

# **FACE MASK AND SOCIAL DISTANCING DETECTION USING ML TECHNIQUE**

## **PROJECT REPORT**

*Submitted by*

|                         |          |                     |
|-------------------------|----------|---------------------|
| <b>AJET GANAPATHY A</b> | <b>-</b> | <b>813819205005</b> |
| <b>MALOLAN B A</b>      | <b>-</b> | <b>813819205034</b> |
| <b>MANOJ DEEPAK S</b>   | <b>-</b> | <b>813819205035</b> |
| <b>SRINIVASAN M A</b>   | <b>-</b> | <b>813819205057</b> |

*in partial fulfillment for the award of the degree*

*of*

**BACHELOR OF TECHNOLOGY**

**in**

**INFORMATION TECHNOLOGY**



**SARANATHAN COLLEGE OF ENGINEERING, TIRUCHIRAPALLI**



**ANNA UNIVERSITY :: CHENNAI 600 025**

**MAY 2023**

**ANNA UNIVERSITY :: CHENNAI 600 025**

**BONAFIDE CERTIFICATE**

Certified that this project report “**FACE MASK AND SOCIAL DISTANCING DETECTION USING ML TECHNIQUE**” is the bonafide work of

|                         |   |                     |
|-------------------------|---|---------------------|
| <b>AJET GANAPATHY A</b> | - | <b>813819205005</b> |
| <b>MANOJ DEEPAK S</b>   | - | <b>813819205035</b> |
| <b>MALOLAN B A</b>      | - | <b>813819205034</b> |
| <b>SRINIVASAN M A</b>   | - | <b>813819205057</b> |

who carried out the project work under my supervision.

**SIGNATURE**

Dr.R.Thillaikarasi, M.Tech., Ph.d.,  
**HEAD OF THE DEPARTMENT,**  
Professor,  
Information Technology,  
Saranathan College of Engineering,  
Panjapur,  
Trichy-620012.

**SIGNATURE**

Ms.A.Sheelavathi, M.E.,  
**SUPERVISOR,**  
Assistant Professor,  
Information Technology,  
Saranathan College of Engineering,  
Panjapur,  
Trichy-620012.

# **VIVA – VOCE EXAMINATION**

## **FACE MASK AND SOCIAL DISTANCING DETECTION USING ML TECHNIQUE**

*Submitted by*

|                         |                     |
|-------------------------|---------------------|
| <b>AJET GANAPATHY A</b> | <b>813819205005</b> |
| <b>MANOJ DEEPAK S</b>   | <b>813819205035</b> |
| <b>MALOLAN B A</b>      | <b>813819205034</b> |
| <b>SRINIVASAN M A</b>   | <b>813819205057</b> |

The viva – voce Examination of this project work done as a part of B.Tech  
Information Technology was held on \_\_\_\_\_

**INTERNAL EXAMINER**

**EXTERNAL EXAMINER**

## ACKNOWLEDGEMENT

We sincerely thank Shri. **S. RAVINDRAN, Secretary, Saranathan College of Engineering**, for giving us a platform to realize our project.

We express our sincere thanks to **Dr. D. VALAVAN, Ph.D., Principal, Saranathan College of Engineering**, for giving us an opportunity and immense support for the successful completion of the project.

We are obliged to **Dr. R. THIILAIKARASI Professor, M. Tech., Ph.D., Professor & Head of Department, Information Technology, Saranathan College of Engineering**, for her valuable suggestion and encouragement to our project. We our heartfelt thanks to our project guide **Ms. A. SHEELAVATHI, M.E., Assistant Professor, Department of Information Technology, Saranathan College of Engineering**, for us with her valuable ideas.

We would like to thank all our department faculty members and technical assistant for their support and help rendered by them in completion of this project.

We would like to thank our parents for their constant encouragement and support without which this project would not be possible. Last but not the least we would like to thank our friends who have been instrumental in providing idea and material for the construction of our project.

Above all, we thank the God almighty for his bountiful blessings.

## ABSTRACT

Face Mask detection is the process to check face mask in their customer face in the shop. While a person enters into the shop the person will get checked by the system, whether the person wear mask or not. If the person does not wear mask means it will send an alert message to wear mask. Firstly, the system will detect their face and then it will detect the face features like nose and mouth. If it detects the face, nose and mouth system concludes that the customer wear mask. If it detects the face but it does not detect their nose and mouth, it will send an alert message as they don't wear mask and please wear mask. The user is required to input the focal length and the sensor dimensions of the camera to be used. The user is then required to position these 2 individuals at the minimum social distance that is to be maintained, henceforth referred to as the reference social distance. The advantage of this is that authorities can actually select the social distance they want maintained, according to the specific guidelines that they wish to follow. For example: WHO guidelines mention the recommended amount of social distance to be a minimum of 3ft, while CDC guidelines recommend the minimum to be 6 ft. The social distance between 2 people is solely judged relative to the initial calibration and the absolute distance need not be provided to the model. This social distancing model uses the principle that a camera lens is essentially a convex lens, where the image is essentially captured on the screen. For this model, as fore mentioned, we need the focal length and sensor dimensions. The focal length of a lens is the distance from the optic center of the lens to its focus. In optics and photography, the focal length is measured in millimeters (mm). Longer the focal length, higher the magnification, but lower the angle of view. An image sensor is a device found in the hardware of a camera, that uses light to detect information to convert a view into an image. The sensor essentially functions as screen where all the pixels in an image are mapped. Greater the pixels mapped, greater the image quality.

## TABLE OF CONTENTS

| CHAPTERS | TITLE   | PAGE<br>NO |
|----------|---|------------|
|          | <b>ABSTRACT</b>   | <b>V</b>   |
|          | <b>LIST OF TABLES</b>   | <b>IX</b>  |
|          | <b>LIST OF FIGURES</b>  | <b>X</b>   |
|          | <b>LIST OF ABBREVIATIONS</b>  | <b>XI</b>  |
| <b>1</b> | <b>INTRODUCTION</b>   |            |
|          | 1.1 Objective   | 1          |
|          | 1.2 Features of mask detection system and social distance maintenance                     | 1          |
|          | 1.3 Existing System   | 1          |
|          | 1.4 Proposed System   | 2          |
| <b>2</b> | <b>LITERATURE SURVEY</b>  |            |
|          | 2.1 A cascade framework for masked face Detection   | 3          |
|          | 2.2 Masked face detection using Viola Jones Algorithm                                     | 3          |
|          | 2..3 Face recognition-based attendance system Using Haar cascade and Local binary pattern | 4          |
|          | 2.4 Masked face recognition using Convolutional Neural Network                            | 5          |
| <b>3</b> | <b>SOFTWARE REQUIREMENT SPECIFICATION</b>   |            |
|          | 3.1 Introduction  | 7          |
|          | 3.1.1 Purpose   | 7          |
|          | 3.1.2 Scope   | 7          |
|          | 3.2 Overall Description   | 7          |

|         |   |    |
|---------|---|----|
| 3.2.1   | System Environment                          | 7  |
| 3.2.2   | Functional Requirements                     | 8  |
| 3.2.3   | Use Cases                                   | 8  |
| 3.2.3.1 | Face Detection module                       | 9  |
| 3.2.3.2 | Database Connection                         | 9  |
| 3.2.3.3 | To detect whether the face is masked or not | 10 |
| 3.2.3.4 | Social Distancing                           | 11 |
| 3.2.4   | Non-functional Requirements                 | 11 |
| 3.3     | Software Requirement Specification          | 11 |
| 3.3.1   | Functional Requirements                     | 11 |
| 3.3.1.1 | Entry module                                | 12 |
| 3.3.1.2 | Exit module                                 | 12 |
| 3.3.2   | Non-functional Requirements                 | 12 |
| 3.3.2.1 | Computer Requirements                       | 12 |

## **4 SYSTEM DESIGN**

|       |                     |    |
|-------|---------------------|----|
| 4.1   | Introduction        | 14 |
| 4.2   | System Architecture | 14 |
| 4.2.1 | Face Detection      | 16 |
| 4.2.2 | Mask Check          | 16 |
| 4.2.3 | Face Recognition    | 16 |
| 4.2.4 | Distance Check      | 18 |

## **5 IMPLEMENTATION**

|       |                                       |    |
|-------|---------------------------------------|----|
| 5.1   | Face detection and recognition module | 19 |
| 5.1.1 | Face detection module                 | 19 |
| 5.1.2 | Face recognition module               | 19 |
| 5.1.3 | Post processing                       | 20 |
| 5.2   | Face mask detection module            | 21 |
| 5.2.1 | Two phase COVID-19 face mask detector | 21 |

|           |                                       |           |
|-----------|---------------------------------------|-----------|
| 5.2.2     | How was our face mask dataset created | 22        |
| 5.3       | Tracking Social distancing            | 24        |
| 5.3.1     | YOLO-v3                               | 27        |
| <b>6</b>  | <b>TESTING</b>                        |           |
| 6.1       | Testing process                       | 32        |
| 6.2       | Testing objectives                    | 32        |
| 6.3       | Unit testing                          | 33        |
| 6.4       | Integration testing                   | 33        |
| 6.5       | Test results                          | 33        |
| 6.6       | Test cases                            | 34        |
| <b>7</b>  | <b>CONCLUSION</b>                     | <b>35</b> |
| <b>8.</b> | <b>FUTURE ENHANCEMENT</b>             | <b>36</b> |
|           | <b>APPENDICES</b>                     | <b>37</b> |
|           | <b>APPENDIX-1 SOURCE CODE</b>         | <b>37</b> |
|           | <b>APPENDIX -2 SCREENSHOTS</b>        | <b>60</b> |
|           | <b>REFERNCES</b>                      | <b>63</b> |

## LIST OF TABLES

| <b>TABLE<br/>NO.</b> | <b>NAME OF THE TABLE</b>          | <b>PAGE<br/>NO.</b> |
|----------------------|-----------------------------------|---------------------|
| Table 3.1            | Entry                             | 12                  |
| Table 3.2            | Exit                              | 12                  |
| Table 6.1            | Inception: Unit Testing           | 34                  |
| Table 6.2            | Batch Normalization: Unit Testing | 34                  |

## **LIST OF FIGURES**

| <b>FIGURE NO.</b> | <b>NAME OF FIGURE</b>          | <b>PAGE NO.</b> |
|-------------------|--------------------------------|-----------------|
| Fig 3.1           | Face Detection and Recognition | 7               |
| Fig 3.2           | Face Detection                 | 9               |
| Fig 3.3           | Mask Check                     | 10              |
| Fig 3.4           | Social Distancing              | 11              |
| Fig 4.1           | System Architecture            | 15              |
| Fig 4.2           | Face Detection                 | 16              |
| Fig 4.3           | Mask Check                     | 16              |
| Fig 4.4           | Face Recognition               | 17              |
| Fig 5.1           | Complete Architecture          | 21              |
| Fig 5.2           | Haar Features                  | 23              |
| Fig 5.3           | Mask Check                     | 24              |
| Fig 5.4           | Neural Net Architecture (YOLO) | 28              |
| Fig 5.5           | Birds Eye View                 | 31              |
| Fig 8.1           | Face Mask Detection            | 60              |
| Fig 8.2           | Face Detection                 | 61              |
| Fig 8.3           | Social distance Detection      | 61              |
| Fig 8.4           | Social distance Detection      | 62              |

## **LIST OF ABBREVIATIONS**

| <b>S. NO.</b> | <b>ABBREVIATIONS</b> | <b>WORDS AND PHRASES</b>       |
|---------------|----------------------|--------------------------------|
| 1             | LBPH                 | Local Binary Pattern Histogram |
| 2             | YOLO                 | You Only Look Once             |
| 3             | ML                   | Machine Learning               |
| 4             | Open CV              | Open Computer Vision           |

# **CHAPTER 1**

## **INTRODUCTION**

### **1.1 OBJECTIVE**

In this pandemic Covid-19, we are in the need of protecting us from spread of the disease. Mainly we are in the need to wear facial mask. People are gathered inside the shops and fail to maintain distance between them. So, the main goal of our project is to detect facial mask in moving video or images and alert the people on shops, if they are gathered together inside the mall. To check and monitor people to follow these safety precautions, we developed a facemask detection system. It will detect the face and then it will identify whether they wear mask or not. This system will distinguish between the faces with and without masks. If more faces detected with minimal distance then it alerts the system that social distancing is not maintained.

### **1.2 FEATURES OF MASK DETECTION SYSTEM AND SOCIAL DISTANCE MAINTAINANCE**

Monitoring whether the people are wearing mask or not is the major problem in this pandemic. This system will detect the faces and identify whether they wear mask or not. Thus, this system will help to prevent people from spreading of the deadly disease. While detecting faces, when more faces are detected with minimal distances then it alerts the system that social distancing is not maintained. This helps us to restrict count of aged people inside the mall and total count too. This helps us to avoid crowd inside shop/mall and they can purchase freely without crowd. While allowing the people inside the mall, checking of mask and saving the samples of their image helps to generate reports and can be retrieved any time (if necessary).

### **1.3 EXISTING SYSTEM**

The existing system has face mask detection and recognition system is implemented using CNN (Convolutional Neural Network) in Deep Learning. Face Detection and

Recognition is done to identify criminals in public crowding areas like railway station, Airports etc. Due to this pandemic, face mask checking is a must and a labour is used to check. a paid labour is used to check whether people are wearing mask or not in a shop/mall but by doing this people may get infected.

## **DRAWBACKS**

1. Occupies more Space
2. Labour cost is high

## **1.4 PROPOSED SYSTEM**

This system helps the people to maintain distance among them and avoid spreading of COVID. Initially, for a particular person who enter the mall their face will be detected with mask and recognized, then age and gender are identified. This system is attached to both the entry and exit system of the shopping mall. A database is created containing all the faces of the customers once their face gets detected. System in the entry area will count the person entry and it is designed to allow only 50 persons in the shopping mall. Once they exceed the limit it will give the alert message.

This alert message will be further carried out by using IoT (if recommended).

## **ADVANTAGES**

1. It is a time consuming process.
2. This frees up a lot of time for developers to use their time to more productive use.
3. Technical supporter is enough to maintain the system.
4. Maintenance cost is less compared to labour cost

## **CHAPTER 2**

### **2.1 A Cascade Framework for Masked Face Detection**

**AUTHOR:** Wei Buse, Jiangjian Xiao, Chuanhong Zhou, Minmin Yang, ChengbinPeng

**YEAR:**2017

#### **ABSTRACT**

Accurately and efficiently detecting masked faces is increasingly meaningful, since it can be applied on tracking and identifying criminals or terrorists. As a unique face detection task, masked face detection is much more difficult because of extreme occlusions which leads to the loss of face details. Besides, there is almost no existing large-scale accurately labeled masked face dataset, which increase the difficulty of masked face detection.

#### **ADVANTAGE**

The CNN-based deep learning algorithms has made great breakthroughs in many computer vision areas including face detection. In this paper, we propose a new CNN-based cascade framework, which consists of three carefully designed convolutional neural networks to detect masked faces. Besides, because of the shortage of masked face training samples, we propose a new dataset called "MASKED FACE dataset" to finetune our CNN models. We evaluate our proposed masked face detection algorithm on the MASKED FACE testing set, and it achieves satisfactory performance.

#### **LIMITATION**

Only after using fine tune model in dataset it worked well.

### **2.2 Masked Face Detection using the Viola Jones Algorithm**

**AUTHOR:** Aishwarya Radhakrishnan Nair, Dr. Amol D. Potgantwar Savitabai Phule

**YEAR:** 2017

## **ABSTRACT**

The motive of the work is to introduce automatic revelation of masked person in real time with a surveillance camera. The main aim is to detect masked person automatically in less time period using Viola Jones Algorithm. In this paper, the researcher proposes a system that uses four variant steps like calculating distance range of person from the camera, eye line detection and face part detection such as mouth detection and at last face detection. During face detection, if eyes are recognized and later if face is recognized, it signifies that there's no mask on the person's face. If eyes are recognized but face is not recognized, it signifies that individual has put a cover on rest of the face, that is the person wore mask. This unique approach for the problem has created a method transparent and easier in complexity so that the real time implementation can be made beneficial and workable.

## **ADVANTAGE**

By this technique, detection is to be done in an efficient way, and they were alerted in lesser time.

## **LIMITATION**

It is sensitive to light and produce wrong result if the face have more lightening effect.

## **2.3 Face Recognition based Attendance System using Haar Cascade and Local Binary Pattern Histogram**

**AUTHOR: Bharath Tej Chinimilli, Anjali T, Akhil Kotturi, Vihas Reddy Kaipu, Jathin Varma Mandapati**

**YEAR: 2020**

## **ABSTRACT**

Here, We have referred the Face Recognition using Haar Cascade and Local Binary Pattern histogram for our Project. In this Project they have used Haar Cascade and Local Binary Pattern Histogram for Face Recognition. The proposed Attendance system is based on haar cascade for face detection and LBPH for face recognition.

This system provides functionalities like taking images of students and training their images in the database. In our dataset, we have 60 samples of images of a single student. During the face detection, conversion of frame from color to grayscale to detect the faces Haar cascade feature is used to train and detect feature in image like eye, line etc. For Face Recognition we choose Local Binary Pattern creates an image which highlights the characteristics of image in a better way. It has the capability to recognize both front and side faces better compared to eigenfaces and Fisherfaces. It works better in different environments and light conditions than other algorithms. While Recognizing the image, the system displays the id of the image in the dataset which is stored during face detection matches with the Recognized image.

## **LIMITATION**

Here the dataset is small. When number of sample image is increased the accuracy of recognition of face gets increased.

## **2.4 Masked face recognition using convolutional neural network.**

**AUTHOR: Md. Sabbir Ejaz and Md. Rabiul Islam**

**YEAR: 2019**

## **ABSTRACT**

Recognition from faces is a popular and significant technology in recent years. Face alterations and the presence of different masks make it too much challenging. In the real world, when a person is uncooperative with the systems such as in video surveillance then masking is further common scenarios. For these masks, current face recognition performance degrades. An abundant number of researchers work has been performed for recognising faces under different conditions like challenging pose or illumination, degraded images, etc. Still, difficulties created by masks are usually disregarded. The primary concern to this work is about facial masks, and especially to enhance the recognition accuracy of different masked faces. A feasible approach has been proposed that consists of first detecting the facial regions.

The occluded face detection problem has been approached using Multi-Task cascaded convolutional neural network.

Then facial features extraction is performed by support vector machine. Experiments signify that this mentioned approach gives a remarkable performance on masked face recognition . Besides , its performance has been also evaluated within excessive facial masks and found attractive outcomes.

## **ADVANTAGE**

In which the Face is detected in well followed algorithm. t ensures a lightweight representation that makes the real-world masked face recognition process a feasible task. Moreover, the masked regions vary from one face to another, which leads to informative images of different sizes.

## **LIMITATION**

To show the performance and correctness of a classification algorithm , the precision -Recall curve is another visualization technique . For various threshold , the precision recall curve has been shown different changes. High precision defines a lower false positive rate.

## CHAPTER-3

### Software Requirement Specification

#### 3.1.1 Introduction

This software requirement specification provides a complete description of all the functions and specifications of our project Face mask and social distancing detection.

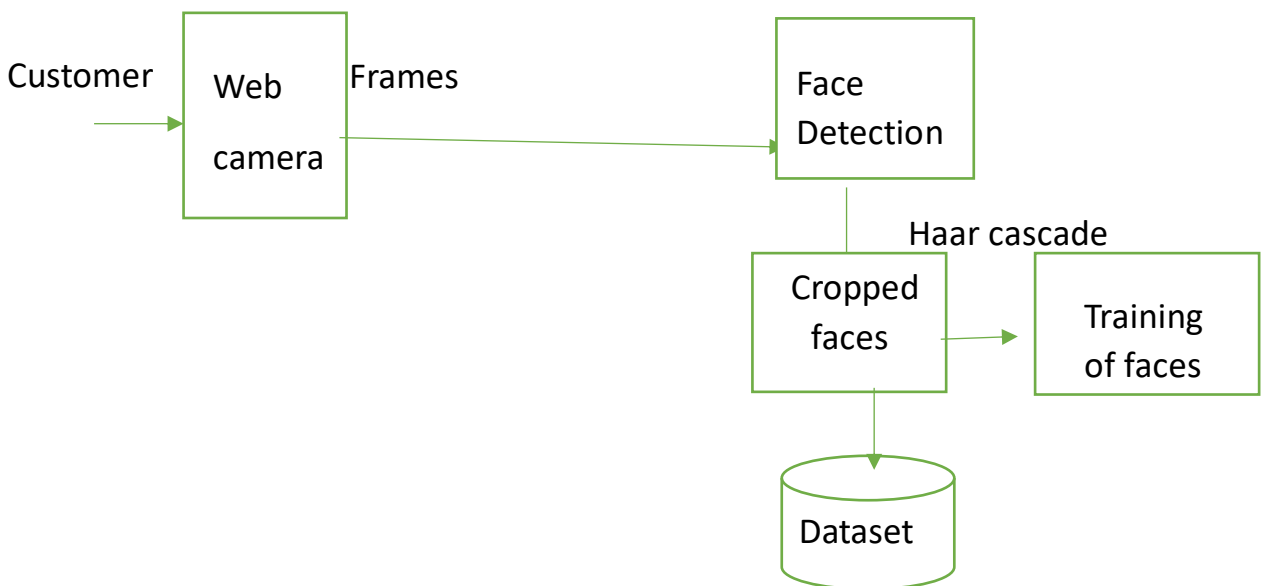
#### 3.1.2. Scope

This project is aimed at detecting face mask and social distancing between the people inside the shop.

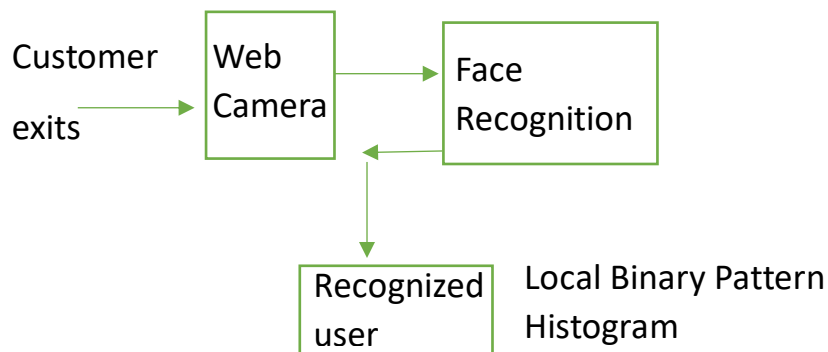
#### 3.2 Overall description

##### 3.2.1 System Environment

**Entry:**



**Exit:**



**Fig 3.1 Face Detection and Recognition**

## **Description**

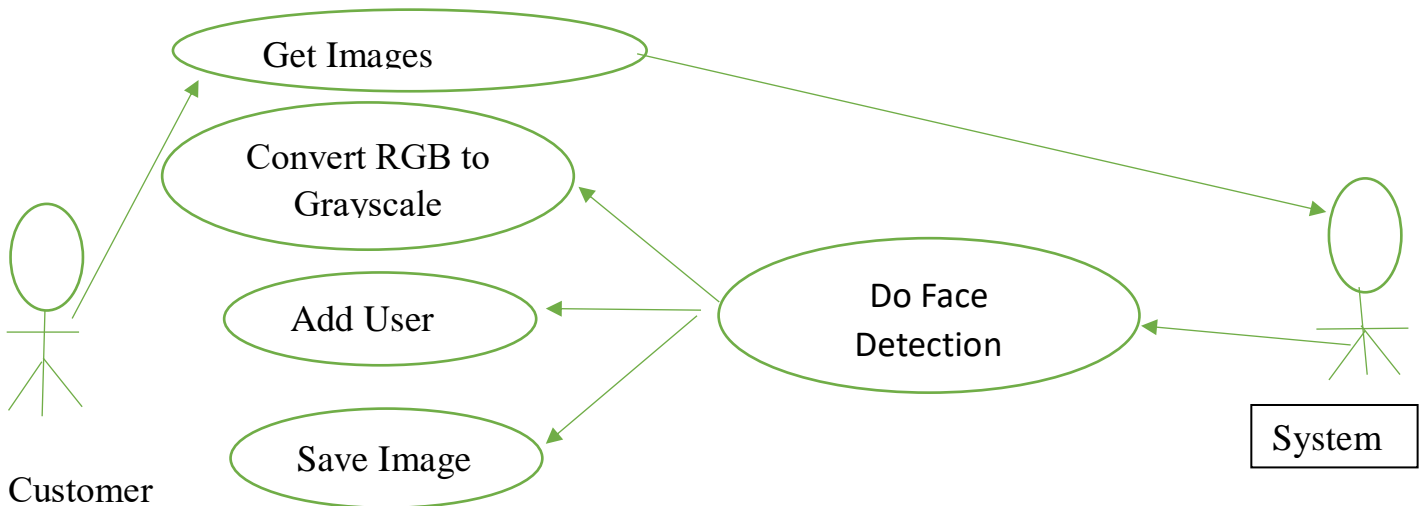
With the recent outbreak and rapid transmission of the COVID-19 pandemic, the need for the public to follow social distancing norms and wear masks in public is only increasing. According to WHO, to follow proper social distancing, people in public places must maintain at least 3ft or 1m distance between each other. This paper focuses on a solution to help enforce proper social distancing and wearing masks in public using object detection on video footage and images in real time. The experimental results shown in this paper infer that the detection of masked faces and human subjects has stronger robustness and faster detection speed as compared to its competitors. Our proposed object detection model achieved a mean average precision score of 94.75% with an inference speed of 38 FPS on video. The network ensures inference speed capable of delivering real-time results without compromising on accuracy, even in complex setups. The social distancing method proposed also yields promising results in several variable scenarios.

### **3.2.2 Functional Requirements**

Functional requirements are those that refer to functionality of the system what services it will provide to other information needed to produce the correct system and are detailed separately.

### 3.2.3 Use cases

#### 3.2.3.1 Face detection module



**Fig 3.2 Face Detection**

#### Description

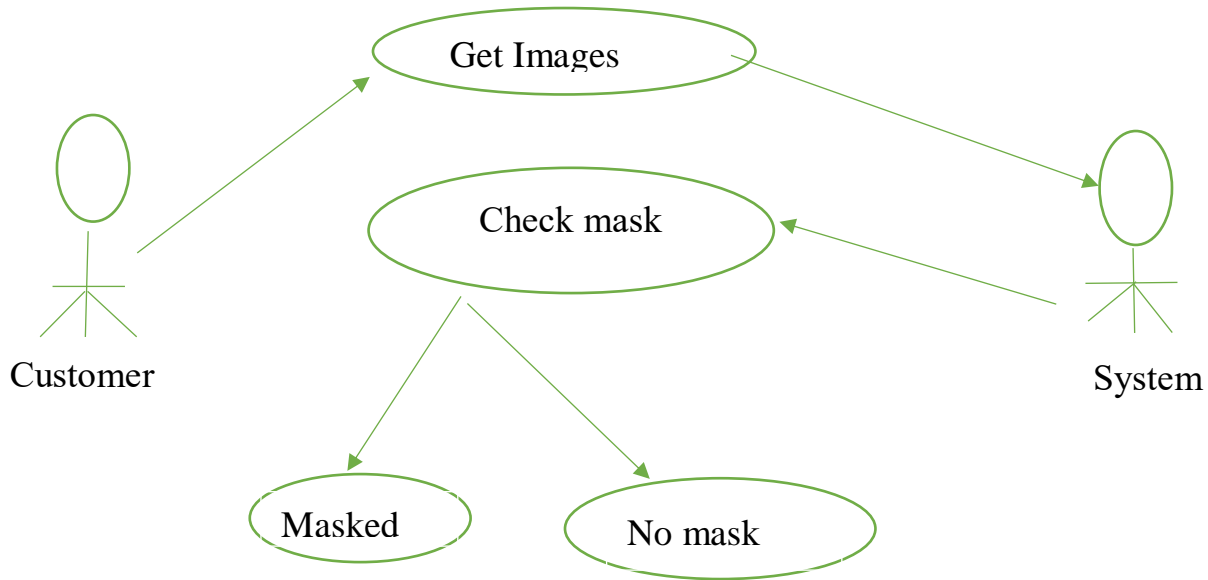
It is a Machine Learning based approach where a cascade function used to detect images. For Face Detection, Haar Cascade is used to detect faces which is an effective detection method. The Frame capture from the camera is initially sent to face detection system which will detect the faces in a frame. The faces are cropped and resized according to the requirement and stored in dataset for recognition. The face is detected using Haar feature based Cascade classifier is an effective detection method. This module is based on general face detection that needs to be generated when the face gets detected in the entry side.

#### 3.2.3.2 Database Connection

#### Description

The Sample face detected when the user enters at the entry point of the shop/mall. When the face gets detected, sample images are cropped and stored in dataset is used for face recognition. HaarCascade algorithm used to detect face and crop the face samples to be stored in dataset.

### 3.2.3.3 To detect whether the face is masked or not

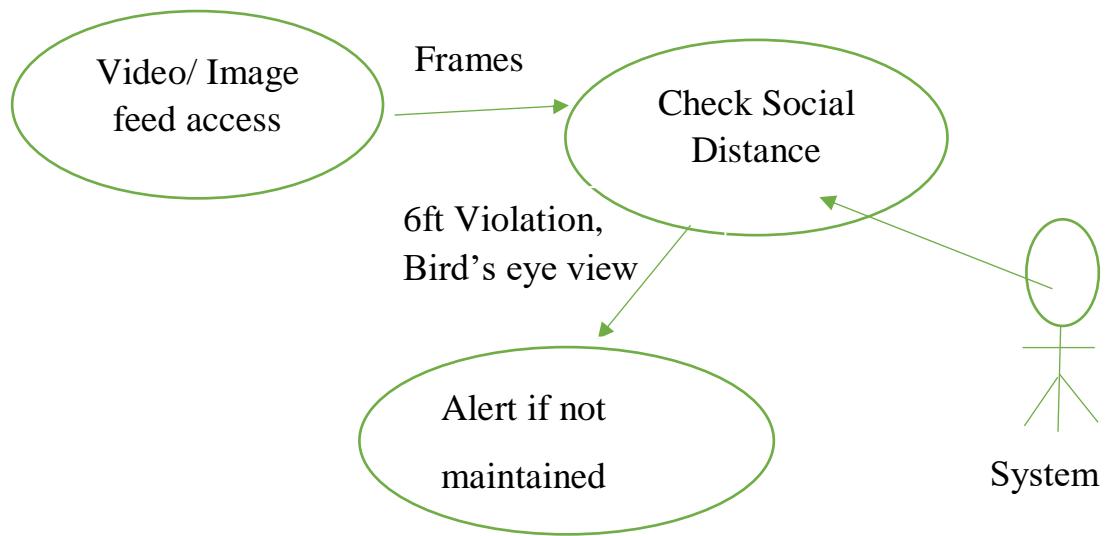


**Fig 3.3 Mask check**

#### **Description**

Due to this pandemic, wearing a face mask is mandatory everywhere, and thus it is important in shops so, the system checks whether a person is wearing a mask or not using HaarCascade algorithm. The person who are wearing mask indicates green framed box and those who are not wearing mask indicated red framed box. The system will detect the face, nose and mouth. If the system detect face, nose and mouth it conclude that the person wore mask otherwise it will send an alert message to wear mask.

### 3.2.3.4 Social Distancing maintenance



**Fig 3.4 Social Distancing**

#### **Description**

When the person who are not maintaining the social distancing, an alert message has been announced.

### 3.2.4 Non-Functional Requirements

Non-Functional Requirements are the requirements that are non-functional in nature. Specifically, these are the constraints that the system must work within.

### 3.3 Software Requirement specification

#### 3.3.1 Functional Requirements

##### 3.3.1.1 Entry Module

**Table 3.1 Entry**

|                |   |
|----------------|---|
| Use case Name  | Entry(check mask ,enter id, detect, train)                      |
| Priority       | Essential   |
| Trigger        | Entry   |
| Pre condition  | Person should be infront of camera                              |
| Basic path     | 1. Detect face of a person<br>2. Detect age, gender of a person |
| Alternate Path | No  |
| Post condition | These data are stored in dataset                                |
| Exceptions     | It will give a pop message like face is not detected.           |

##### 3.3.1.2 Exit Module

**Table 3.2 Exit**

|                |   |
|----------------|---|
| Use case Name  | Exit ( Recognition of face and count gets decreased)    |
| Priority       | Essential   |
| Trigger        | Entry   |
| Pre condition  | Person will be at the exit side                         |
| Basic path     | 1.person count gets decreased                           |
| Alternate Path | No  |
| Post condition | The data verifies the person image                      |
| Exceptions     | If the person face gets matched the count gets reduced. |

## **3.3.2 Non-Functional Requirements**

### **3.3.2.1 Computer Requirements**

#### **System Requirements**

1. System with web camera and good internet connection.
2. Minimum 2 systems needed (one at the entry and other at the exit)
3. A system with minimum 4GB RAM

#### **Software Requirements**

**OS : WINDOWS 10**

**IDE : VISUAL STUDIO CODE , PYTHON IDLE 3.8.7**

**BROWSER : CHROME Version 88.0.4324.150 or lower (64 or 32-bit)**

## **CHAPTER-4**

### **SYSTEM DESIGN**

#### **4.1 INTRODUCTION**

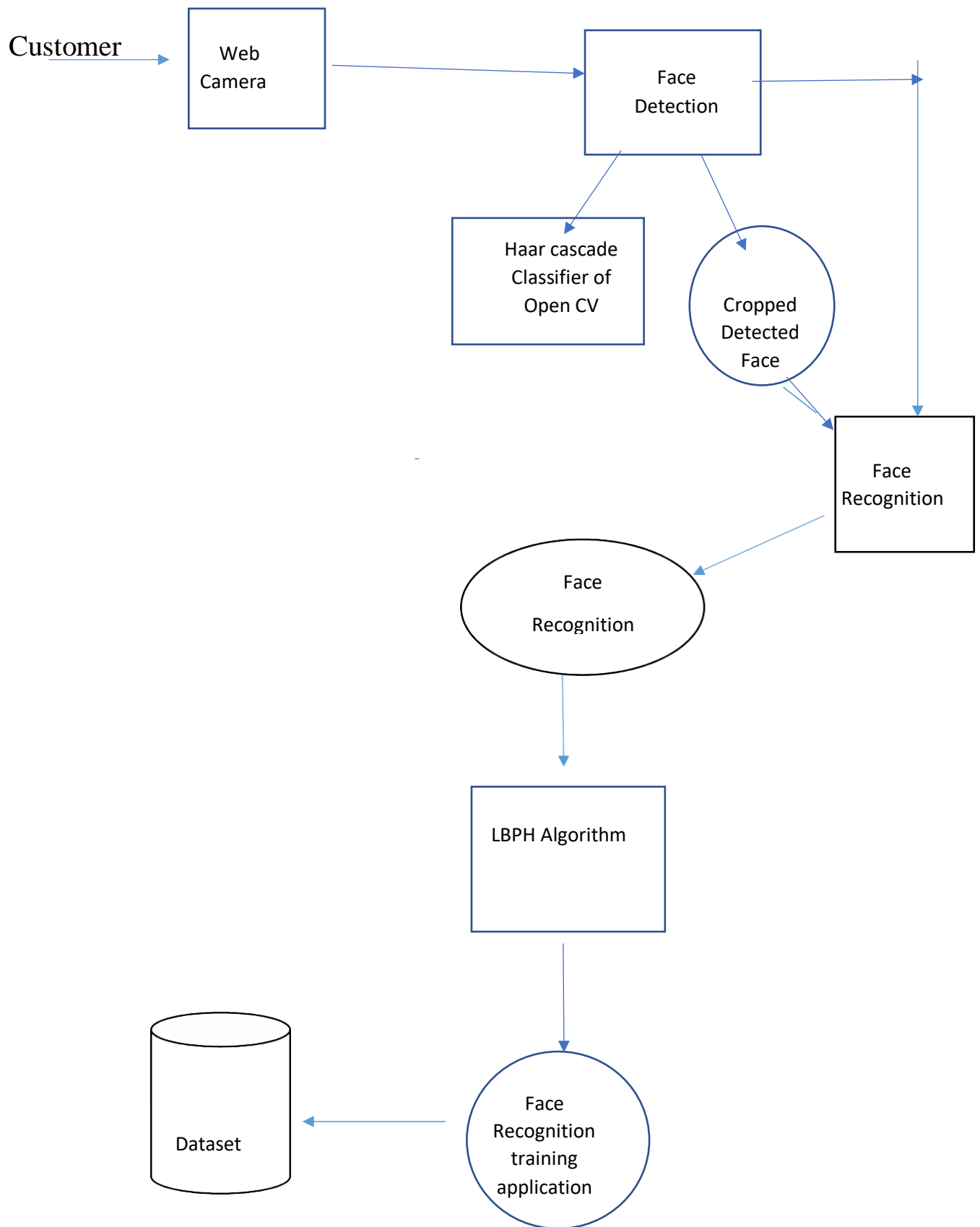
Machine Learning is process of making the computers to think like a human. Machine learning is the computer program for optimizing performance using historical data. For make the computer to think like a human being we have to train the computer. For training there is a lot of algorithm available. Here we take two algorithm namely LBPH and Haar cascade algorithm. Haarcascade algorithm is also known as Viola-Jones Algorithm. This algorithm helps us to detect and recognize face and also for checking mask on the face.

Checking whether the people wear mask or not and whether the person are maintaining social distancing or not is the major problem in this pandemic. In fact shops are the only place where we can see more crowd. So, here we are trying to check whether the people enter inside the shops are wear mask or not and also we are going to check social distancing between them.

Here , we use Haar cascade algorithm and LBPH algorithm for face detection and face recognition . LBPH full form is Local Binary Pattern Histogram. Haarcascade algorithm use haar features to detect faces. It will check for features like eyes, nose, mouth etc., to confirm itself as a face. After, detecting the face we will going to check the mask on the face. For mask check we use XML file to check whether the person wear mask or not. The system will check for nose and mouth. If it detect both the feature it will tell them to wear mask through alert message. If it doesn't mean it will conclude it as the person wear mask.

#### **4.2 SYSTEM ARCHITECTURE**

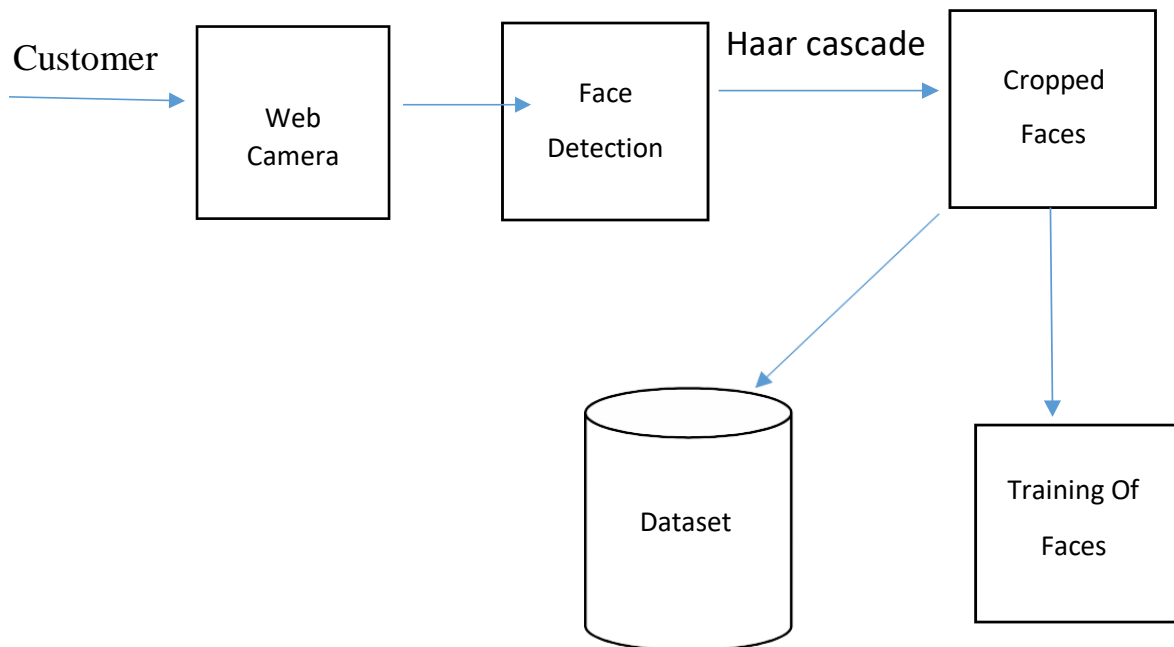
This system architecture describes how the proposed system will work. Here we first detect the face using web camera and then we check for mask in the face.



**Fig 4.1 System Architecture**

Further, we do face recognition by using the images that is stored in dataset during face detection.

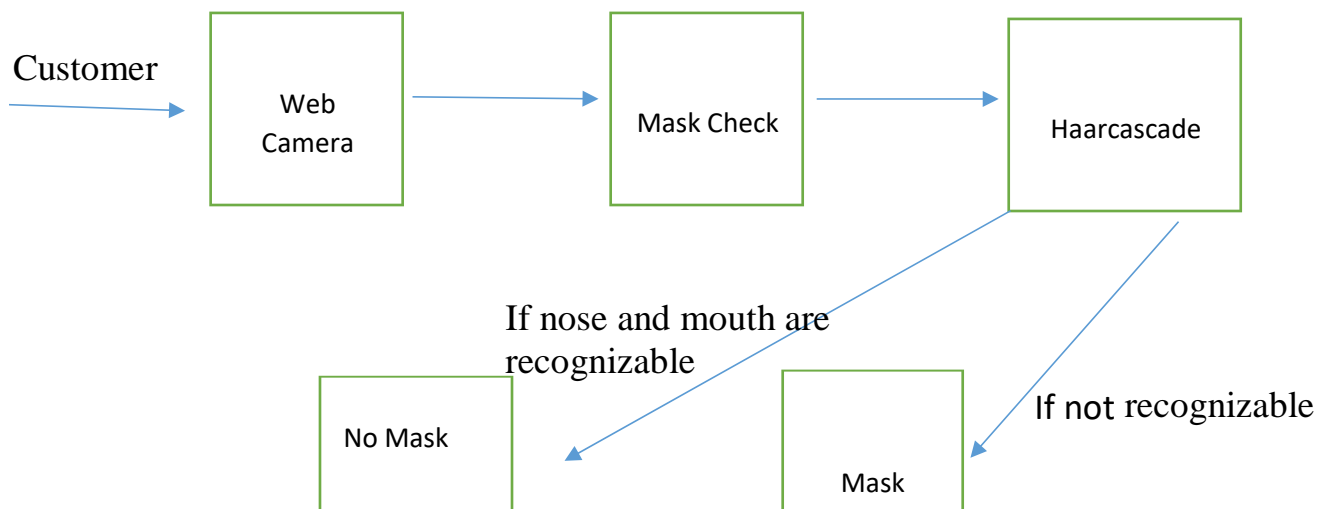
### 4.2.1 FACE DETECTION



**Fig 4.2 Face Detection**

Here, we detect the faces using Haarcascade algorithm. Firstly, web camera will detect the faces and then 50 sample images will get stored in the dataset. We can crop the detected face as needed. This face will further get trained by the system.

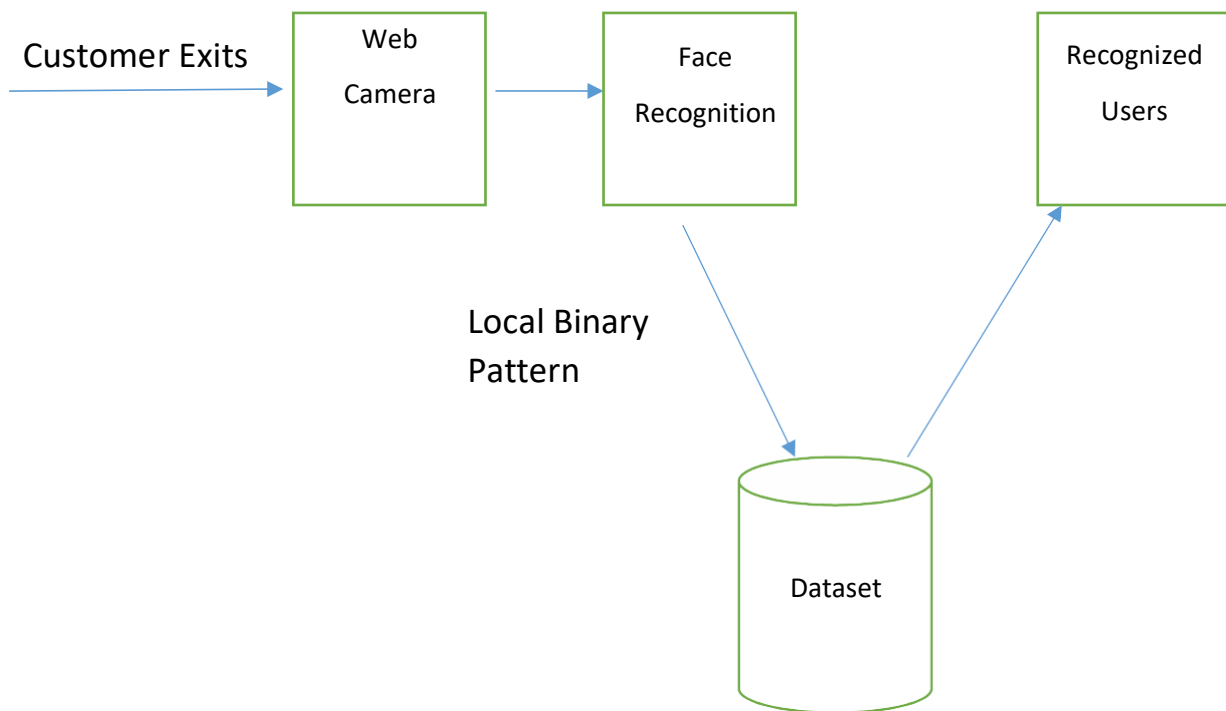
### 4.2.2 Mask Check



**Fig 4.3 Mask Check**

Here, we are going to check whether the person enter inside the shops wear mask or not. For checking process we are going to detect person's nose and mouth using the Haarcascade XML file. If the system able to detect nose and mouth, it mean the person doesn't wear mask So, it will give some alert message to wear mask. If the system does not detect nose and mouth means the person wear mask. So, now the system says the person wore mask.

### 4.2.3 FACE RECOGNITION



**Fig 4.4 Face Recognition**

Here, Face recognition is done at exit side of the shop. The web camera in exit side will recognize the face using LBPH (Local Binary Pattern Histogram) algorithm. It will check for the face detected in exit camera is available on the dataset, where we already stored the sample images of the detected face through entry camera.

#### **4.2.4 Distance Check**

The user is required to input the focal length and the sensor dimensions of the camera to be used.

The user is then required to position these 2 individuals at the minimum social distance that is to be maintained , henceforth referred to as the reference social distance.

The advantage of this is that authorities can actually select the social distance they want maintained , according to the specific guidelines that they wish to follow . For example: WHO guidelines mention the recommended amount of social distance to be a minimum of 3 ft , while CDC guidelines recommend the minimum to be 6 ft . The social distance between 2 people is solely judged relative to the initial calibration and the absolute distance need not be provided to the model .

This social distancing model uses the principle that a camera lens is essentially a convex lens , where the image is essentially captured on the screen . For this model , as fore mentioned , we need the focal length and sensor dimensions.

The focal length of a lens is the distance from the optic center of the lens to its focus. In optics and photography , the focal length is measured in millimeters(mm) . Longer the focal length , higher the magnification , but lower the angle of view.

An image sensor is a device found in the hardware of a camera , that uses light to detect information to convert a view into an image . The sensor essentially functions as screen where all the pixels in an image are mapped .Greater the pixels mapped , greater the image quality.

## **CHAPTER-5**

### **IMPLEMENTATION**

The proposed methodology is implemented in python and packages includes

- Visual studio
- OpenCV
- TensorFlow

The implementation is segregated into modules according to the process of face mask and social distancing detection

- Face detection and recognition module
- Face mask detection module
- Social distance detection module

## **5.1 FACE DETECTION AND RECOGNITION MODUL**

### **5.1.1 FACE DETECTION MODULE**

First we convert the frame from color to grayscale . To detect the faces we used a Haar Cascade classifier where a cascade function is trained and detect features in other images . For this we use haar features like edge , line and four rectangle .

For a large image or variable size of an image , it takes a lot of computation and features and most of them will be irrelevant.The Region Of Interest (ROI) i.e. containing faces is extracted and sent to next stage.

### **5.1.2 FACE RECOGNITION MODULE**

For face recognition , we decided to use the LBPH algorithm because of its robustness , the capability to recognize both front and side faces and better compared to Eigenfaces and Fisherfaces .The LBPH algorithm is used as they find

characteristics that best describe a face in an image. There were many face recognition algorithms and the LBPH algorithm is better. This method is easier , within the sense it characterizes the image within the dataset locally and when a replacement unknown image occurs we perform an equivalent algorithm and compare the results to each of the pictures within the dataset . It works better in different environments and light conditions than other algorithms.

Local Binary Pattern (LBP) operation creates an image which highlights the characteristics of a image in a better way. It uses the concept of sliding window and the parameters , radius and neighbours.

First , we convert the frame into matrices of 3x3 pixels . If a neighbour pixels in a matrix is greater than the median pixel of that matrix set value is 1 else 0 in that pixel position. Now note down the values of neighbour pixels in a line we get a binary number. Convert that binary number to decimal number and replace it with the median pixel value of the matrix.

As the image is now converted into LBP form , we extract histograms from each grid and concatenate to form a new and larger histogram . The concatenated histogram indicates the characteristics of the original image . Each histogram represents the facial image . Each histogram represents the facial image from the database . For the new image , it performs the above steps and gets a new histogram for the image.

### **5.1.3 POST PROCESSING**

Now to recognize the person in the image it compares (by applying Euclidian distance) the new histogram with the histograms from the training dataset and choose the histogram having lowest confidence i.e. least distance , as lower confidence are better and also extract the ID corresponding to that histogram. If confidence is less than 50 then details belong to the extracted ID is on the frame . The names are

updated in the excel sheet to avoid duplicate names . This helps to identify the intruders in the shop.

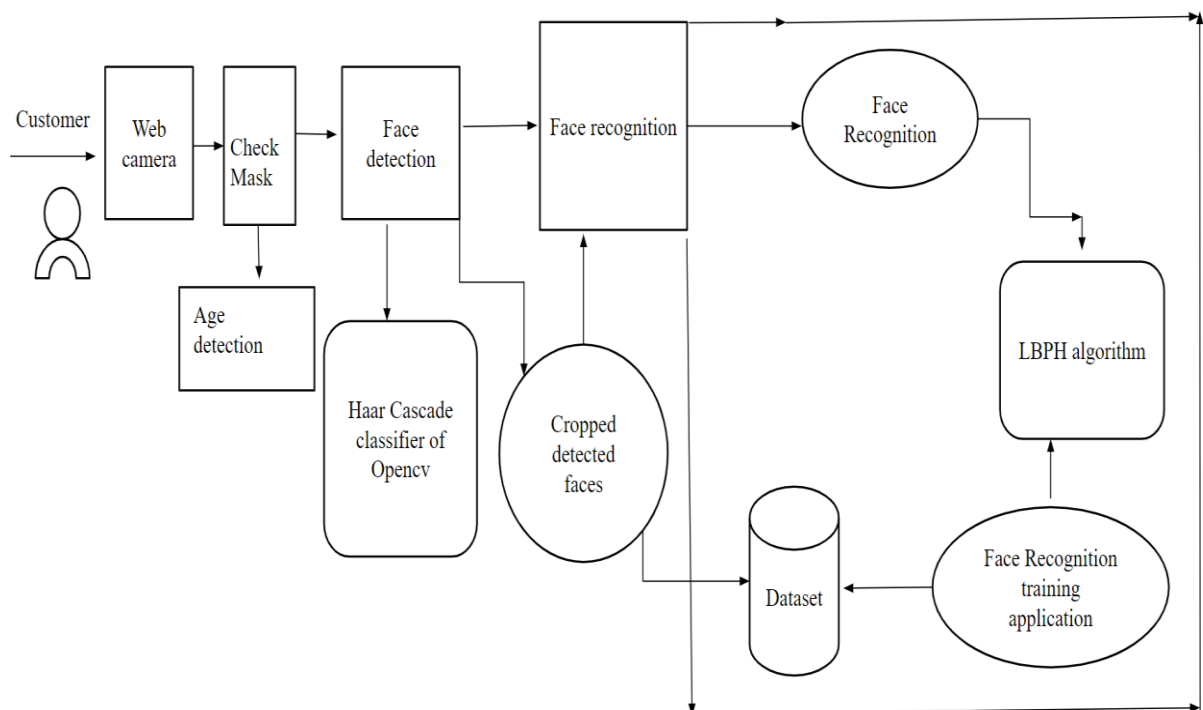
## 5.2 FACE MASK DETECTION MODULE

### 5.2.1 TWO -PHASE COVID-19 FACE MASK DETECTOR

In order to train a custom face mask detector , we need to break our project into two distinct phases , each with its own respective sub-steps:

**1.Training:** Here we'll focus on , loading our face mask detection dataset from disk , training a model (using tensor flow) on this dataset , and then serializing the face mask detector to disk.

**2. Deployment:** Once the face mask detector is trained , we can then move on to loading the mask detector , performing face detection and then classifying each face as with\_mask or without\_mask.



**Fig 5.1 Complete Architecture**

## 5.2.2 HOW WAS OUR FACE MASK DATASET CREATED?

1. Taking normal images of faces
2. Then creating a custom computer Python script to add face masks them.

Facial landmarks allow us to automatically infer the location of facial structures, including :

- Eyes
- Eyebrows
- Nose
- Mouth
- Jawline

To use facial landmarks to build a dataset of faces wearing face masks , we need to first start with an image of a person not wearing a face mask

From there , we apply face detection to computing the bounding box location of the face in the image.

Once we know where in the image the face is , we can extract the face Region Of Interest.

```
import cv2

import sys

import time

face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

nose_cascade = cv2.CascadeClassifier('CascadeFiles_haarcascade_mcs_nose.xml')

mouth_cascade = cv2.CascadeClassifier('Mouth.xml')

video_capture = cv2.VideoCapture(0)
```

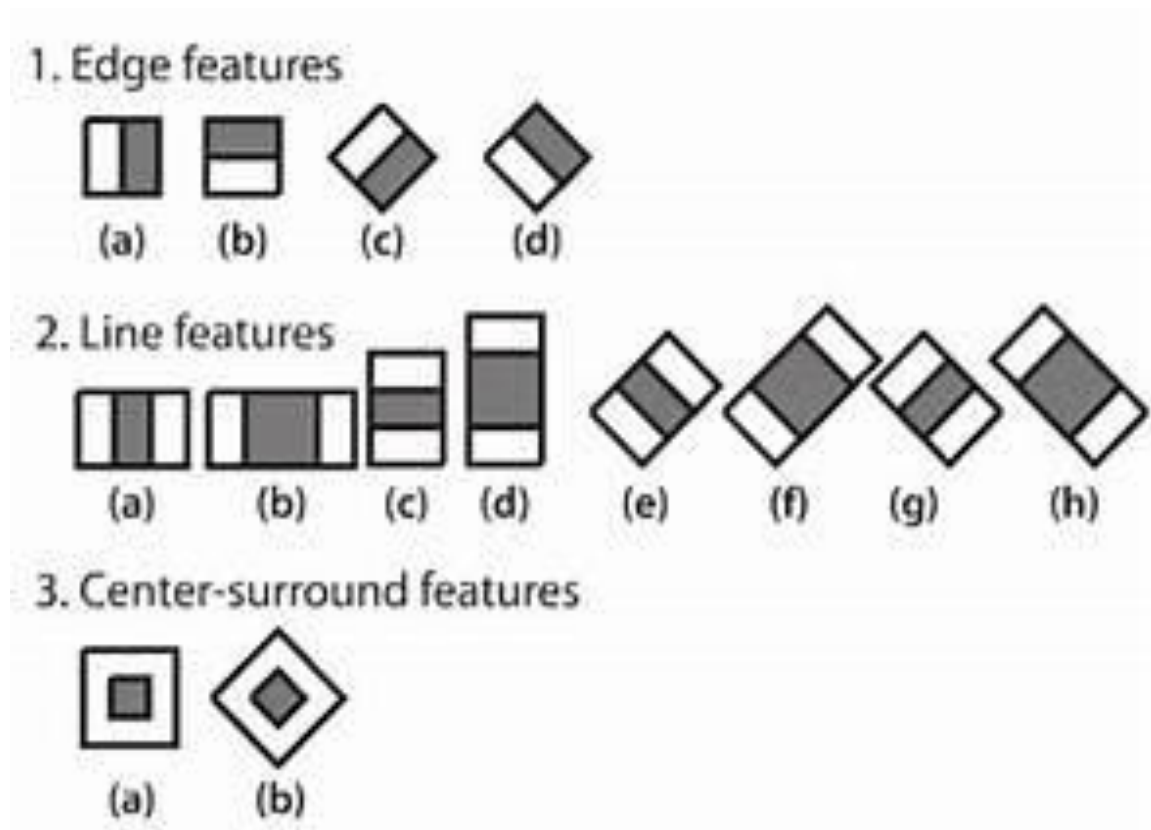
```
def time_reset(timing):

    current_time = time.time()

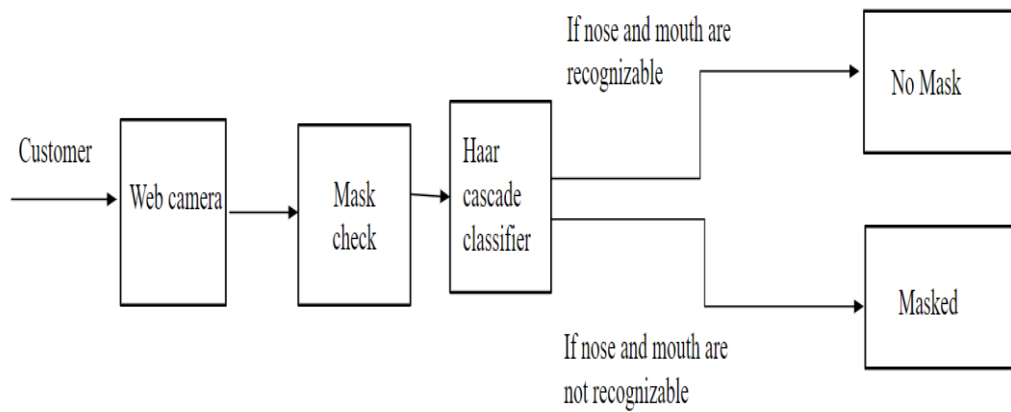
    timing = current_time - timing

    return timing
```

And from there , we apply facial landmarks , allowing us to localize the eyes , nose, mouth , etc..



**Fig 5.2 Haar Features**



**Fig 5.3 Mask Check**

## 5.3 TRACKING SOCIAL DISTANCING

The user is required to input the focal length and the sensor dimensions of the camera to be used.

The user is then required to position these 2 individuals at the minimum social distance that is to be maintained , henceforth referred to as the reference social distance.

The advantage of this is that authorities can actually select the social distance they want maintained , according to the specific guidelines that they wish to follow For example: WHO guidelines mention the recommended amount of social distance to be a minimum of 3 ft , while CDC guidelines recommend the minimum to be 6 ft The social distance between 2 people is solely judged relative to the initial calibration and the absolute distance need not be provided to the model .

This social distancing model uses the principle that a camera lens is essentially a convex lens , where the image is essentially captured on the screen . For this model , as fore mentioned , we need the focal length and sensor dimensions.

The focal length of a lens is the distance from the optic center of the lens to its focus. In optics and photography , the focal length is measured in millimeters(mm) . Longer the focal length , higher the magnification , but lower the angle of view.

An image sensor is a device found in the hardware of a camera , that uses light to detect information to convert a view into an image . The sensor essentially functions as screen where all the pixels in an image are mapped .Greater the pixels mapped , greater the image quality.

Let field width , i.e. The observed width of a real life object be  $w$  and measured distance of an object from the camera be  $d$ .

Let the people between whom the distance is being measured be: Person 1 and Person 2 , that stand at positions  $(x_1 , y_1)$  and  $(x_2 , y_2)$  in a given image . These coordinates are detected at the feet of the individuals .

$$\text{Sensor dimension/ focal length} = \text{field dimension} / \text{distance to field}$$

To find the depth of an object in a photograph , the following formula can be obtained from eqn2 :

$$d = \text{actual ht of object(mm)} \times \text{focal length (mm)} / \text{ht of object on sensor(mm)}$$

where  $ht$  is height . Since we only the height of the object in pixels( $px$ ) in the image , we can use the following formula to obtain the height of the object in the image on the sensor in millimeters . Here 's the height of an actual human is assumed to be 1.6m in this model , since the average height of a human is estimated to be that match.

$$\text{Object hit on sensor(mm)} = \text{object ht in image(Px.)} \times \text{pixel size}$$

where  $px$  is measurement in number of pixels . Hence , the depth of a person would be equal to the distance a person stands from the camera . This distance from the camera for the 2 people will be represented as  $d_1$  and  $d_2$  . To measure the appropriate distance between these two people in the image, the difference of their  $x$ -coordinates are taken to be the social distance width.

$$\text{Social distancing width(mm)} = (|x_1 - x_2|) \times \text{pixel size}$$

$W = \text{sensor width} \times \text{social distancing width (mm)} / \text{focal length(mm)}$

If person 1 is assumed to be at (0 , d1) and person 2 at (w , d2)

$\text{Social distance} = [(w - 0)^2 + (d2 - d1)^2]^2$

To obtain the pixel size , the following formula must be used :

$$\text{Pixel size} = \frac{\text{sensor width(mm)}}{\text{width of image(px)}} + \frac{\text{sensor height(mm)}}{\text{height of image (px) / 2}}$$

The social distance , which is first calculated in calibration mode , will be used as the reference social distance .In testing mode , the social distance between 2 individuals will be calculated using the equations. If the calculated social distance is lesser than the reference social distance , the pair of individuals will be identified as violators.

```
import numpy as np
```

```
import time
```

```
import cv2
```

```
import math
```

```
import imutils
```

```
labelsPath = "./coco.names"
```

```
LABELS = open(labelsPath).read().strip().split("\n")
```

```
np.random.seed(42)
```

```
COLORS = np.random.randint(0, 255, size=(len(LABELS), 3), dtype="uint8")
```

```
weightsPath = "./yolov3.weights"
```

```
configPath = "./yolov3.cfg"
```

```

print("Loading Machine Learning Model ...")
net = cv2.dnn.readNetFromDarknet(configPath, weightsPath)

print("Starting Camera ...")
cap = cv2.VideoCapture(0)

while(cap.isOpened()):

    ret, image = cap.read()

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

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

    ln = net.getLayerNames()

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

    blob = cv2.dnn.blobFromImage(image, 1 / 255.0, (416, 416), swapRB=True,
crop=False)

    net.setInput(blob)

    start = time.time()

    layerOutputs = net.forward(ln)

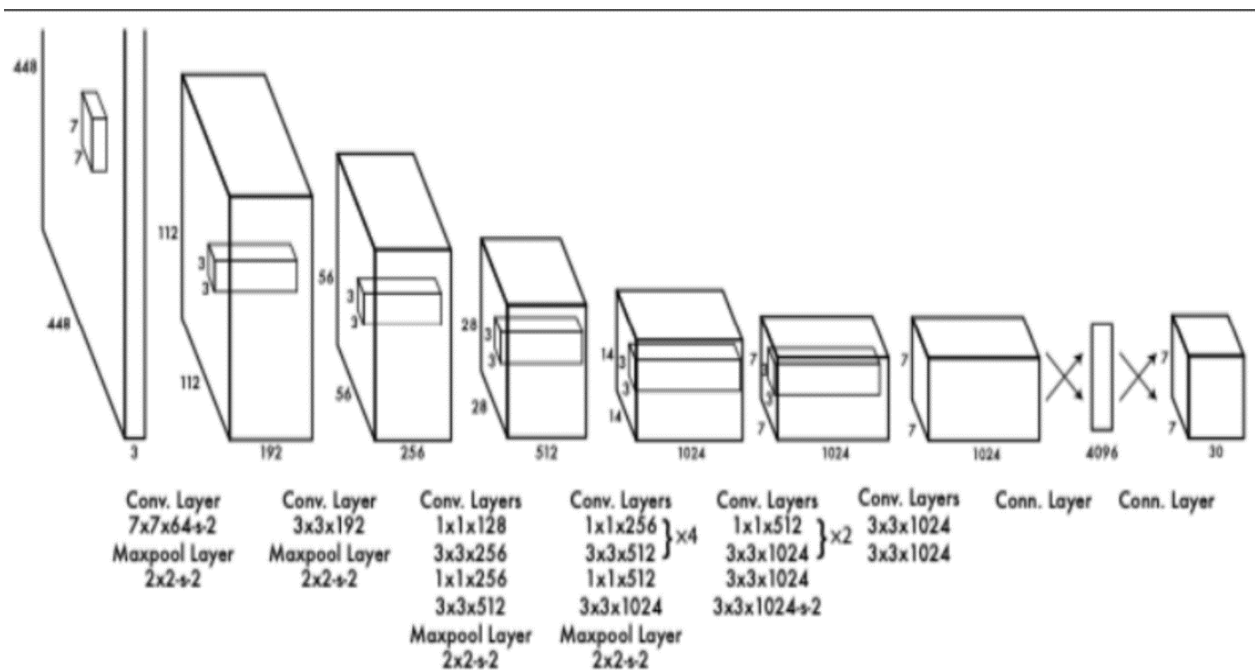
    end = time.time()

```

### 5.3.1 YOLOV3

YOLOV3 is the latest variant of a popular object detection algorithm YOLO You Only Look Once . The published model recognizes 80 different objects in images and videos , but most importantly it is super fast and nearly as accurate as single shot Multibox(SSD) .First , it divides the image into 13 x 13 grid of cells The size of these 169 cells varies depending on the size of the input . For a 416 x

416 input size that we used in our experiments , the cell size was 32 x 32 boxes in the image. For each bounding box , the network also predicts the confidence that the bounding box , the network also encloses an object and the probability of the enclosed object being a particular class . Most of these bounding boxes are eliminated because their confidence is low or because they are enclosing the same object as another bounding box with very high confidence score. This technique is called non-maximum suppression.



**Fig 5.4 Neural net Architecture (yolo)**

## EASY INTEGRATION WITH AN OPENCV APPLICATION

If your application already uses OpenCV and you simply want to use YOLOv3 you don't have to worry about compiling and building the extra Dark net code. OPENCV CPU VERSION IS 9X FASTER OpenCV's CPU implementation of the DNN module is astonishingly fast. For example , Dark net when used with OpenMP takes about 2 seconds on a CPU for inference on a single image. In contrast OpenCV's implementation runs in a mere 0.22 seconds.

As the input video may be taken from an arbitrary perspective view , the first step is to transform perspective of view to a bird's-eye [top-down] view. As the input frames are monocular [taken from a single camera] , the simplest transformation method involves selecting four points in the perspective view which define ROI where we want to monitor social distancing and mapping them to the corners of a rectangle in the bird's-eye view. Also these points should form parallel lines in real world if seen from bird's eye view. This top view has the property that points are distributed uniformly horizontally and vertically .

## **DETECTION**

The second step to detect pedestrians and draw a bounding box around each pedestrians . To clen up the output bounding boxes , we apply minimal post-processing such as non-max suppression[NMS] and various rule-based heuristics , so as to minimize the risk of over fitting.

## **DISTANCE CALCULATION**

Now we have bounding box for each person in the frame . We need to estimate person location in frame .i.e. we can take bottom center point of bounding box as person location in frame. Then we estimate [x,y] location in bird's eye view by applying transformation to the bottom center point of each person's bounding box , resulting in their position in the bird's eye view. Last step is to compute the bird's eye view distance between every pair of people and scale the distances by the scaling factor in horizontal and vertical direction estimated from calibration.

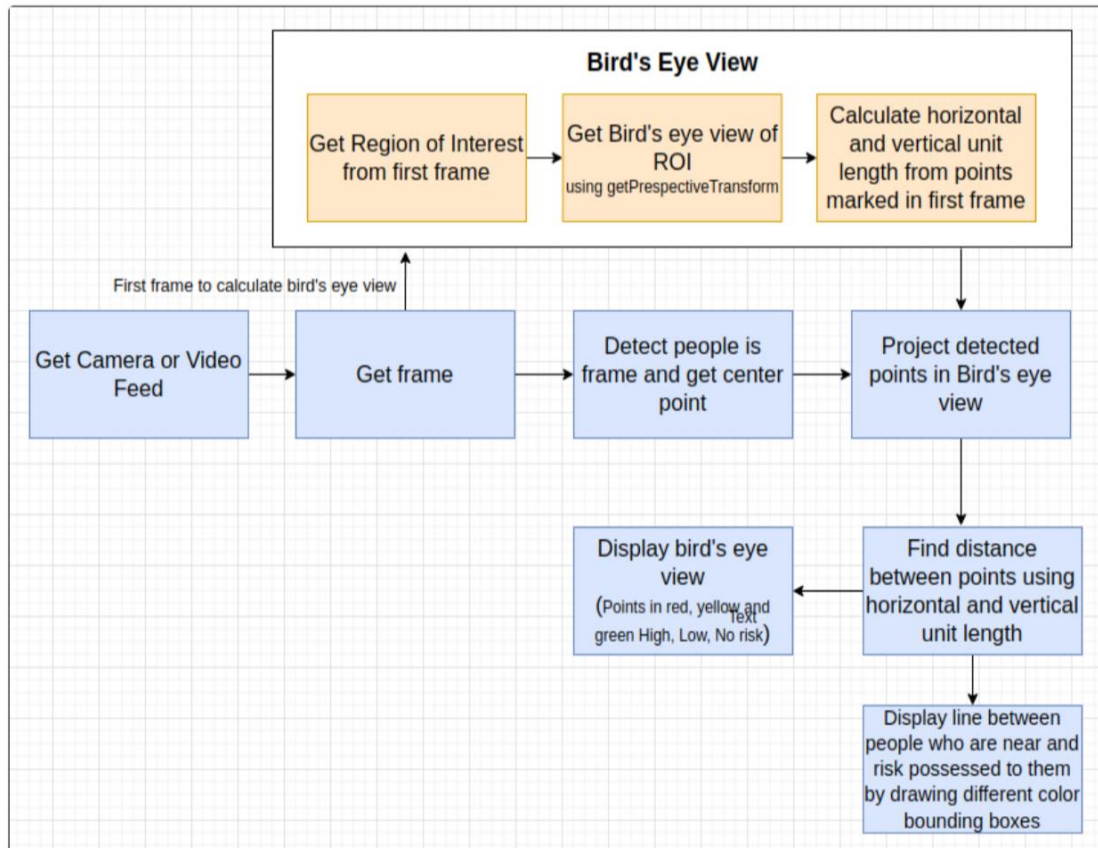
## WORKING

Running the program will give you frame[first frame] where you need to draw ROI and distance scale. To get ROI and distance scale points from first frame code to transform perspective to Bird's eye view [top view] and to calculate horizontal and vertical 180cm distance in Bird's eye view.

ROI and scale points selection for first frame. The second step to detect pedestrians and draw a bounding box around each pedestrians . To detection humans in video and get bounding box details.

Now we have bounding box for each person in the frame. We need to estimate person location in frame .i.e. we can take bottom center point of bounding box as person location in frame. Then we estimate [x ,y] location in bird's eye view by applying transformation to the bottom center point of each person's bounding box,resulting in their position in the bird's eye view. Last step is to compute the bird's eye view distance between every pair of pair of people and vertical direction estimated from calibration.

Lastly we can draw bird's eye view for region of interest [ROI] and draw bounding boxes according to risk factor for humans in a frame and draw lines between boxes according to risk factor between two humans . Red ,yellow , green points represents risk to humans in bird's eye view.



**Fig 5.5 Birds Eye View**

## **CHAPTER-6**

### **TESTING**

#### **6.1 TESTING PROCESS**

The purpose of testing is to discover errors. Testing is the process of trying to discover conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies , assemblies or finished product . It is the process of exercising software with the intend of ensuring that the software system meet its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test.

Each test type addresses a specific testing requirement . Testing is not isolated to only one phase of the project but should be exercised in all phases of the project. After developing each unit of the software product , the developers go through an exercise testing process of the software. After the development of the software modules , developers perform a through unit testing of each software components they also perform integration testing of all combine modules.

#### **6.2 TESTING OBJECTIVES**

There are several rules that can serve as testing objectives, they are

1. Testing is a process of executing a program with the intent of finding anerror.
2. A good test case is one that has high probability of finding an undiscovered error.
3. A successful test is one that uncovers an undiscovered error.

If testing is conducted successfully according to the objectives as stated above it would uncover errors in the software.

There are three ways to test a program:

1. Correctness
2. Implementation efficiency
3. Computational complexity

## **6.3 UNIT TESTING**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program input produces valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration. This is a structural testing that relies on knowledge of its construction and is invasive.

Unit tests perform basic tests at component level and test a specific business process, application, and system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

## **6.4 INTEGRATION TESTING**

Integration tests are design to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfied, as shown by successful unit testing, the combination of components is correct and consistent. Integration testing specifically aimed at exposing the problems that arise from the combination of components.

## **6.5 TEST RESULTS**

All the test cases mentioned below are verified successfully. No defect encountered.

## 6.6 TEST CASES

**Table 6.1 Unit testing**

| S.no | TESTCASE        | INPUT | EXPECTED OUTPUT              | ACTUAL OUTPUT                 | RESULT |
|------|-----------------|-------|------------------------------|-------------------------------|--------|
| 1.   | Trained dataset | Image | Trained datasets with object | Trained datasets with objects | PASS   |

## BATCH NORMALIZATION

**Table 6.2 unit testing**

| S.NO | TESTCASE  | INPUT           | EXPECTED OUTPUT             | ACTUAL OUTPUT               | RESULT |
|------|---|-----------------|-----------------------------|-----------------------------|--------|
| 1.   | Detecting face mask and social distancing detection | Image and video | DETECTING IMAGES AND VIDEOS | DETECTING IMAGES AND VIDEOS | PASS   |

## **CHAPTER-7**

### **CONCLUSION**

We proposed facial mask and social distancing detection using machine learning technique. We demonstrated that our models are able to generate image objects and videos for those detections. We showed that the how that the face has been detected using various algorithms and for distancing we are using real time videos to implement them. By this the face mask and detecting the social distance is to be done by the real time implementation.

## **CHAPTER-8**

### **FUTURE ENHANCEMENT**

The project made is to ensure that the project could be valid in today's challenging real world. It has a vast scope in future. More functionality can be added in accordance with the flexibility of the user requirement and specification. In future using IoT we have planned to implement that people who all are not maintaining the social distance are warned with an alert message.

## **APPENDICES**

### **APPENDIX-I**

#### **SAMPLE SOURCE CODE**

##### **TKINTER MAINPAGE CODE**

```
import os

import sys

from tkinter import *

from tkinter import messagebox

from tkinter import colorchooser

def f2():

    global frame2

    frame1.destroy()

    frame2 = Frame(window,height = 500, width = 700,bg = '#D3D3D3')

    frame2.place(x= 0, y = 0)

    def f221():

        os.system('tkinter_entry.py')

    def f222():

        os.system('tkinter_exit.py')

    def f1():

        global frame1

        try:

            frame2.destroy()
```

```

except:

    pass

frame1 = Frame(window,height = 500, width = 700,bg = '#00B2EE')

frame1.place(x= 0, y = 0)

l0 = Label(frame1,text = 'SHOPS MALL',font = ('normal',25,'bold'),bg =
'#D3D3D3')

l0.place(x = 230 , y = 90)

b1 = Button(frame1,text = 'ENTRY',font = ('normal', 15, 'italic'),bg =
'#00FFFF',command = f221)

b1.place(x = 250, y = 270)

b2 = Button(frame1,text = 'EXIT',font = ('normal', 15, 'italic'),bg =
'#00FFFF',command = f222)

b2.place(x = 370, y = 270)

window.configure(bg='#8A2BE2')

window = Tk()

window.title('HOME')

window.geometry('700x500')

f1()

window.mainloop()

```

## **TKINTER ENTRY PAGE CODE:**

```

import os

import sys

from tkinter import *

```

```
from tkinter import messagebox
```

```
def f2():
```

```
    global frame2
```

```
    frame1.destroy()
```

```
    frame2 = Frame(window,height = 500, width = 700,bg = '#D3D3D3')
```

```
    frame2.place(x= 0, y = 0)
```

```
def f222():
```

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

```
def f333():
```

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

```
def f444():
```

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

```
def f1():
```

```
    global t1,t2,frame1,e1,e2
```

```

try:

    frame2.destroy()

except:

    pass


frame1 = Frame(window,height = 500, width = 700,bg = '#00B2EE')

frame1.place(x= 0, y = 0)

l0      =      Label(frame1,text      =      'SHOPPIE      MALL',font      =
('normal',25,'bold'),bg='#FFF0F5')

l0.place(x = 230 , y = 90)

b1 = Button(frame1,text = 'DETECT FACE',font = ('normal', 15, 'italic'),bg =
'#00FFFF',command = f222)

b1.place(x = 50, y = 270)

b2 = Button(frame1,text = 'CHECK MASK',font = ('normal', 15, 'italic'),bg =
'#00FFFF',command = f333)

b2.place(x = 230, y = 270)

b3=Button(frame1,text = 'CHECK SOCIAL DISTANCE',font = ('normal',15,
'italic'),bg = '#00FFFF',command=f444)

b3.place(x = 390, y = 270)


window = Tk()

window.title('ENTRY PAGE')

window.geometry('700x500')

```

```
f1()
```

```
window.mainloop()
```

### **TKINTER EXIT PAGE CODE:**

```
import os
```

```
import sys
```

```
from tkinter import *
```

```
from tkinter import messagebox
```

```
def f2():
```

```
    global frame2
```

```
    frame1.destroy()
```

```
    frame2 = Frame(window,height = 500, width = 700,bg = '#D3D3D3')
```

```
    frame2.place(x= 0, y = 0)
```

```
def f21():
```

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

```
def f22():
```

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

```
def f1():
```

```
    global frame1
```

```

try:

    frame2.destroy()

except:

    pass

frame1 = Frame(window,height = 500, width = 700,bg = '#00B2EE')

frame1.place(x= 0, y = 0)

l0 = Label(frame1,text = 'SHOPS MALL',font = ('normal',25,'bold'),bg =
'#F5F5DC')

l0.place(x = 230 , y = 90)

b1 = Button(frame1,text = 'TRAIN IMAGES',font = ('normal', 15, 'italic'),bg =
'#00FFFF',command = f21)

b1.place(x = 150, y = 270)

b2 = Button(frame1,text = 'RECOGNIZE FACE',font = ('normal', 15,
'italic'),bg = '#00FFFF',command = f22)

b2.place(x = 330, y = 270)

window = Tk()

window.title('EXIT PAGE')

window.geometry('700x500')

f1()

window.mainloop()

```

## **MASK CHECKING CODE:**

```
import cv2

import sys

import time

face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

nose_cascade = cv2.CascadeClassifier('CascadeFiles_haarcascade_mcs_nose.xml')

mouth_cascade = cv2.CascadeClassifier('Mouth.xml')

video_capture = cv2.VideoCapture(0)

def time_reset(timing):

    current_time = time.time()

    timing = current_time - timing

    return timing

def main():

    nose_start_time = time.time()

    mouth_start_time = time.time()

    nose_start_time = time_reset(nose_start_time)

    mouth_start_time = time_reset(mouth_start_time)

    while True:
```

```

ret, frame = video_capture.read()

gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

faces = face_cascade.detectMultiScale(gray, 1.1, 4)

noses = nose_cascade.detectMultiScale(gray, 1.3, 5)

mouths = mouth_cascade.detectMultiScale(gray, 1.7, 5)

nose = True

mouth = True

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

    for (x_nose, y_nose, w_nose, z_nose) in noses:

        if x <= x_nose and x + w >= x_nose and y <= y_nose and y + h >=
y_nose:

            nose = False

            nose_start_time = time_reset(nose_start_time)

    for (x_mouth, y_mouth, w_mouth, z_mouth) in mouths:

```

```
        if x <= x_mouth and x + w >= x_mouth and y <= y_mouth and y + h  
>= y_mouth:
```

```
            mouth = False
```

```
            mouth_start_time = time_reset(mouth_start_time)
```

```
        if mouth_start_time > 2 and nose_start_time > 2 and mouth and nose:
```

```
            cv2.putText(frame, "MASK", (x, y - 10),  
cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0))
```

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

```
        else:
```

```
            cv2.putText(frame, "NO MASK", (x, y - 10),  
cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255))
```

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

```
cv2.imshow('Video', frame)
```

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

```
        break

    video_capture.release()

    cv2.destroyAllWindows()

    return

main()
```

## **FACE DETECTION CODE:**

```
import cv2

import os


cam = cv2.VideoCapture(0)

cam.set(3, 640)

cam.set(4, 480)


face_detector = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')


face_id = input("\n enter user id")


print("\n [INFO] Initializing face capture....")

# Initialize individual sampling face count

count = 0
```

```

while(True):

    ret, img = cam.read()

    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    faces = face_detector.detectMultiScale(gray, 1.3, 5)

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

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

        count += 1

        cv2.imwrite("dataset/User." + str(face_id) + '.' + str(count) + ".jpg",
gray[y:y+h,x:x+w])

        cv2.imshow('image', img)

    k = cv2.waitKey(100) & 0xff

    if k == 27:

        break

    elif count >= 5:

```

```
break
```

```
print("\n [INFO] Exiting Program")
```

```
cam.release()
```

```
cv2.destroyAllWindows()
```

```
import cv2
```

```
import numpy as np
```

```
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
```

```
smile_cascade = cv2.CascadeClassifier('haarcascade_smile.xml')
```

```
def detect(gray, frame):
```

```
    faces = face_cascade.detectMultiScale(gray,1.3,5)
```

```
    if(True):
```

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

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

```
            cv2.putText(frame, ' Please Wear a Mask', (100, 120), font, 1, (255, 0, 0),  
2, cv2.LINE_4)
```

```
            roi_gray=gray[y:y+h,x:x+w]
```

```
            roi_color=frame[y:y+h,x:x+w]
```

```
            smiles = smile_cascade.detectMultiScale(roi_gray,1.8,20)
```

```
            for (sx,sy,sw,sh) in smiles:
```

```

        cv2.rectangle(roi_color,(sx, sy),((sx+sw),(sy+sh)),(255,0,0),2)

    else:

        call()

    return frame

def call():

    while(True):

        ret,frame=cap.read()

        cv2.putText(frame, 'Please , follow social distancing', (50, 50), font, 1, (0,
0, 255), 2, cv2.LINE_4)

        gray=cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)

        canvas=detect(gray,frame)

        cv2.imshow('frame',canvas)

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

            break

cap=cv2.VideoCapture(0)

font = cv2.FONT_HERSHEY_SIMPLEX

call()

cap.release()

cv2.waitKey(0)

cv2.destroyAllWindows()

```

## FACE RECOGNITION CODE:

```
import cv2

import numpy as np

import os


recognizer = cv2.face.LBPHFaceRecognizer_create()

recognizer.read('trainer/trainer.yml')

cascadePath = "haarcascade_frontalface_default.xml"

faceCascade = cv2.CascadeClassifier(cascadePath);

font = cv2.FONT_HERSHEY_TRIPLEX

id = 0

names = [0, 1, 2, 3, 'Z', 'W']


cam = cv2.VideoCapture(0)

cam.set(3, 640)

cam.set(4, 480)


minW = 0.1*cam.get(3)

minH = 0.1*cam.get(4)
```

```

while True:

    ret, img =cam.read()

    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    faces = faceCascade.detectMultiScale(

        gray,

        scaleFactor = 1.2,

        minNeighbors = 5,

        minSize = (int(minW), int(minH)),

    )

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

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

        id, confidence = recognizer.predict(gray[y:y+h,x:x+w])

        if (confidence < 100):

            id = names[id]

            confidence = " {0}%".format(round(100 - confidence))

        else:

            id = "unknown"

```

```

confidence = " {0}%".format(round(100 - confidence))

cv2.putText(img, str(id), (x+5,y-5), font, 1, (255,255,255), 2)

cv2.putText(img, str(confidence), (x+5,y+h-5), font, 1, (255,255,0), 1)


cv2.imshow('camera',img)


k = cv2.waitKey(10) & 0xff

if k == 27:

    break

print("\n [INFO] Exiting Program")

cam.release()

cv2.destroyAllWindows()

```

## **TRAINING THE FACE CODE:**

```

import cv2

import numpy as np

from PIL import Image

import os


path = 'dataset'

```

```

recognizer = cv2.face.LBPHFaceRecognizer_create()

detector = cv2.CascadeClassifier("haarcascade_frontalface_default.xml");

def getImagesAndLabels(path):

    imagePath = [os.path.join(path,f) for f in os.listdir(path)]

    faceSamples=[]

    ids = []

    for imagePath in imagePath:

        PIL_img = Image.open(imagePath).convert('L')

        img_numpy = np.array(PIL_img,'uint8')

        id = int(os.path.split(imagePath)[-1].split(".")[1])

        faces = detector.detectMultiScale(img_numpy)

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

            faceSamples.append(img_numpy[y:y+h,x:x+w])

            ids.append(id)

```

```

return faceSamples,ids

print ("\n [INFO] Training faces. It will take a few seconds. Wait ...")

faces,ids = getImagesAndLabels(path)

recognizer.train(faces, np.array(ids))

recognizer.write('trainer/trainer.yml')

print("\n [INFO] {0} faces trained.".format(len(np.unique(ids))))

```

## **SOCIAL DISTANCE CODE:**

```

import numpy as np

import time

import cv2

import math

import imutils

labelsPath = "./coco.names"

LABELS = open(labelsPath).read().strip().split("\n")

np.random.seed(42)

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

weightsPath = "./yolov3.weights"

configPath = "./yolov3.cfg"

```

```

print("Loading Machine Learning Model ...")

net = cv2.dnn.readNetFromDarknet(configPath, weightsPath)

print("Starting Camera ...")

cap = cv2.VideoCapture(0)

while(cap.isOpened()):

    ret, image = cap.read()

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

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

    ln = net.getLayerNames()

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

    blob = cv2.dnn.blobFromImage(image, 1 / 255.0, (416, 416), swapRB=True,
crop=False)

    net.setInput(blob)

    start = time.time()

    layerOutputs = net.forward(ln)

    end = time.time()

    print("Prediction time/frame : {:.6f} seconds".format(end - start))

    boxes = []

    confidences = []

```

```

classIDs = []

for output in layerOutputs:

    for detection in output:

        scores = detection[5:]

        classID = np.argmax(scores)

        confidence = scores[classID]

        if confidence > 0.1 and classID == 0:

            box = detection[0:4] * np.array([W, H, W, H])

            (centerX, centerY, width, height) = box.astype("int")

            x = int(centerX - (width / 2))

            y = int(centerY - (height / 2))

            boxes.append([x, y, int(width), int(height)])

            confidences.append(float(confidence))

            classIDs.append(classID)


idxs = cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.3)

ind = []

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

    if(classIDs[i] == 0):

        ind.append(i)

a = []

b = []

```

```

if len(idxs) > 0:

    for i in idxs.flatten():

        (x, y) = (boxes[i][0], boxes[i][1])

        (w, h) = (boxes[i][2], boxes[i][3])

        a.append(x)

        b.append(y)


distance=[]

nsd = []

for i in range(0,len(a)-1):

    for k in range(1,len(a)):

        if(k==i):

            break

        else:

            x_dist = (a[k] - a[i])

            y_dist = (b[k] - b[i])

            d = math.sqrt(x_dist * x_dist + y_dist * y_dist)

            distance.append(d)

            if(d <=220):

                nsd.append(i)

                nsd.append(k)

```

```

        nsd = list(dict.fromkeys(nsd))

        print(nsd)

color = (0, 0, 255)

for i in nsd:

    (x, y) = (boxes[i][0], boxes[i][1])

    (w, h) = (boxes[i][2], boxes[i][3])

    cv2.rectangle(image, (x, y), (x + w, y + h), color, 2)

    text = "Red Alert"

    cv2.putText(image, text, (x, y - 5), cv2.FONT_HERSHEY_SIMPLEX, 0.5,
color, 2)

color = (0, 255, 0)

if len(idxs) > 0:

    for i in idxs.flatten():

        if (i in nsd):

            break

        else:

            (x, y) = (boxes[i][0], boxes[i][1])

            (w, h) = (boxes[i][2], boxes[i][3])

            cv2.rectangle(image, (x, y), (x + w, y + h), color, 2)

            text = 'Normal'

            cv2.putText(image, text, (x, y - 5),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, color, 2)

```

```
cv2.imshow("Social Distancing Detector", image)

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

    break

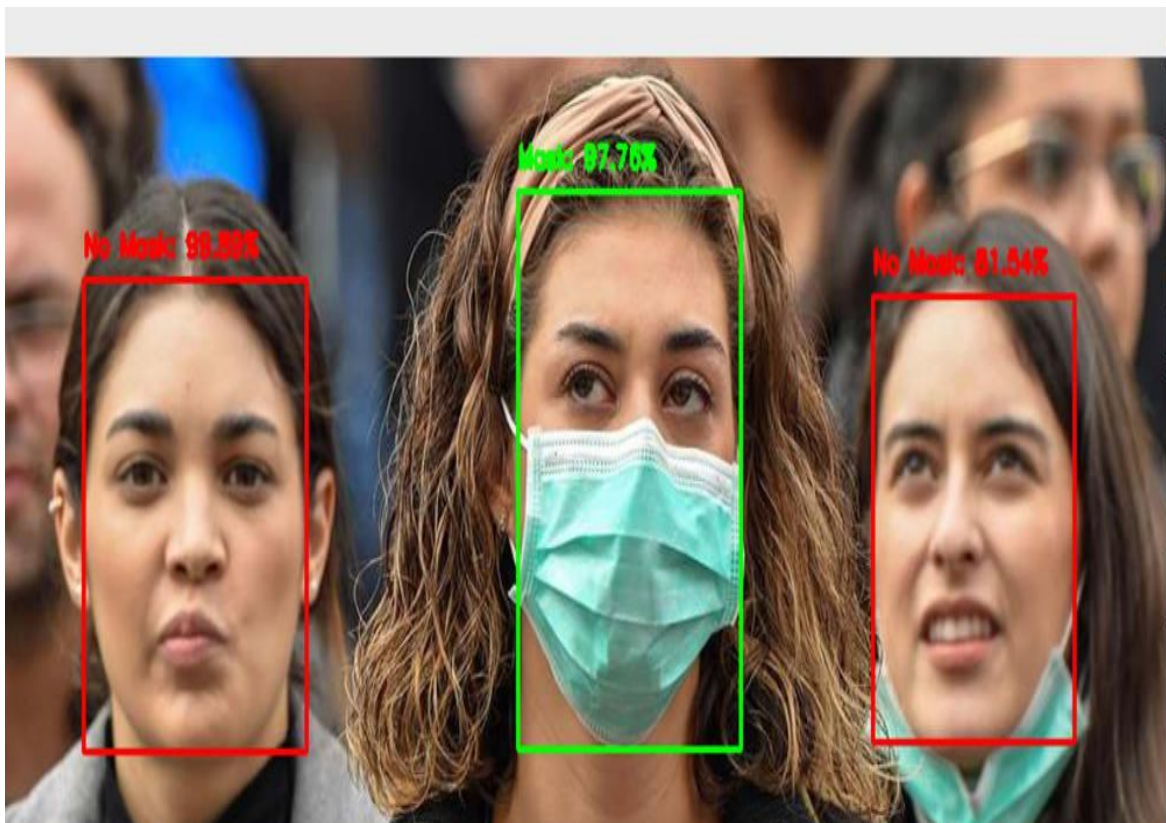
cap.release()

cv2.destroyAllWindows()
```

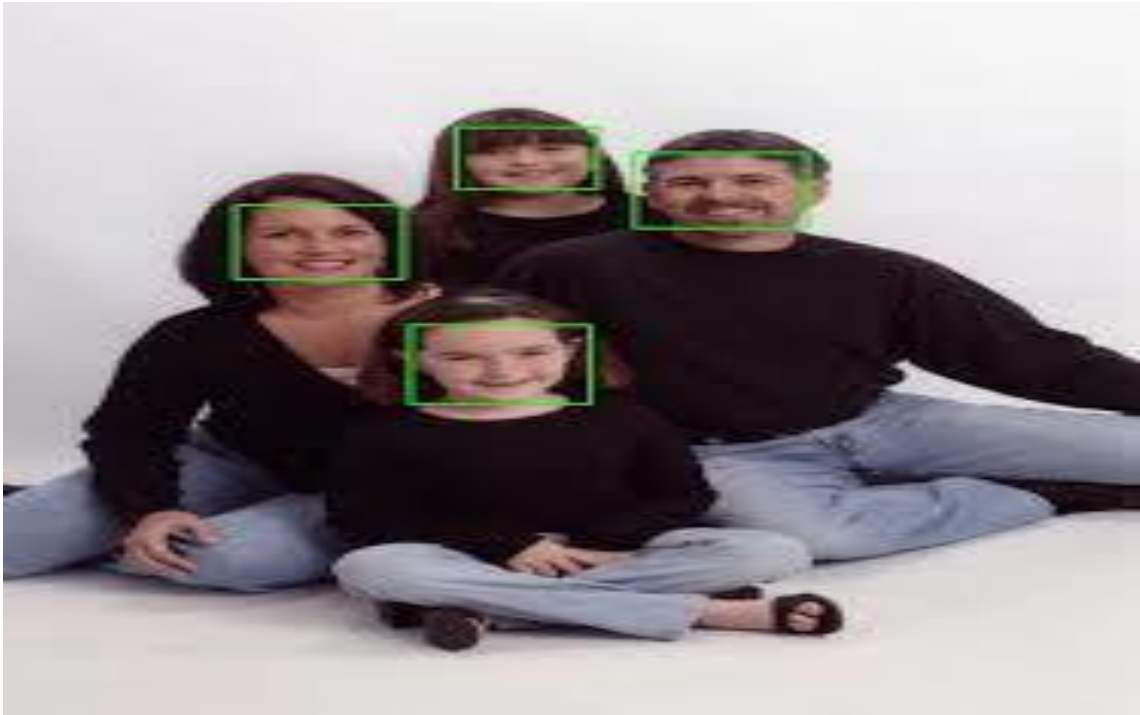
## APPENDIX-II

### SCREENSHOT

#### FACE MASK DETECTION:

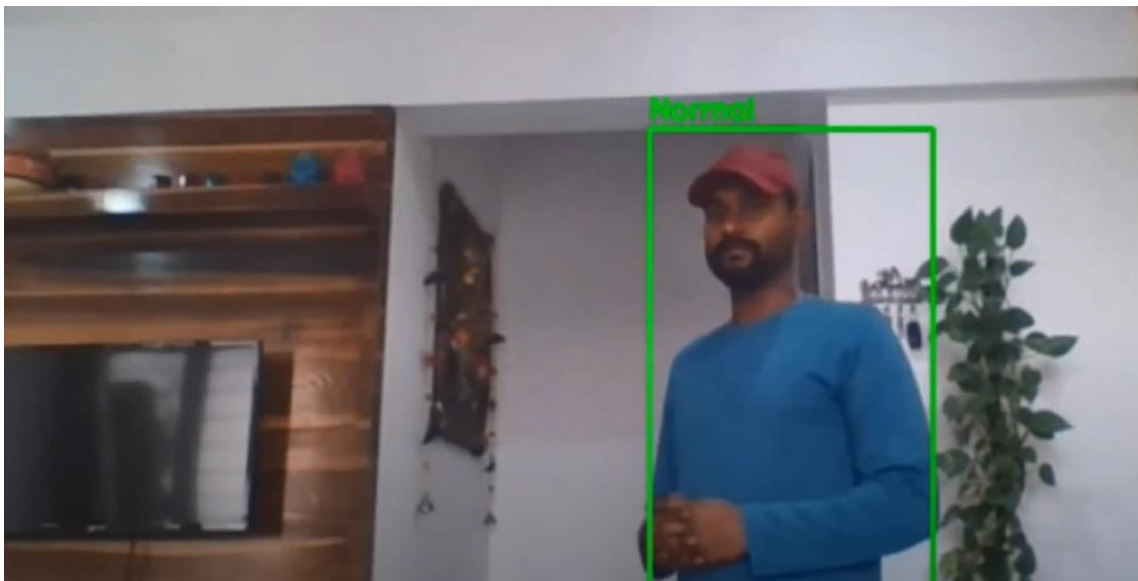


**Fig 8.1 Face mask detecting**



**Fig 8.2 Face Detection**

## **SOCIAL DISTANCING DETECTION:**



**Fig 8.3 Social distance detection**



**Fig 8.4 Social distance detection**

## REFERENCES

- [1] Learning OpenCV –Computer Vision with the OpenCV Library  
O'Reilly Publication.
- [2] Learning OpenCV: Computer Vision with OpenCV Library, Kindle  
Edition. Gary Bradski and Andrei Kheifman
- [3] M.A. Turk and A.P. Pentland, “Face Recognition Using Eigenfaces”,  
IEEE Conf. on Computer Vision and Pattern Recognition, pp. 586-591,  
1991
- [4] G B Huang, H Lee, E L. Miller, “Learning hierarchical representation  
for Face verification with convolution deep belief  
networks[C]”, Proceedings of International Conference on Computer  
Vision and Pattern Recognition, pp. 223-226, 2012.
- [5] Computer Vision Papers, <http://www.cvpapers.com>
- [6] Learning OpenCV: Computer Vision with the OpenCV Library 1st  
Edition, Kindle Edition [2]
- [7] OpenCV Homepage <http://opencv.willowgarage.com>
- [8] Recognition Homepage <http://www.face-rec.org/algorithms>.
- [9] Paul Viola, Matthew Jones Conference paper- IEEE Computer Society  
Conference on Computer Vision and Pattern Recognition. “Rapid object  
detection using a boosted cascade of simple features.”

- [10] P. Dollar, V. Rabaud, G. Cottrell, and S. Belongie, “Behavior recognition via sparse spatio-temporal features,” in 2005 IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance. IEEE, 2005, pp. 65–72.
- [11] M. Piccardi, “Background subtraction techniques: a review,” in 2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No. 04CH37583), vol. 4. IEEE, 2004, pp. 3099–3104.
- [12] Y. Xu, J. Dong, B. Zhang, and D. Xu, “Background modeling methods in video analysis: A review and comparative evaluation,” CAAI Transactions on Intelligence Technology, vol. 1, no. 1, pp. 43–60, 2016.
- [13] H. Tsutsui, J. Miura, and Y. Shirai, “Optical flowbased person tracking by multiple cameras,” in Conference Documentation International
- [14] A. Faizi (2008), “Robust Face Detection using Template Matching Algorithm,” University of Toronto, Canada.
- [15] P. Feng (2004), “Face Recognition based on Elastic Template,” Beijing University of Technology, China.
- [16] L.H. Liang, H.ZH. Ai & G.Y. Xu (2002), “A Survey of Human Face Detection,” J.Computers. China, Vol. 25, Pp. 1– 10.
- [17] K.J. Wang, SH.L. Duan & W.X. Feng (2008), “A Survey of Face Recognition using Single Training Sample”, Pattern Recognition and Artificial Intelligence, China, Vol. 21, Pp. 635–642.