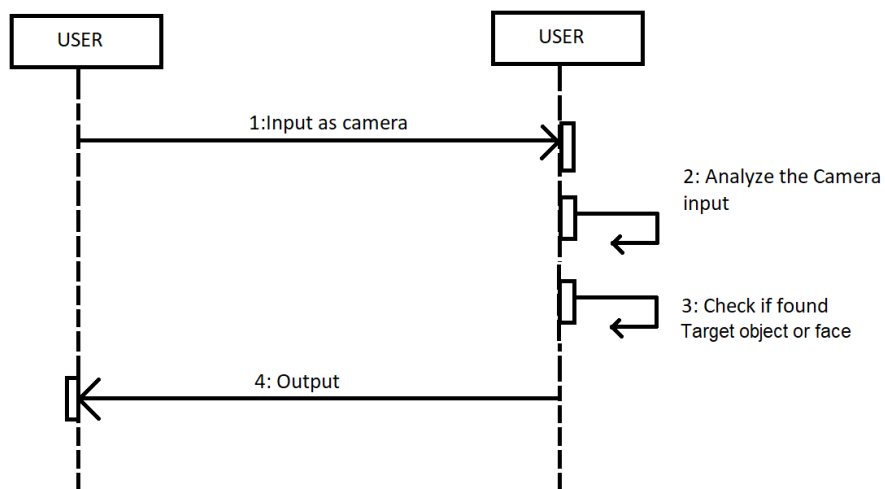


Fig 4.6 Sequence Diagram

CHAPTER 5

SOFTWARE DESCRIPTION

5.1 OVERVIEW

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation and it has fewer syntactical constructions than other languages. Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. You can actually sit at a Python prompt and interact with the interpreter directly to write your programs. It supports Object-Oriented style or technique of programming that encapsulates code within objects. Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games. Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows and Macintosh. It supports functional and structured programming methods as well as OOP. It can be used as a scripting language or can be compiled to byte-code for building large applications. It provides very high-level dynamic data types and supports dynamic type checking. It supports automatic garbage collection. It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

5.2 WORKING OF PYTHON

Python is an object oriented programming language like Java. Python doesn't convert its code into machine code, something that hardware can understand. It actually converts the code into something called byte code. So within python, compilation happens, but it's just not into a machine language. It is into byte code and this byte code can't be understood by CPU. So we need actually an interpreter called the Python Virtual Machine (PVM). The python virtual machine executes the byte codes. Working diagram of Python shown in Fig.5.1.

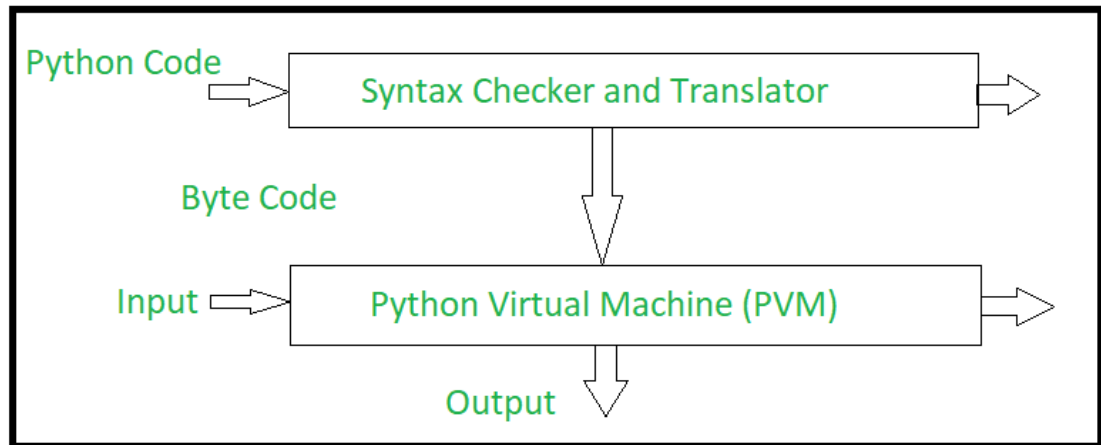


Fig.5.1 Working of Python

The Python interpreter performs following tasks to execute a Python program:

- Step 1: The interpreter reads a python code or instruction. Then it verifies that the instruction is well formatted, i.e. it checks the syntax of each line. If it encounters any error, it immediately halts the translation and shows an error message.
- Step 2: If there is no error, i.e. if the python instruction or code is well formatted then the interpreter translates it into its equivalent form in intermediate language called “Byte code”. Thus, after successful execution of Python script or code, it is completely translated into Byte code.
- Step 3: Byte code is sent to the Python Virtual Machine (PVM). Here again the byte code is executed on PVM. If an error occurs during this execution, then the execution is halted with an error message.
- Step 4: After successful execution of Python script or code, the PVM produce an output.

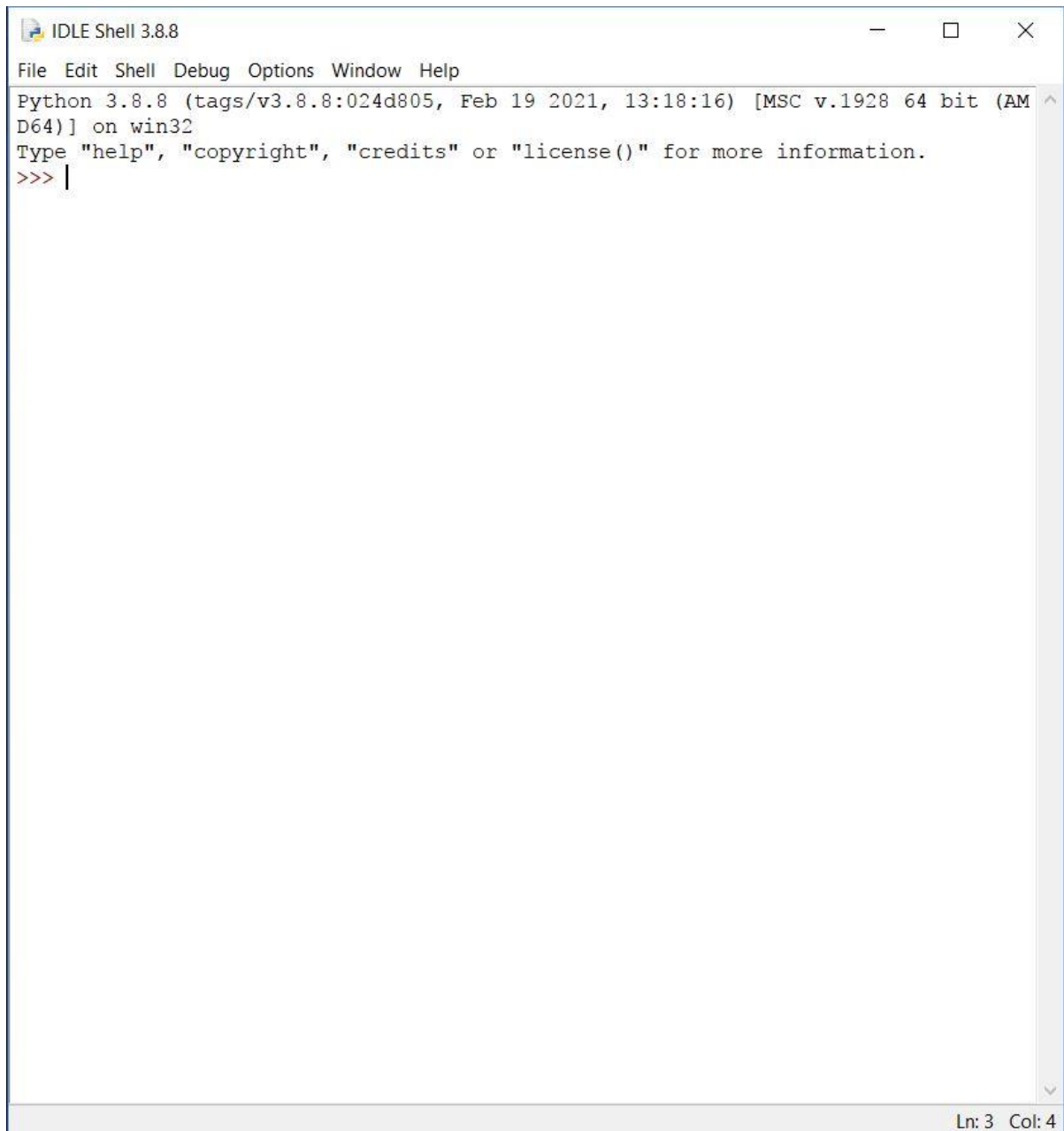


Fig 5.2 Python IDLE Shell

5.3 OPENCV (CV2)

The Open Source Computer Vision Library (OpenCV) is the most well-known computer vision library. It contains a comprehensive set of machine learning algorithms to perform common tasks such as image classification, face recognition, and object detection and tracking. It is widely used by companies and research groups, as it can be used via its native C++ interface, or through Java and Python wrappers.

Since it is a general computer vision library, it is possible to implement a video analysis system with OpenCV. However, as it is not a specialized video analytics library, it may be more interesting to turn to other available libraries.

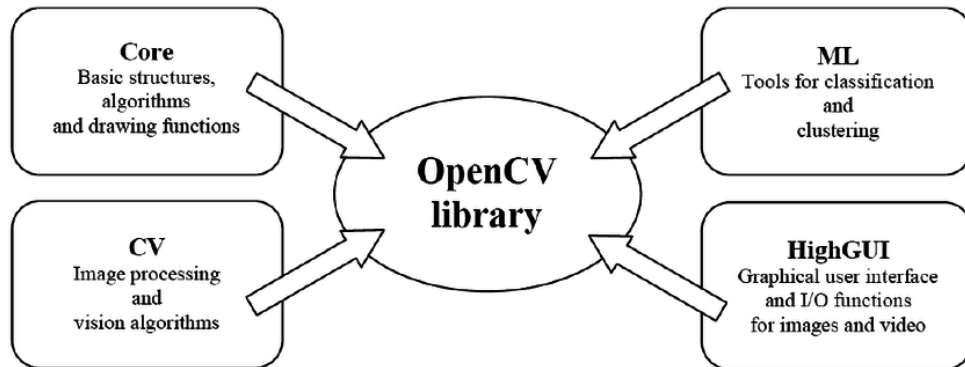


Fig 5.3 OpenCV library

5.4 TENSORFLOW

TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications. Build and train ML models easily using intuitive high-level APIs like Keras with eager execution, which makes for immediate model iteration and easy debugging.

5.5 NUMPY

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

At the core of the NumPy package, is the *ndarray* object. This encapsulates *n*-dimensional arrays of homogeneous data types, with many operations being performed in compiled code for performance. There are several important differences between NumPy arrays and the standard Python sequences:

- NumPy arrays have a fixed size at creation, unlike Python lists (which can grow dynamically). Changing the size of an *ndarray* will create a new array and delete the original.
- The elements in a NumPy array are all required to be of the same data type, and thus will be the same size in memory. The exception: one can have arrays of (Python, including NumPy) objects, thereby allowing for arrays of different sized elements.
- NumPy arrays facilitate advanced mathematical and other types of operations on large numbers of data. Typically, such operations are executed more efficiently and with less code than is possible using Python's built-in sequences.
- A growing plethora of scientific and mathematical Python-based packages are using NumPy arrays; though these typically support Python-sequence input, they convert such input to NumPy arrays prior to processing, and they often output NumPy arrays. In other words, in order to efficiently use much (perhaps even most) of today's scientific/mathematical Python-based software, just knowing how to use Python's built-in sequence types is insufficient - one also needs to know how to use NumPy arrays.

The points about sequence size and speed are particularly important in scientific computing. As a simple example, consider the case of multiplying each element in a 1-D sequence with the corresponding element in another sequence of the same length. If the data are stored in two Python lists, *a* and *b*, we could iterate over each element:

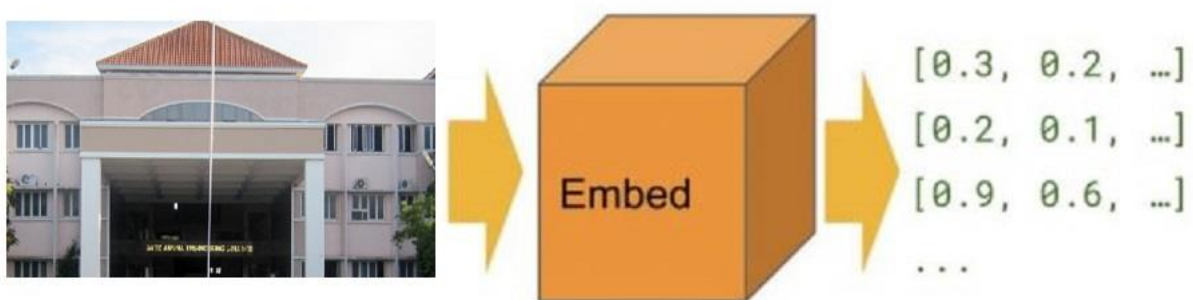


Fig 5.4 Numpy image to array conversion

5.6 IMUTILS

A series of convenience functions to make basic image processing functions such as translation, rotation, resizing, skeletonization, displaying Matplotlib images, sorting contours, detecting edges, and much more easier with OpenCV and both Python.

5.7 FACE RECOGNITION

Recognize and manipulate faces from Python or from the command line with the world's simplest face recognition library. Built using dlib's state-of-the-art face recognition built with deep learning. The model has an accuracy of 99.38% on the Labeled Faces in the Wild benchmark. This also provides a simple face_recognition command line tool that lets you do face recognition on a folder of images from the command line.

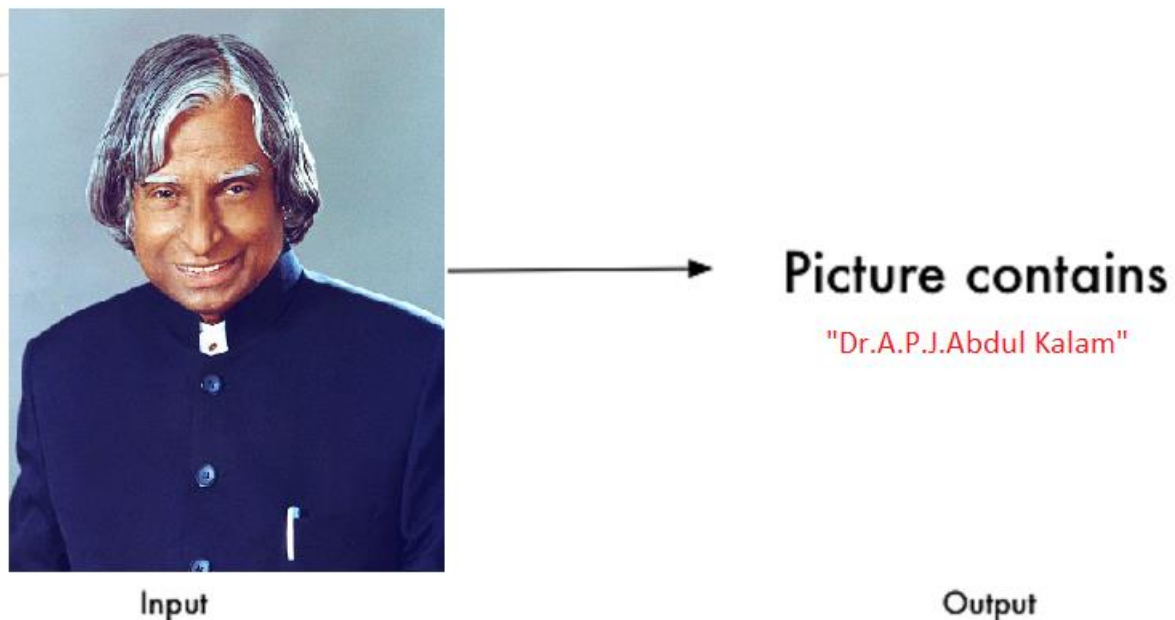


Fig 5.5 Face Recognition

CHAPTER 6

6.1 FACE MASK DETECTION

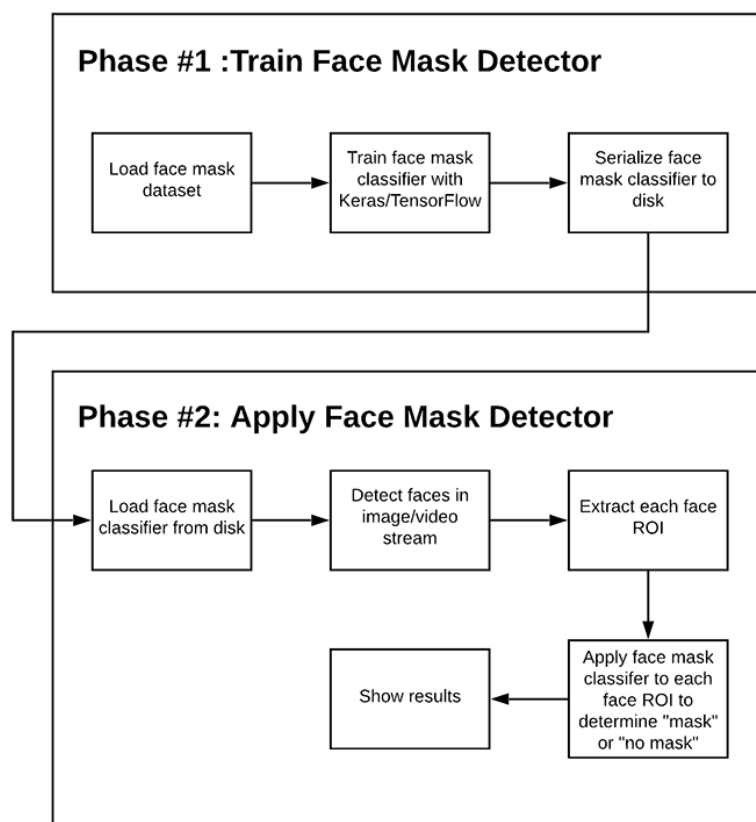


Fig 6.1 Face Mask Detection workflow

The Face Mask Detection module is designed to detect people not wearing medical face masks in the frame. When such persons are detected, the module highlights them in the frame with a square in real time and enters the event in the event log. The module is capable of detecting up to ten people not wearing face masks in the frame at the same time (if allowed by the computing capacity). The module does not recognize (identify) faces, it cannot tell one person from the other or compare a person's face with the faces from a database; it only finds people without face masks in the frame. When an infringement event (no mask) is detected, the module briefly highlights the person's face with a red square in the client application and creates a corresponding event in the event log. The repeated detection of the infringement by the same person will become possible only after the disappearance of this person from the frame for three seconds minimum (e.g. when the person leaves the frame or covers his/her face completely).

6.2 WORKING OF OPENCV

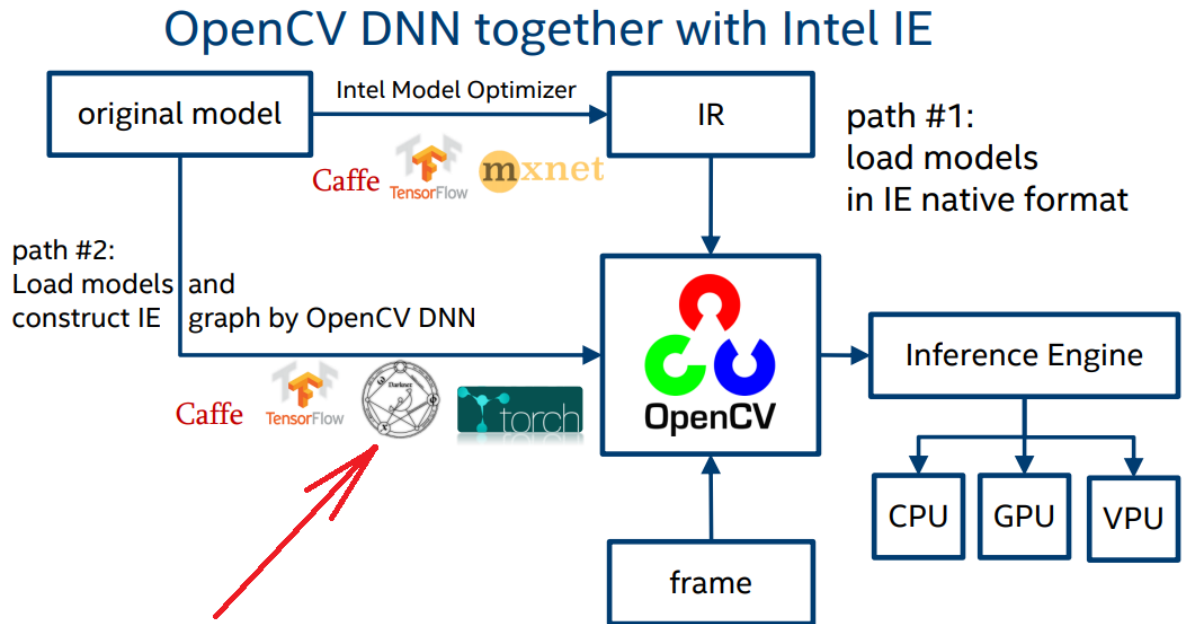


Fig 6.2 Working of opencv

That's it, now, we know how to process videos in python, and we know how to access each frame of a video file. From here, we can add any kind of image processing inside this cycle, it may be either object detection, or human pose estimation model, or both of them. The further dive into OpenCV for video processing is up to the reader. There are a lot of useful tools that will meet any of your requirement, from the easiest task (resize, crop, color/brightness adjustment), passing by image filtering to Delaunay Triangulation.

OpenCV Video Analysis modules

- Motion analysis
- Object Tracking
- C API

6.3 MOTION ANALYSIS

Motion analysis is used in computer vision, image processing, high-speed photography and machine vision that studies methods and applications in which two or more consecutive images from an image sequences, e.g., produced by a video camera or high-speed camera, are processed to produce information based on the apparent motion in the images. In some applications, the camera is fixed relative to the scene and objects are moving around in the scene, in some applications the scene is more or less fixed and the camera is moving, and in some cases both the camera and the scene are moving.

The motion analysis processing can in the simplest case be to detect motion, i.e., find the points in the image where something is moving. More complex types of processing can be to track a specific object in the image over time, to group points that belong to the same rigid object that is moving in the scene, or to determine the magnitude and direction of the motion of every point in the image. The information that is produced is often related to a specific image in the sequence, corresponding to a specific time-point, but then depends also on the neighbouring images. This means that motion analysis can produce time-dependent information about motion.

Applications of motion analysis can be found in rather diverse areas, such as surveillance, medicine, film industry, automotive crash safety,^[1] ballistic firearm studies,^[2] biological science,^[3] flame propagation,^[4] and navigation of autonomous vehicles to name a few examples.

6.4 OBJECT DETECT AND TRACKING

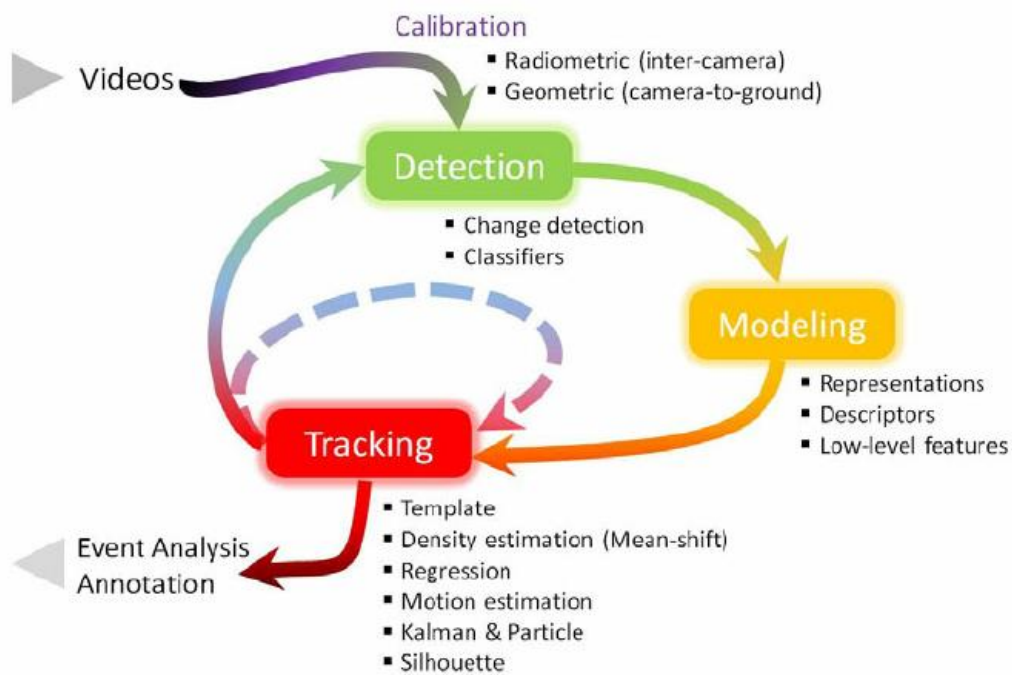


Fig 6.3 Object Detect and tracking flow diagram

Detecting and tracking objects are among the most prevalent and challenging tasks that a surveillance system has to accomplish in order to determine meaningful events and suspicious activities, and automatically annotate and retrieve video content. Under the business intelligence notion, an object can be a face, a head, a human, a queue of people, a crowd as well as a product on an assembly line.

In this chapter we introduce the reader to main trends and provide taxonomy of popular methods to give an insight to underlying ideas as well as to show their limitations in the hopes of facilitating integration of object detection and tracking for more effective business oriented video analytics.

CHAPTER 7

SYSTEM STRATEGY

7.1 TESTING OVERVIEW

System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently before the live operation commences. Testing is the process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an error. A successful test is one that answers a yet undiscovered error.

Testing is vital to the success of the system. System testing makes a logical assumption that if all parts of the system are correct, the goal will be successfully achieved. The candidate system is subject to a variety of tests-on-line responses, Volume Street, recovery and security, and usability tests. A series of tests are performed before the system is ready for user acceptance testing. Any engineered product can be tested in one of the following ways. Knowing the specified function that a product has been designed to form, a test can be conducted to demonstrate each function is fully operational. Knowing the internal working of a product, tests can be conducted to ensure that “all gears mesh”, that is the internal operation of the product performs according to the specification and all internal components have been adequately exercised.

7.2 UNIT TESTING

Unit testing is the testing of each module and the integration of the overall system is done. Unit testing becomes verification efforts on the smallest unit of software design in the module. This is also known as ‘module testing’. The modules of the system are tested separately. This testing is carried out during the programming itself. In this testing step, each model is found to be working satisfactorily as regard to the expected output from the module. For example, the validation check is done for verifying the data given by the user where both format and validity of the data entered is included. It is very easy to find an error and debug the system. We have applied Unit testing for our project and the test cases are in Table 7.1.

Table 7.1 Test Cases for Unit Testing

| Case Id | Test Case | Purpose | Expected Result | Error Message |
|----------------|-------------------------------|--|---------------------------|---------------------------|
| 1 | Open Camera | Analyse the input & Perform the Operation | Finally get expect result | There is no error message |
| 2 | Check camera working properly | Process video frame one by one & Apply operation each frame in live or loaded video stream | Successfully | There is no error message |

7.3 INTEGRATION TESTING

Data can be lost across an interface, one module can harm the other sub function, when combined, may not produce the desired major function. Integrated testing is systematic testing that can be done with sample data. The need for the integrated test is to find the overall system performance. There are two types of integration testing. They are:

- I. Top-down integration testing.
- II. Bottom-up integration testing. We have applied Top-down Integration testing for our project and the test cases are in Table 7.2.

Table 7.2 Test Cases for Integration Testing

| Case Id | Test Case | Purpose | Expected Result | Error Message |
|---------|-----------|---|--------------------------------|---------------------------|
| 1 | Data loss | To avoid data losing during video analysing | Data is successfully analysed. | There is no error message |

7.4 VALIDATION TESTING

After the culmination of black-box testing, the software is completed assembly as a package, interfacing errors have been uncovered and corrected and the final series of software validation tests begin validation testing can be defined as many, but a single definition is that validation succeeds when the software functions in a manner that can be reasonably expected by the user. We have applied Validation testing for our project and the test cases are in Table 7.3.

Table 7.3 Test Cases for Validation Testing

| Case Id | Test Case | Purpose | Test Data | Expected Result | Error Message |
|---------|---------------------|------------------------------|-----------------|-----------------|---------------|
| 1 | New partition entry | To check exit patient or not | Data from local | | |

7.5 OUTPUT TESTING

After performing the validation testing, the next step is output asking the user about the format required testing of the proposed system, since no system could be useful if it does not produce the required output in the specific format. The output is displayed or generated by the system under consideration. Here the output format is considered in two ways. One is a screen and the other is printed format. The output format on the screen is found to be correct as the format was designed in the system phase according to the user needs. For the hard copy also output comes out as the specified requirements by the user. Hence the output testing does not result in

any connection in the system. We have applied Output testing for our project and the test cases are in Table 7.4.

Table 7.4 Test Cases for Output Testing

| Case Id | Test Case | Purpose | Test Data | Expected Result |
|----------------|-------------------------|---------------------|-------------------------|------------------------|
| 1 | Check the output result | Collect information | Data from output folder | View Data Easily |

CHAPTER 8

CONCLUSION AND FUTURE ENHANCEMENT

8.1 CONCLUSION

In the field of Artificial Intelligence, Computer Vision is one of the most interesting and Challenging tasks. Computer Vision acts like a bridge between Computer Software and visualizations around us. It allows computer software to understand and learn about the visualizations in the surroundings. For Example: Based on the color, shape and size determining the fruit. This task can be very easy for the human brain however in the Computer Vision pipeline, first we gather the data, then we perform the data processing activities and then we train and teach the model to understand how to distinguish between the fruits based on size, shape and color of fruit.

8.2 FUTURE ENHANCEMENT

The library is open-source which means that the source code is publicly available. We can customize the code to meet the specific business requirements. We can even write more code to add extra functionality. It is available for free to use in commercial products. The OpenCV Python library makes use of numpy, which is a highly optimized library to perform numerical operations on multidimensional arrays. So all the OpenCV array-like structures are converted into numpy arrays and this makes it easier to integrate with other libraries that use numpy like SciPy and Matplotlib. OpenCV contains implementations of more than 2500 algorithms. It also has interfaces for multiple languages like Python, java, etc that makes it easy to integrate. Python has a rich set of *libraries* that provides a powerful environment for scientific calculations. The libraries like SciPy, numpy, scikit-learn, matplotlib made complex tasks easier to do in Python. Building prototypes application is fast in Python. We can integrate our application with web frameworks like *Django* which is a high-level web framework that encourages rapid development. Computer vision is a field of study where we get computers to understand digital images or a group of images. It's very easy for humans to recognize an object by looking at them but in computers, it is not an easy task.

Let us first understand how the computer interprets an image. To represent an image in a computer, we use a grid structure in which each pixel in the grid contains the color

of that particular pixel. Each color is represented by 3 values (c_1, c_2, c_3) which are basically red, green and blue and their values range from 0-255 depending on the intensity of each color. **For example** – the red color can be represented as (255,0,0), green color is represented as (0,255,0). Using different values for different colors, we can create any color we want. So for an image of size 200×200 px we will have 40,000 pixels and each pixel has 3 values. $40,000 * 3 = 1,20,000$ numbers to represent an image of size 200×200 px. Now imagine, we have a group of photos of different people and we have to classify which photo is of which person. The computer vision algorithms do this task by operating on the data on each image and look for patterns that help in classifying problems. The visual system of humans has evolved over thousands of years. The images that are projected on our retina are converted to neuron signals. Computer vision understands how human vision works and then we apply this knowledge on computers. It becomes challenging to build robust applications because the minute changes in any of the parameters of the image can change the resulting outcomes.

The way our machine perceives an image will differ according to lighting conditions, rotation of the image or even the perspective of the image. This is why it is a complex task and needs careful attention to the small details while implementing computer vision solutions.

APPENDIX-1

A.1 SOURCE CODE

main_window.py

```
import sys

import os

import tkinter as tk

import tkinter.font as tkFont

from tensorflow.keras.applications.mobilenet_v2 import preprocess_input

from tensorflow.keras.preprocessing.image import img_to_array

from tensorflow.keras.models import load_model

from imutils.video import VideoStream

import numpy as np

import imutils

import time

import cv2

import os

import face_recognition as fr

main_menu=tk.Tk()

main_menu.geometry("1920x1080")
```

```
main_menu.title("INTELLIGENCE VIDEO MANAGEMENT")
```

```
main_menu.configure(bg='black')
```

```
def mask_detect():
```

```
    os.system('Face_Mask_Detection.py')
```

```
def face_or_obj_tracking():
```

```
    os.system('Face_or_object_tracking.py')
```

```
def theft_detection():
```

```
    os.system('Theft_Detection.py')
```

```
def watch_accident():
```

```
    os.system('watch_accident.py')
```

```
def watch_144():
```

```
    os.system('watch_144.py')
```

```
fontStyle = tkFont.Font(family="Exotic-Bold", size=20)
```



```
label = tk.Label(main_menu, text="INTELLIGENCE VIDEO MANAGEMENT")
```

```
Mask_detection_btn=tk.Button(main_menu,text="Mask  
Detection",command=mask_detect,activeforeground = "green",height = 20, width =  
20,bg="blue",fg="white")
```

```
Face_or_obj_tracking_btn=tk.Button(main_menu,text="Face or Object  
Tracking",command=face_or_obj_tracking,activeforeground = "green",height = 20, width =  
20,bg="red",fg="white")
```

```
theft_detection_btn=tk.Button(main_menu,text="Theft  
Detection",command=theft_detection,activeforeground = "green",height = 20, width =  
20,bg="yellow",fg="white")
```

```
watch_accident_btn=tk.Button(main_menu,text="Watch  
Accident",command=watch_accident,activeforeground = "green",height = 20, width =  
20,bg="blue",fg="white")
```

```
watch_144_btn=tk.Button(main_menu,text="Watch  
144",command=watch_144,activeforeground = "green",height = 20, width =  
20,bg="green",fg="white")
```

```
names = tk.Label(main_menu, text="Develop by",bg="black",fg="white")
```

```
nd = tk.Label(main_menu, text="N.D.NAVIN KUMAR",bg="black",fg="white")
```

```
navin = tk.Label(main_menu, text="R.NAVIN BALA",bg="black",fg="white")
```

```
sakthi = tk.Label(main_menu, text="R.SAKTHIVEL PRASANNA",bg="black",fg="white")
```

```
selva = tk.Label(main_menu, text="M.SELVA",bg="black",fg="white")
```

```
label.config(font=("Carbon Block", 30))
```

```
names.config(font=("Carbon Block", 20))
```

```
nd.config(font=("Carbon Block", 20))
```

```
navin.config(font=("Carbon Block", 20))
```

```
sakthi.config(font=("Carbon Block", 20))
```

```
selva.config(font=("Carbon Block", 20))
```

```
Mask_detection_btn.config(font=("Carbon Block", 20))
```

```
Face_or_obj_tracking_btn.config(font=("Carbon Block", 20))
```

```
theft_detection_btn.config(font=("Carbon Block", 20))
```

```
watch_accident_btn.config(font=("Carbon Block", 20))
```

```
watch_144_btn.config(font=("Carbon Block", 20))
```

```
label.pack()
```

```
Mask_detection_btn.pack()
```

```
Face_or_obj_tracking_btn.pack()
```

```
theft_detection_btn.pack()
```

```
watch_accident_btn.pack()
```

```
watch_144_btn.pack()
```

```
names.pack()
```

```
nd.pack()

navin.pack()

sakthi.pack()

selva.pack()

Mask_detection_btn.place(x=250, y=250)

Face_or_obj_tracking_btn.place(x=470, y=250)

theft_detection_btn.place(x=700, y=250)

watch_accident_btn.place(x=930, y=250)

watch_144_btn.place(x=1130, y=250)

main_menu.mainloop()
```

Face_Mask_Detection.py

```
# import the necessary packages

from tensorflow.keras.applications.mobilenet_v2 import preprocess_input

from tensorflow.keras.preprocessing.image import img_to_array

from tensorflow.keras.models import load_model

from imutils.video import VideoStream

import numpy as np

import imutils

import time
```

```

import cv2

import os

def detect_and_predict_mask(frame, faceNet, maskNet):

    # grab the dimensions of the frame and then construct a blob

    # from it

    (h, w) = frame.shape[:2]

    blob = cv2.dnn.blobFromImage(frame, 1.0, (224, 224),

                                  (104.0, 177.0, 123.0))

    # pass the blob through the network and obtain the face detections

    faceNet.setInput(blob)

    detections = faceNet.forward()

    print(detections.shape)

    # initialize our list of faces, their corresponding locations,

    # and the list of predictions from our face mask network

    faces = []

    locs = []

    preds = []

```

```

# loop over the detections

for i in range(0, detections.shape[2]):

    # extract the confidence (i.e., probability) associated with

    # the detection

    confidence = detections[0, 0, i, 2]

    # filter out weak detections by ensuring the confidence is

    # greater than the minimum confidence

    if confidence > 0.5:

        # compute the (x, y)-coordinates of the bounding box for

        # the object

        box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])

        (startX, startY, endX, endY) = box.astype("int")

        # ensure the bounding boxes fall within the dimensions of

        # the frame

        (startX, startY) = (max(0, startX), max(0, startY))

        (endX, endY) = (min(w - 1, endX), min(h - 1, endY))

```

```

        # extract the face ROI, convert it from BGR to RGB channel

        # ordering, resize it to 224x224, and preprocess it

        face = frame[startY:endY, startX:endX]

        face = cv2.cvtColor(face, cv2.COLOR_BGR2RGB)

        face = cv2.resize(face, (224, 224))

        face = img_to_array(face)

        face = preprocess_input(face)


        # add the face and bounding boxes to their respective

        # lists

        faces.append(face)

        locs.append((startX, startY, endX, endY))


# only make a predictions if at least one face was detected
if len(faces) > 0:

    # for faster inference we'll make batch predictions on *all*

    # faces at the same time rather than one-by-one predictions

    # in the above `for` loop

    faces = np.array(faces, dtype="float32")

    preds = maskNet.predict(faces, batch_size=32)

```

```

        # return a 2-tuple of the face locations and their corresponding
        # locations

    return (locs, preds)

# load our serialized face detector model from disk

prototxtPath = r"face_detector\deploy.prototxt"

weightsPath = r"face_detector\res10_300x300_ssd_iter_140000.caffemodel"

faceNet = cv2.dnn.readNet(prototxtPath, weightsPath)

# load the face mask detector model from disk

maskNet = load_model("mask_detector.model")

# initialize the video stream

print("Loading.....")

vs = VideoStream(src=0).start()

img_counter = 0

# loop over the frames from the video stream

while True:

```

```

# grab the frame from the threaded video stream and resize it

# to have a maximum width of 400 pixels

frame = vs.read()

frame = imutils.resize(frame, width=800)


# detect faces in the frame and determine if they are wearing a
# face mask or not

(locs, preds) = detect_and_predict_mask(frame, faceNet, maskNet)


# loop over the detected face locations and their corresponding
# locations

for (box, pred) in zip(locs, preds):

    # unpack the bounding box and predictions

    (startX, startY, endX, endY) = box

    (mask, withoutMask) = pred


    # determine the class and color we'll use to draw

    # the bounding box and text

    label = "Mask Found" if mask > withoutMask else "Wear Your Mask"

    color = (0, 255, 0) if label == "Mask" else (0, 0, 255)

```



```
cv2.imwrite("Without_Mask_{}.png".format(img_counter), frame) if  
label=="Wear Your Mask" else ""
```

```
# include the probability in the label
```

```
#label = "{}: {:.2f}%".format(label, max(mask, withoutMask) * 100)
```

```
label = label
```

```
# display the label and bounding box rectangle on the output
```

```
# frame
```

```
cv2.putText(frame, label, (startX, startY -  
10),cv2.FONT_HERSHEY_SIMPLEX, 0.45, color, 2)
```

```
cv2.rectangle(frame, (startX, startY), (endX, endY), color, 2)
```

```
img_counter=img_counter+1
```

```
# show the output frame
```

```
cv2.imshow("Frame", frame)
```

```
key = cv2.waitKey(1) & 0xFF
```

```
# if the `q` key was pressed, break from the loop
```

```
if key == ord("q"):
```

```
break
```

```
# do a bit of cleanup
```

```
cv2.destroyAllWindows()
```

```
vs.stop()
```

Face_or_object_tracking.py

```
import cv2
```

```
cap=cv2.VideoCapture(0)
```

```
tracker=cv2.TrackerMOSSE_create()
```

```
#tracker=cv2.TrackerCSRT_create()
```

```
success, img= cap.read()
```

```
box=cv2.selectROI("Tracking",img,False)
```

```
#image=cv2.imread("Selva.jpg",1)
```

```
#box=cv2.selectROI("Tracking",image,False)
```

```
tracker.init(img,box)
```

```
def drawBox(img,box):
```

```
    x,y,w,h=int(box[0]),int(box[1]),int(box[2]),int(box[3])
```

```

cv2.rectangle(img,(x,y),((x+w),(y+h)),(255,0,255),3,1)

#cv2.putText(img,"Tracking",(75,75),cv2.FONT_HERSHEY_SIMPLEX,0.7,(0,0,255),2)

while True:

    timer=cv2.getTickCount()

    success, img= cap.read()

    success, box=tracker.update(img)

    print(box)

    if success:

        drawBox(img,box)

    else:

        cv2.putText(img,"No
Object",(75,75),cv2.FONT_HERSHEY_SIMPLEX,0.7,(0,0,255),2)

    fps=cv2.getTickFrequency()/(cv2.getTickCount()-timer)

    #cv2.putText(img,str(int(fps)),(75,50),cv2.FONT_HERSHEY_SIMPLEX,0.7,(0,0,255),2)

    cv2.imshow("Tracking",img)

    if cv2.waitKey(1) & 0xff==ord('q'):

        break;

```

Theft_Detection.py

```
import numpy as np

import face_recognition as fr

import cv2

video_capture = cv2.VideoCapture(0)

bruno_image = fr.load_image_file("Selva.jpg")

bruno_face_encoding = fr.face_encodings(bruno_image)[0]

known_face_encodings = [bruno_face_encoding]

known_face_names = ["Selva"]

while True:

    ret, frame = video_capture.read()

    rgb_frame = frame[:, :, ::-1]

    face_locations = fr.face_locations(rgb_frame)
```

```

face_encodings = fr.face_encodings(rgb_frame, face_locations)

for (top, right, bottom, left), face_encoding in zip(face_locations, face_encodings):

    matches = fr.compare_faces(known_face_encodings, face_encoding)

    name = "Unknown"

    face_distances = fr.face_distance(known_face_encodings, face_encoding)

    best_match_index = np.argmin(face_distances)

    if matches[best_match_index]:

        name = known_face_names[best_match_index]

    cv2.rectangle(frame, (left, top), (right, bottom), (0, 0, 255), 2)

    cv2.rectangle(frame, (left, bottom - 35), (right, bottom), (0, 0, 255), cv2.FILLED)

    font = cv2.FONT_HERSHEY_SIMPLEX

    cv2.putText(frame, name, (left + 6, bottom - 6), font, 1.0, (255, 255, 255), 1)

cv2.imshow('Webcam_facerecognition', frame)

```

```
    if cv2.waitKey(1) & 0xFF == ord('q'):

        break

video_capture.release()

cv2.destroyAllWindows()
```

watch_accident.py

```
import os

import cv2

import numpy as np

import tensorflow as tf

import sys

from utils import label_map_util

# This is needed since the notebook is stored in the object_detection folder.

sys.path.append("..")

# Import utilites

from utils import label_map_util

from utils import visualization_utils as vis_util
```

```
# Name of the directory containing the object detection module we're using

MODEL_NAME = 'inference_graph'

VIDEO_NAME = 'test.mov'


# Grab path to current working directory

CWD_PATH = os.getcwd()


# Path to frozen detection graph .pb file, which contains the model that is used
# for object detection.

PATH_TO_CKPT =
os.path.join(CWD_PATH,MODEL_NAME,'frozen_inference_graph.pb')


# Path to label map file

PATH_TO_LABELS = os.path.join(CWD_PATH,'training','labelmap.pbtxt')


# Path to video

PATH_TO_VIDEO = os.path.join(CWD_PATH,VIDEO_NAME)


# Number of classes the object detector can identify

NUM_CLASSES = 6
```

```

label_map = label_map_util.load_labelmap(PATH_TO_LABELS)

categories = label_map_util.convert_label_map_to_categories(label_map,
max_num_classes=NUM_CLASSES, use_display_name=True)

category_index = label_map_util.create_category_index(categories)


# Load the Tensorflow model into memory.

detection_graph = tf.Graph()

with detection_graph.as_default():

    od_graph_def = tf.GraphDef()

    with tf.gfile.GFile(PATH_TO_CKPT, 'rb') as fid:

        serialized_graph = fid.read()

        od_graph_def.ParseFromString(serialized_graph)

        tf.import_graph_def(od_graph_def, name='')


    sess = tf.Session(graph=detection_graph)


# Define input and output tensors (i.e. data) for the object detection classifier


# Input tensor is the image

image_tensor = detection_graph.get_tensor_by_name('image_tensor:0')

```



```

# Output tensors are the detection boxes, scores, and classes

# Each box represents a part of the image where a particular object was detected

detection_boxes = detection_graph.get_tensor_by_name('detection_boxes:0')


# Each score represents level of confidence for each of the objects.

# The score is shown on the result image, together with the class label.

detection_scores = detection_graph.get_tensor_by_name('detection_scores:0')

detection_classes = detection_graph.get_tensor_by_name('detection_classes:0')


# Number of objects detected

num_detections = detection_graph.get_tensor_by_name('num_detections:0')


# Open video file

video = cv2.VideoCapture(PATH_TO_VIDEO)


while(video.isOpened()):

    # Acquire frame and expand frame dimensions to have shape: [1, None, None, 3]

    # i.e. a single-column array, where each item in the column has the pixel RGB value

    ret, frame = video.read()

    frame_expanded = np.expand_dims(frame, axis=0)

```

```
# Perform the actual detection by running the model with the image as input
```

```
(boxes, scores, classes, num) = sess.run(
```

```
[detection_boxes, detection_scores, detection_classes, num_detections],
```

```
feed_dict={image_tensor: frame_expanded})
```

```
# Draw the results of the detection (aka 'visulaize the results')
```

```
vis_util.visualize_boxes_and_labels_on_image_array(
```

```
frame,
```

```
np.squeeze(boxes),
```

```
np.squeeze(classes).astype(np.int32),
```

```
np.squeeze(scores),
```

```
category_index,
```

```
use_normalized_coordinates=True,
```

```
line_thickness=8,
```

```
min_score_thresh=0.80)
```

```
# All the results have been drawn on the frame, so it's time to display it.
```

```
cv2.imshow('Object detector', frame)
```

```
# Press 'q' to quit

if cv2.waitKey(1) == ord('q'):

    break


# Clean up

video.release()

cv2.destroyAllWindows()
```

watch_144.py

```
from mylib.centroidtracker import CentroidTracker

from mylib.trackableobject import TrackableObject

from imutils.video import VideoStream

from imutils.video import FPS

from mylib.mailer import Mailer

from mylib import config, thread

import time, schedule, csv

import numpy as np

import argparse, imutils

import time, dlib, cv2, datetime

from itertools import zip_longest
```

```
t0 = time.time()
```

```
def run():
```

```
    # construct the argument parse and parse the arguments
```

```
    ap = argparse.ArgumentParser()
```

```
    ap.add_argument("-p", "--prototxt", required=False,
```

```
                    help="path to Caffe 'deploy' prototxt file")
```

```
    ap.add_argument("-m", "--model", required=True,
```

```
                    help="path to Caffe pre-trained model")
```

```
    ap.add_argument("-i", "--input", type=str,
```

```
                    help="path to optional input video file")
```

```
    ap.add_argument("-o", "--output", type=str,
```

```
                    help="path to optional output video file")
```

```
    # confidence default 0.4
```

```
    ap.add_argument("-c", "--confidence", type=float, default=0.4,
```

```
                    help="minimum probability to filter weak detections")
```

```
    ap.add_argument("-s", "--skip-frames", type=int, default=30,
```

```
                    help="# of skip frames between detections")
```

```

args = vars(ap.parse_args())

# initialize the list of class labels MobileNet SSD was trained to
# detect

CLASSES = ["background", "aeroplane", "bicycle", "bird", "boat",
            "bottle", "bus", "car", "cat", "chair", "cow", "diningtable",
            "dog", "horse", "motorbike", "person", "pottedplant", "sheep",
            "sofa", "train", "tvmonitor"]

# load our serialized model from disk

net = cv2.dnn.readNetFromCaffe(args["prototxt"], args["model"])

# if a video path was not supplied, grab a reference to the ip camera
if not args.get("input", False):

    print("[INFO] Starting the live stream..")

    vs = VideoStream(config.url).start()

    time.sleep(2.0)

# otherwise, grab a reference to the video file
else:

```

```
print("[INFO] Starting the video..")

vs = cv2.VideoCapture(args["input"])

# initialize the video writer (we'll instantiate later if need be)

writer = None

# initialize the frame dimensions (we'll set them as soon as we read
# the first frame from the video)

W = None

H = None

# instantiate our centroid tracker, then initialize a list to store
# map each unique object ID to a TrackableObject

ct = CentroidTracker(maxDisappeared=40, maxDistance=50)

trackers = []

trackableObjects = { }

totalFrames = 0

totalDown = 0
```

```

totalUp = 0

x = []

empty=[]

empty1=[]


# start the frames per second throughput estimator

fps = FPS().start()


if config.Thread:

    vs = thread.ThreadingClass(config.url)


# loop over frames from the video stream

while True:

    # grab the next frame and handle if we are reading from either

    # VideoCapture or VideoStream

    frame = vs.read()

    frame = frame[1] if args.get("input", False) else frame


    # if we are viewing a video and we did not grab a frame then we

```

```

if args["input"] is not None and frame is None:

    break

# resize the frame to have a maximum width of 500 pixels (the
# less data we have, the faster we can process it), then convert
# the frame from BGR to RGB for dlib

frame = imutils.resize(frame, width = 500)

rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)


# if the frame dimensions are empty, set them

if W is None or H is None:

    (H, W) = frame.shape[:2]


# if we are supposed to be writing a video to disk, initialize
# the writer

if args["output"] is not None and writer is None:

    fourcc = cv2.VideoWriter_fourcc(*"MJPG")

    writer = cv2.VideoWriter(args["output"], fourcc, 30,

        (W, H), True)

```



```

# initialize the current status along with our list of bounding

status = "Waiting"

rects = []

if totalFrames % args["skip_frames"] == 0:

    # set the status and initialize our new set of object trackers

    status = "Detecting"

    trackers = []

    # convert the frame to a blob and pass the blob through the

    blob = cv2.dnn.blobFromImage(frame, 0.007843, (W, H), 127.5)

    net.setInput(blob)

    detections = net.forward()

    # loop over the detections

    for i in np.arange(0, detections.shape[2]):

        # extract the confidence (i.e., probability) associated

        # with the prediction

        confidence = detections[0, 0, i, 2]

```

```

# filter out weak detections by requiring a minimum

# confidence

if confidence > args["confidence"]:

    # extract the index of the class label from the

    # detections list

    idx = int(detections[0, 0, i, 1])

    # if the class label is not a person, ignore it

    if CLASSES[idx] != "person":

        continue

    # compute the (x, y)-coordinates of the bounding box

    # for the object

    box = detections[0, 0, i, 3:7] * np.array([W, H, W, H])

    (startX, startY, endX, endY) = box.astype("int")

    tracker = dlib.correlation_tracker()

    rect = dlib.rectangle(startX, startY, endX, endY)

    tracker.start_track(rgb, rect)

```

```
trackers.append(tracker)

else:

    # loop over the trackers

    for tracker in trackers:

        status = "Tracking"

        # update the tracker and grab the updated position

        tracker.update(rgb)

        pos = tracker.get_position()

        # unpack the position object

        startX = int(pos.left())

        startY = int(pos.top())

        endX = int(pos.right())

        endY = int(pos.bottom())
```

```

        # add the bounding box coordinates to the rectangles list

        rects.append((startX, startY, endX, endY))

cv2.line(frame, (0, H // 2), (W, H // 2), (0, 0, 0), 3)

cv2.putText(frame, "-Prediction border - Entrance-", (10, H - ((i * 20) + 200)),

            cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 0), 1)

# use the centroid tracker to associate the (1) old object

# centroids with (2) the newly computed object centroids

objects = ct.update(rects)

# loop over the tracked objects

for (objectID, centroid) in objects.items():

    # check to see if a trackable object exists for the current

    # object ID

    to = trackableObjects.get(objectID, None)

    # if there is no existing trackable object, create one

    if to is None:

        to = TrackableObject(objectID, centroid)

```

else:

```
y = [c[1] for c in to.centroids]
```

```
direction = centroid[1] - np.mean(y)
```

```
to.centroids.append(centroid)
```

```
# check to see if the object has been counted or not
```

```
if not to.counted:
```

```
    if direction < 0 and centroid[1] < H // 2:
```

```
        totalUp += 1
```

```
        empty.append(totalUp)
```

```
        to.counted = True
```

```
    elif direction > 0 and centroid[1] > H // 2:
```

```
        totalDown += 1
```

```
        empty1.append(totalDown)
```

```
        #print(empty1[-1])
```

```

x = []

# compute the sum of total people inside

x.append(len(empty1)-len(empty))

#print("Total people inside:", x)

# if the people limit exceeds over threshold,
send an email alert

if sum(x) >= config.Threshold:

    cv2.putText(frame, "-ALERT: People
limit exceeded-", (10, frame.shape[0] - 80),

    cv2.FONT_HERSHEY_COMPLEX, 0.5, (0, 0, 255), 2)

    if config.ALERT:

        print("[INFO] Sending email
alert..")

        Mailer().send(config.MAIL)

        print("[INFO] Alert sent")

to.counted = True

# store the trackable object in our dictionary

trackableObjects[objectID] = to

```

```

text = "ID {}".format(objectID)

cv2.putText(frame, text, (centroid[0] - 10, centroid[1] - 10),

              cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 2)

cv2.circle(frame, (centroid[0], centroid[1]), 4, (255, 255, 255), -1)


# construct a tuple of information we will be displaying on the

info = [

    ("Exit", totalUp),

    ("Enter", totalDown),

    ("Status", status),

]


info2 = [

    ("Total people inside", x),

]


# Display the output

for (i, (k, v)) in enumerate(info):

    text = "{}: {}".format(k, v)

```

```

        cv2.putText(frame, text, (10, H - ((i * 20) + 20)),
cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 0, 0), 2)

    for (i, (k, v)) in enumerate(info2):

        text = "{}: {}".format(k, v)

        cv2.putText(frame, text, (265, H - ((i * 20) + 60)),
cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255, 255, 255), 2)

    # Initiate a simple log to save data at end of the day

    if config.Log:

        datetimee = [datetime.datetime.now()]

        d = [datetimee, empty1, empty, x]

        export_data = zip_longest(*d, fillvalue = "")

        with open('Log.csv', 'w', newline=") as myfile:

            wr = csv.writer(myfile, quoting=csv.QUOTE_ALL)

            wr.writerow(("End Time", "In", "Out", "Total Inside"))

            wr.writerows(export_data)

    # show the output frame

    cv2.imshow("Real-Time Monitoring/Analysis Window", frame)

    key = cv2.waitKey(1) & 0xFF

```



```

# if the `q` key was pressed, break from the loop

if key == ord("q"):

    break

totalFrames += 1

fps.update()

if config.Timer:

    # Automatic timer to stop the live stream. Set to 8 hours (28800s).

    t1 = time.time()

    num_seconds=(t1-t0)

    if num_seconds > 28800:

        break

# stop the timer and display FPS information

fps.stop()

print("[INFO] elapsed time: {:.2f}".format(fps.elapsed()))

print("[INFO] approx. FPS: {:.2f}".format(fps.fps()))

cv2.destroyAllWindows()

```

```
if config.Scheduler:
```

```
    schedule.every().day.at("9:00").do(run)
```

```
    while 1:
```

```
        schedule.run_pending()
```

```
else:
```

```
    run()
```

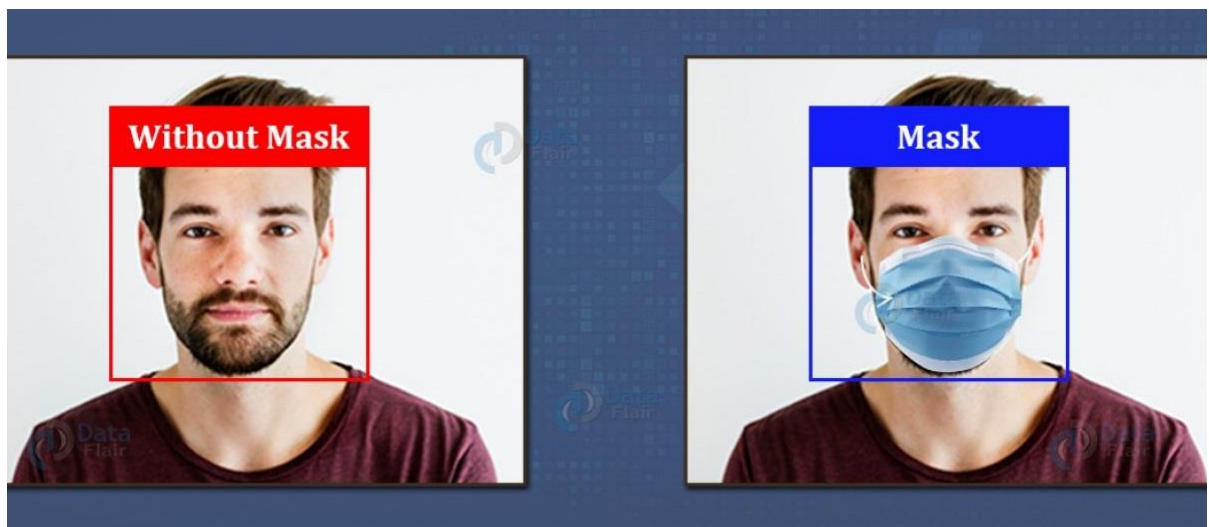
APPENDIX – 2

A.2 SCREN SHOTS

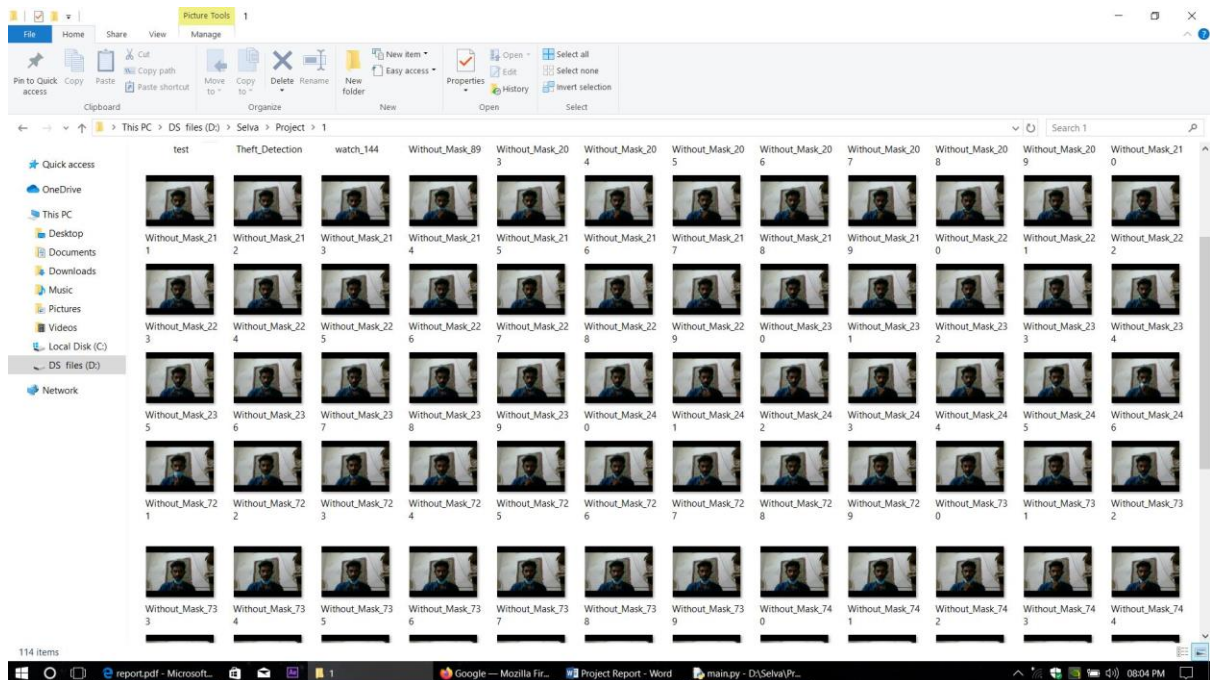
Main_window.py



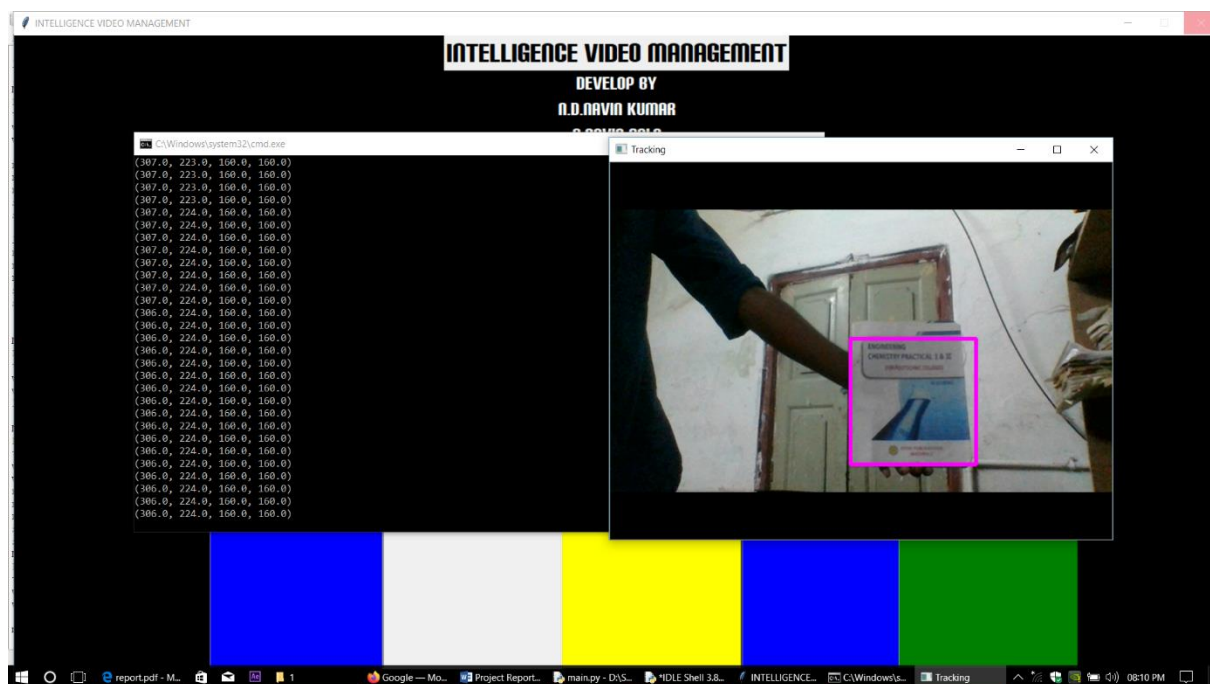
Face_Mask_Detection



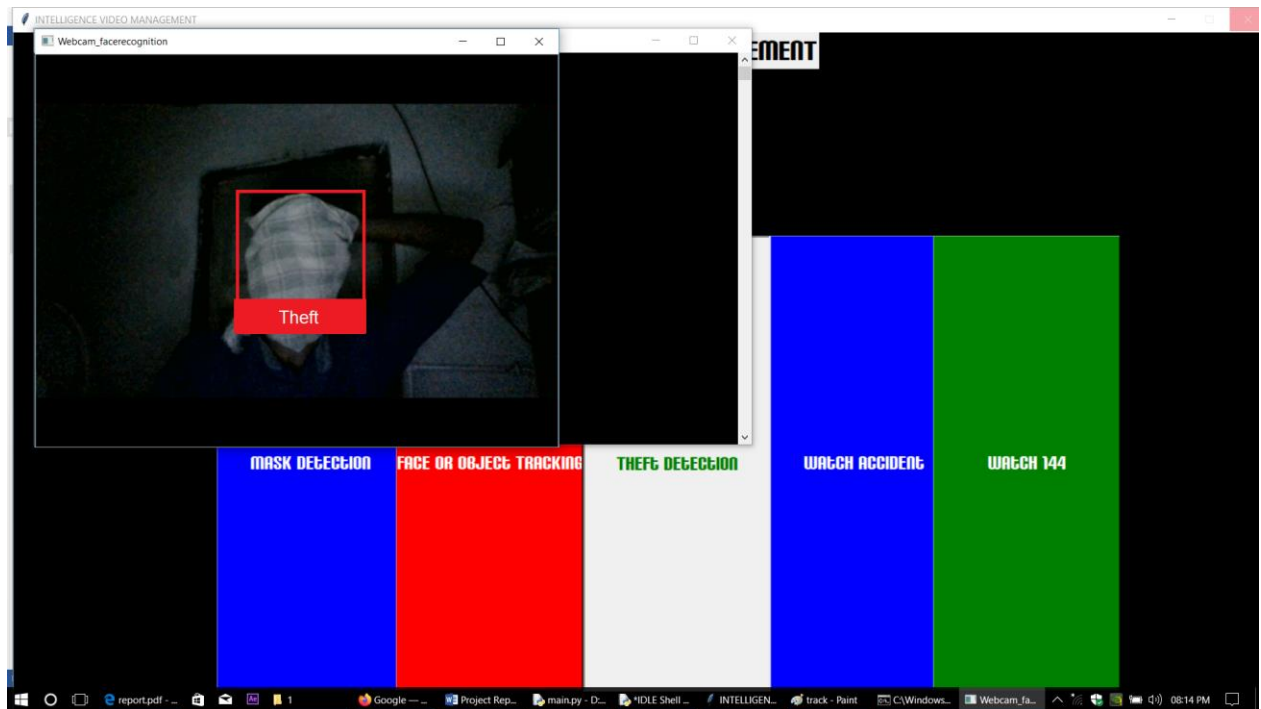
If you without wearing the mask, now capture the data.



Object tracking



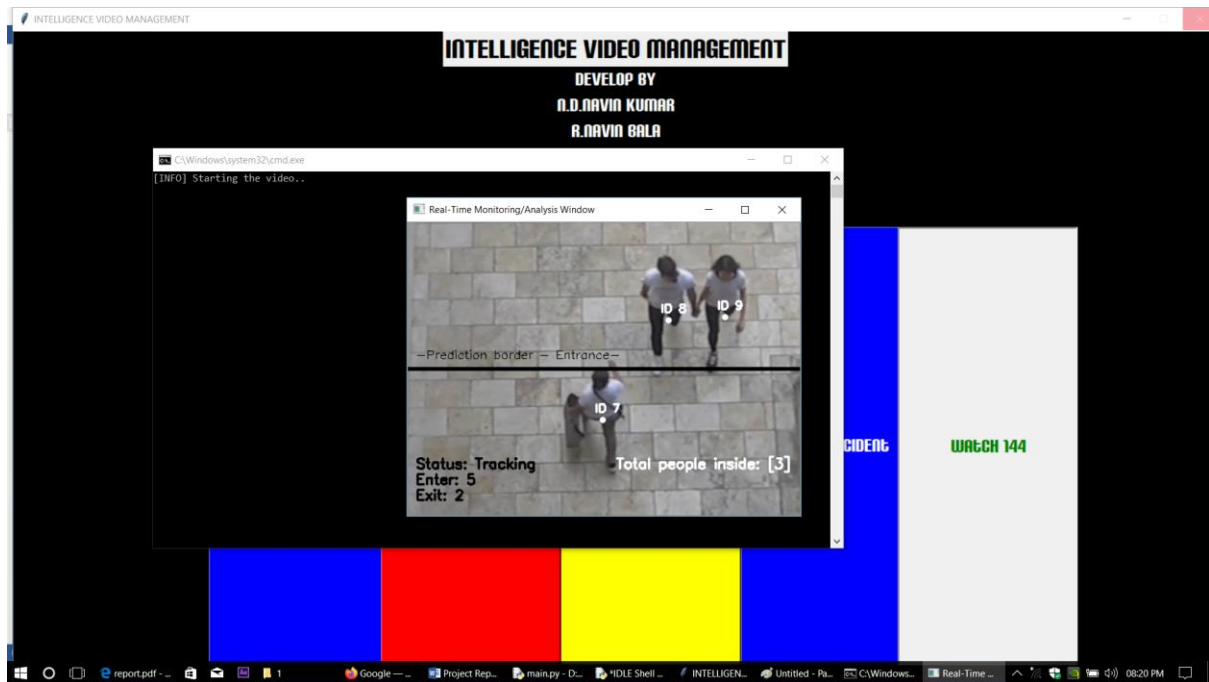
Theft Detection



Accident Detection



Watching People Count



REFERENCES

1. Felzenszwalb, P. F. and Girshick, R. B. and McAllester, D. and Ramanan, September 2010, D.Object Detection with Discriminatively Trained Part Based Models. PAMI, vol. 32, no. 9, pp. 1627-1645.
2. Martinez, A and Kak, A.PCA versus LDAIEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 23, No.2, pp. 228-233, 2001.
3. Brunelli, R., Poggio, T.Face Recognition through Geometrical Features.European Conference on Computer Vision (ECCV) 1992, S. 792–800.
4. Ahonen, T., Hadid, A., and Pietikainen, M.Face Recognition with Local Binary Patterns.Computer Vision- ECCV 2004 (2004), 469–481.
5. Burt, P., and Adelson, E. H., A Multiresolution Spline with Application to Image Mosaics. ACM Transactions on Graphics, 2(4):217-236, 1983.
6. Brown and D. Lowe. Automatic Panoramic Image Stitching using Invariant Features. International Journal of Computer Vision, 74(1), pages 59-73, 2007.
7. LeCun, L. Bottou, G.B. Orr and K.-R. Muller, Efficient backprop, in Neural Networks—Tricks of the Trade, Springer Lecture Notes in Computer Sciences 1524, pp.5-50, 1998.
8. Hastie, T., Tibshirani, R., Friedman, J. H.The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer Series in Statistics. 2001.
9. A. P. Jain and P. Dandannavar, “Application of machine learning techniques to sentiment analysis,” in 2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT), 2016, pp. 628–632.
10. Khodi Jahhulliyah, Normi shamavang Abubakar, “Emotion recognition and brain mapping for sentiment analysis”, 2017.