VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JNANA SANGAMA", MACHHE, BELAGAVI-590018

Final Project Report

On

"**Social Distance Monitor: An AI Tool Using Computer vision and Deep learning**"

Submitted in partial fulfillment of the requirements for the VIII semester

Bachelor of Engineering

in

Information Science and Engineering

of

Visvesvaraya Technological University, Belagavi

by

Prasun Kumar **:1CD20IS081**

Monish Krishna K **:1CD20IS066**

Under the Guidance

of

Prof. Sudarsanan D

Assistant Professor,

Department of ISE, CITech.

Department of Information Science and Engineering

CAMBRIDGE INSTITUTE OF TECHNOLOGY, BANGALORE-560 036

2023-2024

CAMBRIDGE INSTITUTE OF TECHNOLOGY

K.R. Puram, Bangalore-560 036

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

CERTIFICATE

Certified that **Mr. Prasun Kumar, Mr. Monish Krishna K** bearing USN **1CD20IS081** and **1CD20IS066** bonafide students of **Cambridge Institute of Technology,** has successfully completed the Project Work entitled "**Social Distance Monitor: An AI Tool Using Computer vision and Deep learning**" in partial fulfillment of the requirements for VIII semester **Bachelor of Engineering** in **Information Science and Engineering** of **Visvesvaraya Technological University, Belagavi** during academic year 2023-2024. It is certified that all Corrections/Suggestions indicated for Internal Assessment have been incorporated in the report deposited in the department library. The Project report has been approved as it satisfies the academic requirements prescribed for the Bachelor of Engineering degree.

Project Guide **Prof. Sudarsanan D,** **Assistant Professor,** **Dept.of ISE, CITech**	**Project Co-ordinator** **Dr.Priyanka Desai,** **Prof.Shivakumar M,** **Dept.of ISE, CITech**	**Head Of Department** **Dr. Preethi S,** **Dept.of ISE, CITech**	**Principal** **Dr. G Indumathi,** **CITech**

Name of the Examiners	**Signature**
1. _____	_____
2. _____	_____

DECLARATION

We**, Mr. Prasun Kumar** and **Mr. Monish Krishna K** bearing USN **1CD20IS081** and **1CD20IS066** of VIII semester BE, Information Science and Engineering, Cambridge Institute of Technology, hereby declare that the project entitled "**Social Distance Monitor: An AI Tool Using Computer vision and Deep learning**" has been carried out by us and submitted in partial fulfillment of the program requirements of **Bachelor of Engineering** in **Information Science and Engineering** as prescribed by **Visvesvaraya Technological University, Belagavi,** during the academic year 2023- 2024.

Date:

Name	**USN**	**Signature**
Prasun Kumar	1CD20IS081	
Monish Krishna K	1CD20IS066	

ACKNOWLEDGEMENT

We would like to place on record our deep sense of gratitude to **Shri. D. K. Mohan,** Chairman, Cambridge Group of Institutions, Bangalore, India for providing excellent Infrastructure and Academic Environment at CITech without which this work would not have been possible**.**

We are extremely thankful to **Dr. Indumathi G,** Principal, CITech, Bangalore, for providing me the academic ambience and everlasting motivation to carry out this work and shaping our careers.

We express our sincere gratitude to **Dr. Preethi S,** HOD, Dept. of Information Science and Engineering, CITech, Bangalore, for her stimulating guidance, continuous encouragement and motivation throughout the course of present work.

We also wish to extend our thanks to Project Coordinators, **Dr. Priyanka Desai,** Assistant Professor and **Mr Shivakumar M**, Assistant Professor, Dept. of ISE, CITech, Bangalore for the critical, insightful comments, guidance and constructive suggestions to improve the quality of this work.

We also wish to extend our thanks to Project guide, **Prof. Sudarsanan D,** Assistant Professor, Dept. of ISE, CITech for her guidance and impressive technical suggestions to complete my Project work.

Finally, to all our friends, classmates who always stood by our side in difficult situation and also helped us in some technical aspects and last but not the least, we wish to express deepest sense of gratitude to our parents who were constant source of encouragement and stood by our side as pillar of strength for completing this work successfully.

Prasun Kumar

Monish Krishna K

ABSTRACT

The coronavirus disease (COVID-19) caused by the severe acute respiratory syndrome coronavirus 2(SARS-CoV-2) has resulted in a global pandemic. There are many preventive ways and one such is maintaining physical distancing also known as Social Distancing. According to the World Health Organization (WHO) maintaining a distance of 1 meter at least is enough to stop the transmission of the virus through air. Although the safe physical distancing to be maintained varies country to country, we have taken 1.8 meters (6 ft) as the safe distance for our project as it is the most acceptable one. We have built a Machine Learning based model which utilizes OpenCV computer vision framework to monitor if the social distancing is being maintained by the persons detected in the frame. There are three level of detection red, yellow and green. When persons are maintaining minimum physical distancing then the detected boxes are green indicating that its safe. When the persons are getting closer to the minimum distance to be maintained then it turns yellow, finally the boxes turn red if the persons are violating the minimum physical distancing. The project is dynamic and can be customized to run on a variety of inputs, it can take any video as input file, it can take input from the webcam of the laptop and also it can take input from a CCTV camera using its IP address and produce the real time output. The project also sends an alert email when the violations exceed a certain threshold. The project produces output with high accuracy while still leaving some room for future developments.

CONTENTS

List of Figures

List of Tables

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

Social distancing also known as physical distancing is one of the most effective measures to control the spread of coronavirus or any other communicable disease by minimising the close contacts between individuals. The CDC defines social distancing as it applies to COVID-19 as "remaining out of congregate settings, avoiding mass gatherings, and maintaining distance (approximately 6 feet or 2 meters) from others when possible." Individuals can practice social distancing by physically distancing themselves by at least 2 meters from others. Other social distancing method includes staying at home, limiting travel, avoiding crowded areas, using no-contact greetings like Namaskar. Many governments have made it mandatory for people to maintain physical distance in public areas.

The World Health Organisation has recommended that a distance of 1m (3.3ft) or more is safe and practicing it can reduce the spread of infection significantly. The minimum physical distance described as safe is variable and different countries have adopted different social distancing policy. China, Singapore, Denmark, Hong Kong, France and Lithuania have adopted 1m social distancing policy. South Korea has adopted 1.4m (4.6ft). Italy, Spain, Portugal, Netherland, Austria, Germany, Portugal and Greece adopted 1.5m (4.9ft). Canada adopted 2m (6.6ft). The United Kingdom first advised 2m then later reduced this to 1m. The United States has adopted 1.8 m (6 ft).

Although the term ‘Social Distancing’ was first coined in 21st century, evidence of practicing has been found dated back to 5th century. The Bible contains one of the earliest known references to the practice in the Book of Leviticus 13:46: "And the leper in whom the plague is... he shall dwell alone; [outside] the camp shall his habitation be."

In modern times, social distancing measures have been successfully implemented in several epidemics. For the 1918 flu, also known as the Spanish Flu, lasted until 1920 which is considered as the deadliest pandemic in the modern history, studies have found out that social distancing was one of the key to flattening the curve. And that likely remains true a century later, in the current battle against coronavirus.

1.2 REVELENCE OF PROJECT

The project leverages several key features and technologies to achieve its goals, which primarily involve human detection and social distancing monitoring using computer vision. Here's an overview of the relevance of each feature:

OpenCV as Computer Vision Library:

Relevance: OpenCV is a powerful open-source computer vision library that provides a comprehensive set of algorithms. Its relevance lies in enabling the project to utilize a wide range of optimized computer vision and machine learning algorithms for tasks such as image processing, object detection, and more. This library serves as the foundation for implementing the various functionalities required for human detection and distance monitoring.

YOLOv3 Model for Human Detection:

Relevance: YOLOv3 is chosen for its speed in object detection. In the context of real-time human detection, speed is crucial for maintaining accuracy without significant lag. While it might not be the most accurate algorithm, its speed makes it an excellent choice for applications where quick detection is essential, such as monitoring crowded areas in real time.

Threading for Lag Reduction:

Relevance: Threading is implemented to address the lag between the captured frame and the displayed frame. This is crucial for real-time applications, as it ensures that the system can process and display the video feed without noticeable delays. Threading helps optimize the performance of the application, contributing to a smoother and more responsive user experience.

Alert System with Email Notification:

Relevance: The alert system serves as a critical component for notifying users about potential violations of social distancing rules. Email alerts enhances the versatility of the system. The use of the smtplib library in Python allows for seamless integration of email alerts. Users can customize the sender's email, receiver's email, and the content of the email, providing flexibility in configuring the alert system according to specific requirements.

Real-Time Output with Color-Coded Boxes:

Relevance: The real-time visualization of the detected objects, along with color-coded boxes, enhances the interpretability of the system output. The changing color of the rectangular box based on the relative distance between individuals provides an intuitive visual representation. This feature aids in quickly identifying potential violations of social distancing guidelines, with green indicating a safe distance, yellow indicating proximity, and red indicating a serious violation.

1.3 PROBLEM STATEMENT

Social Distancing is one of the effective approaches to reduce the community spread of coronavirus. However, it is very common that people tend to break the rules of social distancing every now and then thus a constant supervision is required upon the public to ensure that required physical distance is maintained. Current solutions to practice social distancing requires large human resources and so they are neither practical nor viable in the long term among other issues. So, aim of our project is to design a Machine Learning based model which utilizes Computer Vision to ensure that social distancing is maintained and sends an alert in form of email and alarm when violations exceed a certain threshold limit**.**

1.4 METHODOLOGY

We can use OpenCV, computer vision, and deep learning to implement social distancing detectors. The steps to build a social distancing monitor tool include:

1. Apply object detection to detect all people (and only people) in a video stream Compute the pairwise distances between all detected people.

2. Based on these distances, check to see if any two people are less than N pixels apart.

3. Raise alarm and send email alert when violation count exceeds the threshold

CHAPTER 2

LITERATURE SURVEY

2.1 INTRODUCTION

A literature review is the identification and examining of the existing research work in the chosen field to gain valuable information. Literature review was performed to understand the existing learning algorithms and to choose the suitable supervised and unsupervised method for image classification. As the study was made to compare supervised and unsupervised algorithms,the literature review was performed to identify the most effective algorithm of each kind. The algorithms identified were further used in experimentation.

2.2 RELATED WORKS

1. **Imran Ahmed, Misbah Ahmad, Joel J.P.C. Rodrigues, Gwanggil Jeon, Sadia Dinf. "A deep learning-based social distance monitoring framework for COVID-19". Sustainable Cities and Society, Elsevier, February 2021**. [1]

   - This paper has proposed a pre-trained Yolo v3 based model to detect social distancing violations in the input video. They have then compared the results of the said model with other state-of-art Deep Learning based object detection models such as Fast-RCNNs, Faster-RCNNs and Mask-RCNNs. Furthermore, they implemented the transfer learning methodology to increase the accuracy of the model where the detection algorithm uses a pre-trained algorithm that is connected to an extra trained layer using an overhead human data set. Their pre-trained model achieved detection accuracy of 92% and 95% with transfer learning methodology and tracking accuracy of their model is 95%.

2. **Narinder Singh Punn, Sanjay Kumar Sonbhadra, Sonali Agarwal and Gaurav Rai. "Monitoring COVID-19 social distancing with person detection and tracking via fine-tuned YOLO v3 and Deepsort techniques". arXiv:2005.01385v4.** [2]

- This paper proposed a Deep Learning based framework for automating the task of social distance monitoring using surveillance videos. They have made used of Yolo v3 to detect humans and Deepsort approach to track humans and have further compared the results with models such as single shot detector (SSD) and faster R-CNNs. The comparison included real-time FPS and mean average precision (mAP).

3. **Vinitha V, Velantina V. "SOCIAL DISTANCING DETECTION SYSTEM WITH ARTIFICIAL INTELLIGENCE USING COMPUTER VISION AND DEEP LEARNING". International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056 Volume: 07 Issue: 08.** [3]

- This paper proposed an AI based tool to ensure social distancing protocol. They made use of Yolo v3 for object detection. The model fetches the Region of Interest (ROI) from the user and generates a Bird's eye view (top-down view) of the frame. This birds-eye view transformation helps in proper calibration of video which could differ because of camera's position. This assumes that every person is standing on same flat ground plane. This system provided better violation detection but lacked real-time output as outputs were stored in frames.

4. **"Mahdi Rezaei, Mohsen Azarmi." DeepSOCIAL: Social Distancing Monitoring and Infection Risk Assessment in COVID-19 Pandemic by Mahdi Rezaei and Mohsen Azarmi". Applied Sciences 2020, 10, 7514; MPDI Open Access Journals.** [4]

- In this paper, the authors have developed a hybrid Computer Vision and YOLOv4-based Deep Neural Network model for human detection from CCTV security camera footages. The proposed model is a combination of SORT tracking algorithm and inverse perspective mapping (IMP). The evaluation of model resulted in a mean average precision of 99.8% with real time speed of 24 FPS. The resulted model is generic and the model can be applied in any outdoor or indoor scenario.

2.3 EXISTING SYSTEM VS PROPOSED SYSTEM

Existing System

There are two types of existing system in use today to monitor social distancing viz manual monitoring using human resources and utilizing R-CNN based AI tools to monitor.
The existing system of manual process of determining social distancing between humans by utilizing human resources. It is tedious for any human to continuously keep an eye on the people who are not obeying the social distancing rule. There is no technology used in this system.

Disadvantages:

- The main drawback of this scheme is that it is not accurate and human error can occur.
- Tedious job for a human to monitor the rule.
- Multiple places are not possible for a human to monitor from a single place .
- Time consuming and difficult.

Another existing system which was developed for similar kind of use was object detection using R-CNN.

This R-CNN is considered as the first ever deep learning based object detectors. Before YOLOv3, R-CNN was used for deep leaning based object detection.

R-CNNs are one of the first deep learning-based object detectors and are an example of a two-stage detector. While R-CNNs tend to very accurate, the biggest problem with the R-CNN family of networks is their speed — they were incredibly slow, obtaining only 5 FPS on a GPU.

Proposed System

Our approach uses artificial intelligence tools such as deep learning and computer vision to track people in different environments and measure adherence to social distancing guidelines and can give notifications each time social distancing rules are violated also known as Smart Distancing. We have built a Machine Learning based model which utilizes OpenCV computer vision framework to monitor if the social distancing is being maintained by the persons detected in the frame. There are three level

of detection red, yellow and green. When persons are maintaining minimum physical distancing then the detected boxes are green indicating that its safe. When the persons are getting closer to the minimum distance to be maintained then it turns yellow, finally the boxes turn red if the persons are violating the minimum physical distancing. The project is dynamic and can be customized to run on a variety of inputs, it can take any video as input file, it can take input from the webcam of the laptop and also it can take input from a CCTV camera using its IP address and produce the real time output. The project also sends an alert email and rings an alarm bell when the violations exceed a certain threshold. The project produces output with high accuracy while still leaving some room for future developments.

CHAPTER 3

SYSTEM ANALYSIS

Analysis is the process of breaking a complex topic or substance into smaller parts to gain a better understanding of it. Gathering requirements is the main attraction of the Analysis Phase. The process of gathering requirements is usually more than simply asking the users what they need and writing their answers down. Depending on the complexity of the application, the process for gathering requirements has a clearly defined process of its own.

3.1 HARDWARE REQUIREMENTS

The hardware requirements include the requirements specification of the physical computer resources for a system to work efficiently. The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. The Hardware Requirements are listed below:

Processor	Any Processor above 500MHz, preferably an intel core i7 7th Gen
RAM	8GB
Hard Disk	50GB, preferably an SSD
Input Device	Standard Keyboard, Mouse and Camera
Output Device	VGA and High-Resolution Monitor
Internet	Internet connection is required to send email alert.

Table.3.1 Hardware Requirement

3.2 SOFTWARE REQUIREMENTS

The software requirements are description of features and functionalities of the target system. Requirements convey the expectations of users from the software product. The requirements can be obvious or hidden, known or unknown, expected or unexpected from client's point of view.

Operating System	Windows 10
IDE	Any Python IDE with installed libraries
Python	Python 3
Python Libraries	OpenCV, Sckit-learn, NumPy, Argparse, Imutlis, Winsound, Smtplib, SSL, OS, Time

Table.3.2 Software Requirements

CHAPTER 4

MODELING

4.1 PURPOSE

This chapter gives an overview of the design of the proposed system. The design covers the overall architecture of the system, starting with conceptual design and details added during subsequent phases of design. The static as well as dynamic behavior of the individual entities is detailed. The implementation and testing phases of the project are influenced by this documentation. The details are expected to evolve during the entire design process.

4.2 LEVELS OF SOFTWARE DESIGN

In the software engineering field software designing is a main section. Software design can be categorized into:

1. Architectural Design

This is the first level of the designing. Architectural design is the greatest summarize edition. of the system. This determines to the software or application as a method with more elements collaborating with each. Generally, this is a designing level where the designers obtain the ideaor thought of a suggested clarification domain or province.

2. High-level Design

This is the second level of the designing. The high-level design splits the theory or conceptsingle entity-multiple components into less-abstracted prospect of modules and subsystems. It represents their cooperation with each. Basically, the high-level design concentrates on how the system besides its entire element can be applied as modules. High level design concept identifies modular arrangement of all subsystems and also their connection and cooperation between them.

3. Detailed Design

This is the third level of the designing. This is the level where the designing deals with the accomplishment part, which will be finally seen by the system. This level of designing is more elaborated as compared to previous modules design and implementations. In this level of design, we determine the logical structure of all previous modules.

4.3 SCOPE

This application will analyze the distance among people and according to the set distance among them this system indicates that they are following the protocol or not. It can be implemented into various fields for example: in radioactive labs to maintain distance between people for precautionary measurement, in examination hall to check the distance between student when invigilator is busy in doing documentation or in absence of invigilator or it can be implemented in the places where act 144 is proceeded. This application can be implemented in the large field of society where social distance is required.

4.4 SYSTEM ARCHITECTURE

4.4.1 Block Diagram

A block diagram is a graphical representation of a system or process that uses simple, labeled shapes or "blocks" to represent different components or stages and lines to indicate the relationships between them. It provides a high-level overview and helps in understanding the structure and interactions within a system. The Block Diagram of the proposed system is shown below in fig. 4.1

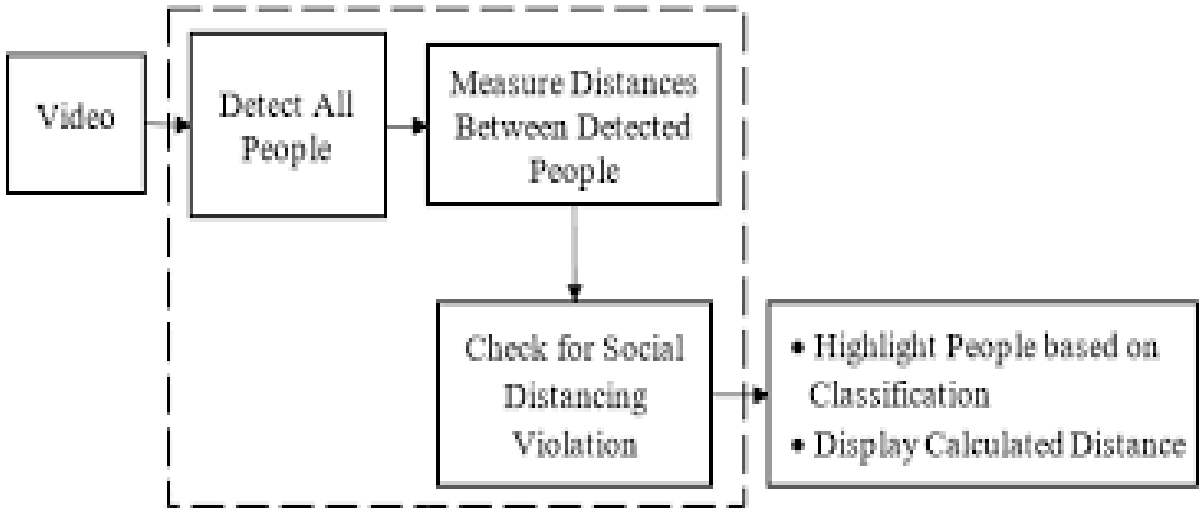

Fig.4.1 Block Diagram

4.4.2 Flow Chart

A flowchart is a visual representation of a process or system, depicting the steps or activities involved and the sequence in which they occur. It uses different shapes to represent different types of actions or decisions, and arrows to show the flow or direction of the process. Flowcharts are widely used in various fields, including software development, business processes, project management, and more, to illustrate and analyze complex workflows. The flow chart for the proposed system is shown below in fig. 4.2

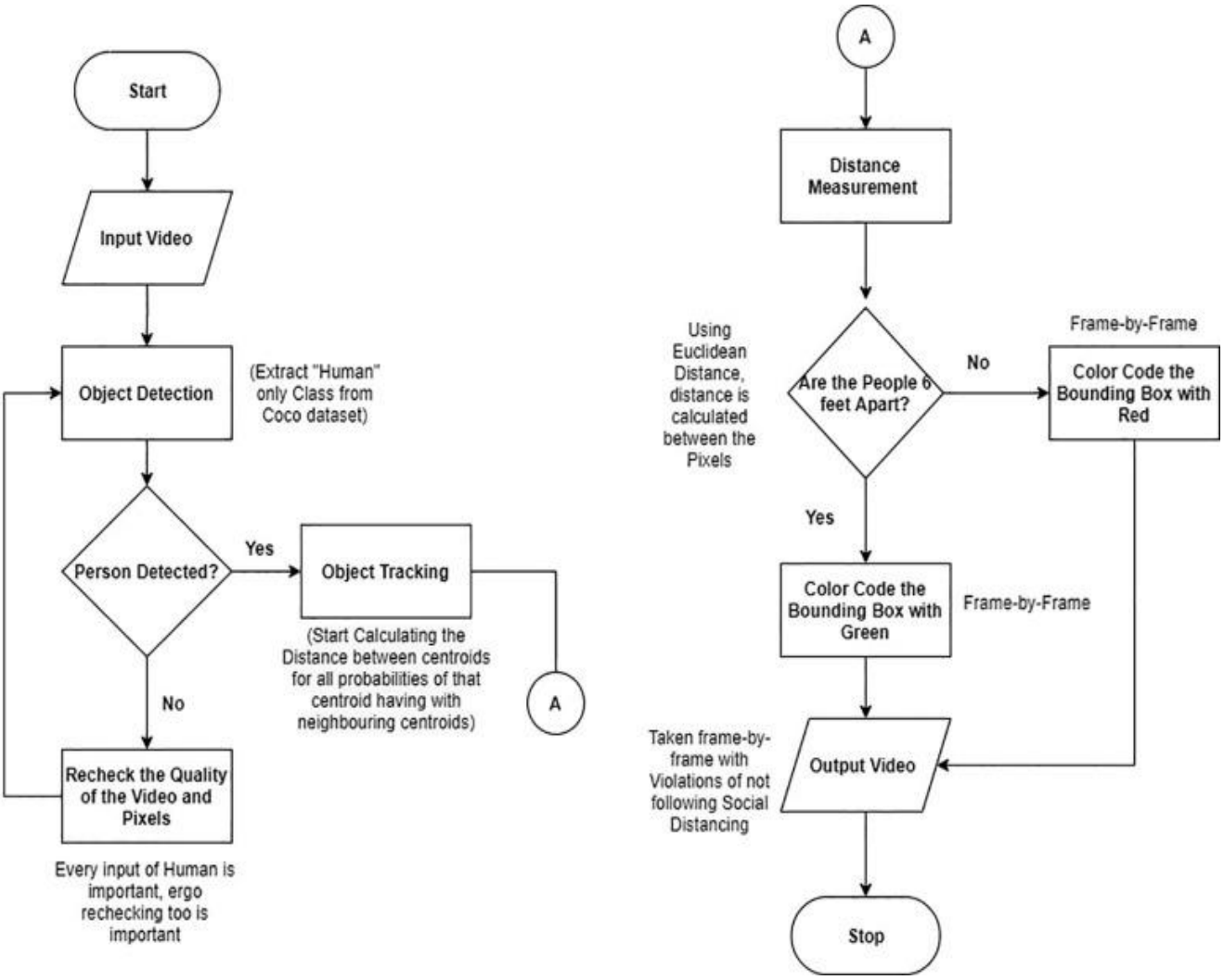

Fig.4.2 Flow Chart

CHAPTER 5

IMPLEMENTATION

5.1 HUMAN DETECTION

There are primarily three types of Deep Learning-based object detection models:

• RNN and its types which is Fast RCNN, Faster RCNN and Mask RCNN

• Single Shot Detectors

• YOLO

Each models have their own merits and demerits. The R-CNNs are examples of two-stage detector and were also the first Deep Learning based object detector models. They required an algorithm that proposed bounding boxes that contained objects and then these regions were passed to a CNN for classification leading to one of the first two stage detectors. The disadvantage of R-CNNs were that they were very slow, obtaining around 5 FPS even on a GPU. To increase the speed of detection sing-stage detectors like SSD and YOLO were proposed. These algorithms treat object detection as a regression problem, taking a given input image and simultaneously learning bounding box coordinates and corresponding class label probabilities

5.1.1 Yolo v3

The YOLO model was first proposed by Joseph Redmon in 2015. The model could detect in real time as fast as 45 FPS using a GPU and reaching up to 155 FPS on smaller variant of the model.

Instead of treating it as a classification, YOLO treats the object detection as a single regression problem. This means that the model looks only once at the image to detect what objects are present and where they are, hence it is named YOLO. The input image provided by the system is divided in to SxS grid. The grid cells individually predict B bounding boxes and calculates confidence scores for these bounding boxes. The confidence score indicates how sure the model is that the box contains an object and also how accurate it thinks the box is that predicts. The confidence score can be calculated using the formula:

$$C = Pr(object) * IoU$$

5.2 CENTROID CALCULATION

Once the detected bounding boxes are predicted and finalised using NMS. We proceed to calculate centroid of the rectangle. Centroid is the point of intersection of diagonals of the rectangular boxes. The detected bounding box coordinates (x,y) are used to compute the bounding box's centroid.

Formula: ((x1 + x2) / 2, (y1 + y2) / 2)

5.3 DISTANCE CALCULATION

After calculating centroid if each bounding boxes detected, the next step is to calculate the distance between each detected centroid using Euclidean distance. Euclidean distance between two points is the length of line segment jointing those two points.

$$\text{Euclidean distance} = \sqrt{\Sigma(xi\text{-}yi)^2}$$

We calculate the Euclidean distance between all the centroids detected. The process is repeated for every frame in the video stream when the count of person is greater than 2. The function will not work when the human count is 1 or 0.

5.4 VOILATION DETECTION

Violations between persons detected in the frame is measured based on the distance values. A threshold is defined to check if any two people are less than N pixels apart or not. The threshold value is taken from the user depending upon the video such than a CCTV footage taken from a distance has less threshold value and a closeup video has greater threshold value. added into the violation set. The bounding box colour is initialized as green. The information is checked in the violation set; if the current index exists violation set, the colour is updated to red. When the detected persons are getting closer i.e., within the violation range then the colour of box is updated to yellow. When the violations count exceeds a threshold a real time alert is displayed on the screen.

5.5 ALERT SYSTEM

When the violations count exceeds a threshold along with the alert displayed on screen an email alert and a sound alert get activated. The email alert is sent using the smtplib library on the recipient's email given by the user in configuration file. Along with the email alert, the system rings an alarm bell when number of violations exceeds the threshold limit. The alarm is generated using Python winsound library.

5.6 THREADING

While running the program on CPU there could be a delay in real-time stream. This arises due to the fact that OpenCV stores new frames which are yet to be processed until the system processes the old frames. This results in lag in stream and thus increases FPS. To reduce this lag, we make use of threading. Threading reads the frames as soon as they are available and discards the unprocessed frames.

5.7 CODE SNIPPET

main.py

```
from Files import config, thread
from Files.mailer import Mailer
from Files.detection import detect_people
from imutils.video import VideoStream, FPS
from scipy.spatial import distance as dist
import numpy as np
import argparse, imutils, cv2, os, time
```

```
import winsound

duration=1000

freq=440

ap = argparse.ArgumentParser()
ap.add_argument("-i", "--input", type=str, default=r"", help="path to
(optional) input video file")  #C:\Users\nikhi\Desktop\Social Distance
Monitor\Videos\test.mp4
ap.add_argument("-o", "--output", type=str, default=r"", help="path to
(optional) output video file") #C:\Users\nikhi\Desktop\Social Distance
Monitor\Output\test.avi
ap.add_argument("-d", "--display", type=int, default=1, help="whether or not
output frame should be displayed")
args = vars(ap.parse_args())
labelsPath = os.path.sep.join([config.MODEL_PATH, "coco.names"])
LABELS = open(labelsPath).read().strip().split("\n")
weightsPath = os.path.sep.join([config.MODEL_PATH, "yolov3.weights"])
configPath = os.path.sep.join([config.MODEL_PATH, "yolov3.cfg"])
net = cv2.dnn.readNetFromDarknet(configPath, weightsPath)
if config.USE_GPU:
    print(" ")
    print("--> Looking for GPU")
    net.setPreferableBackend(cv2.dnn.DNN_BACKEND_CUDA)
    net.setPreferableTarget(cv2.dnn.DNN_TARGET_CUDA)
```

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

if not args.get("input", False):
   print("--> Starting the live stream..")
   vs = cv2.VideoCapture(config.url)
   if config.Thread:
                cap = thread.ThreadingClass(config.url)
   time.sleep(1.0)

else:
   print("--> Starting the video..")
   vs = cv2.VideoCapture(args["input"])
   if config.Thread:
                cap = thread.ThreadingClass(config.url)
writer = None
fps = FPS().start()

while True:
   if config.Thread:
        frame = cap.read()

   else:
        (grabbed, frame) = vs.read()
        if not grabbed:
        break
```

```
frame = imutils.resize(frame, width=1000)
results = detect_people(frame, net, ln,
        personIdx=LABELS.index("person"))

serious = set()
abnormal = set()

if len(results) >= 2:
        centroids = np.array([r[2] for r in results])
        D = dist.cdist(centroids, centroids, metric="euclidean")

        for i in range(0, D.shape[0]):
                for j in range(i + 1, D.shape[1]):
                        if D[i, j] < config.MIN_DISTANCE:
                                serious.add(i)
                                serious.add(j)
                        if (D[i, j] < config.MAX_DISTANCE) and not serious:
                                abnormal.add(i)
                                abnormal.add(j)

for (i, (prob, bbox, centroid)) in enumerate(results):
        (startX, startY, endX, endY) = bbox
        (cX, cY) = centroid
        color = (0, 255, 0)

if i in serious:
```

```
            color = (0, 0, 255)
            elif i in abnormal:
                color = (0, 255, 255) #orange = (0, 165, 255)
            cv2.rectangle(frame, (startX, startY), (endX, endY), color, 2)
            cv2.circle(frame, (cX, cY), 5, color, 2)

    Safe_Distance = "Safe Distance: >{}
px".format(config.MAX_DISTANCE)
    cv2.putText(frame, Safe_Distance, (600, frame.shape[0] - 25),
            cv2.FONT_HERSHEY_SIMPLEX, 0.60, (255,255,255), 2)
    Threshold = "Threshold Limit: {}".format(config.Threshold)
    cv2.putText(frame, Threshold, (600, frame.shape[0] - 50),
            cv2.FONT_HERSHEY_SIMPLEX, 0.60, (255,255,255), 2)

    text = "    Total Serious Violations: {}".format(len(serious))
    cv2.putText(frame, text, (100, frame.shape[0] - 55),
            cv2.FONT_HERSHEY_SIMPLEX, 0.70, (0, 0, 255), 2)

    text1 = "    Total Abnormal Violations: {}".format(len(abnormal))
    cv2.putText(frame, text1, (100, frame.shape[0] - 25),
            cv2.FONT_HERSHEY_SIMPLEX, 0.70, (0, 255, 255), 2)
    if len(serious) >= config.Threshold:
            winsound.Beep(freq,duration)
            cv2.putText(frame, " ALERT!: VIOLATION OVER LIMIT", (20,
frame.shape[0] - 80),
            cv2.FONT_HERSHEY_COMPLEX, 1.0, (0, 0, 255), 2)
```

```
            if config.ALERT:
                print("")
                print('--> Sending mail alert...')
                Mailer().send(config.MAIL)
                print('--> Mail sent')
    if args["display"] > 0:
            cv2.imshow("Real-Time Monitoring Window", frame)
            key = cv2.waitKey(1) & 0xFF
            if key == ord("q"):
                break
    fps.update()
    if args["output"] != "" and writer is None:
            fourcc = cv2.VideoWriter_fourcc(*"XVID")
            writer = cv2.VideoWriter(args["output"], fourcc, 2,
                (frame.shape[1], frame.shape[0]), True)

    if writer is not None:
            writer.write(frame)

fps.stop()
print("============================")
print("\n--> Elasped Time: {:.2f}".format(fps.elapsed()))
print("--> Approx FPS: {:.2f}\n".format(fps.fps()))
print("============================")

cv2.destroyAllWindows()
```

config.py

```
MODEL_PATH = "yolo"
MIN_CONF = 0.3
NMS_THRESH = 0.3

People_Counter = True

Thread = False

Threshold = 8

url = 0 #'http://192.168.1.9:8080/video'

ALERT = True
MAIL = 'prasundwivedi9@gmail.com'

USE_GPU = False

MAX_DISTANCE = 500
MIN_DISTANCE = 400
#MAX_DISTANCE = 80
#MIN_DISTANCE = 50
```

detection.py

```
from .config import NMS_THRESH, MIN_CONF, People_Counter
import numpy as np
import cv2

def detect_people(frame, net, ln, personIdx=0):

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

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

    boxes = []
    centroids = []
    confidences = []

    for output in layerOutputs:

            for detection in output:

                    scores = detection[5:]
                    classID = np.argmax(scores)
            confidence = scores[classID]
```

```
                if classID == personIdx and confidence > MIN_CONF:

                    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)])
                    centroids.append((centerX, centerY))
                    confidences.append(float(confidence))

    idxs = cv2.dnn.NMSBoxes(boxes, confidences, MIN_CONF,
NMS_THRESH)

    if People_Counter:
        human_count = "Human count: {}".format(len(idxs))
        cv2.putText(frame, human_count, (600, frame.shape[0] - 75),
cv2.FONT_HERSHEY_SIMPLEX, 0.70, (0, 0, 0), 2)


    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])
```

```
                r = (confidences[i], (x, y, x + w, y + h), centroids[i])
                results.append(r)

    return
```

mailer.py

```
import smtplib, ssl

class Mailer:

  def __init__(self):
     self.EMAIL = "socialdistancingmonitorproject@gmail.com"
     self.PASS = "hyrv jagm ulgt tspq"
     self.PORT = 465
     self.server = smtplib.SMTP_SSL('smtp.gmail.com', self.PORT)

  def send(self, mail):
     self.server = smtplib.SMTP_SSL('smtp.gmail.com', self.PORT)
     self.server.login(self.EMAIL, self.PASS)
     SUBJECT = 'ALERT!'
     TEXT = f'Social Distancing Violations Exceeded! Need Urgent Supervision'
     message = 'Subject: {}\n\n{}'.format(SUBJECT, TEXT)

     self.server.sendmail(self.EMAIL, mail, message)
     self.server.quit()
```

thread.py

```
import cv2, threading, queue

class ThreadingClass:
  def __init__(self, name):
    self.cap = cv2.VideoCapture(name)
    self.q = queue.Queue()
    t = threading.Thread(target=self._reader)
    t.daemon = True
    t.start()

  def _reader(self):
    while True:
      (ret, frame) = self.cap.read()
      if not ret:
        break
      if not self.q.empty():
        try:
          self.q.get_nowait()
        except queue.Empty:
          pass
      self.q.put(frame)

  def read(self):
    return self.q.get()
```

CHAPTER 6

RESULTS

6.1 WHEN A VIDEO IS GIVEN AS INPUT

6.1.1 Sample Video 1

Fig. 6.1 Output Screenshot when a test pedestrian video given as input

6.1.2 Sample Video 2

Fig. 6.2 Output Screenshot when a second pedestrian test video given as input

6.2 WHEN LIVE WEBCAM FEED IS GIVEN AS INPUT

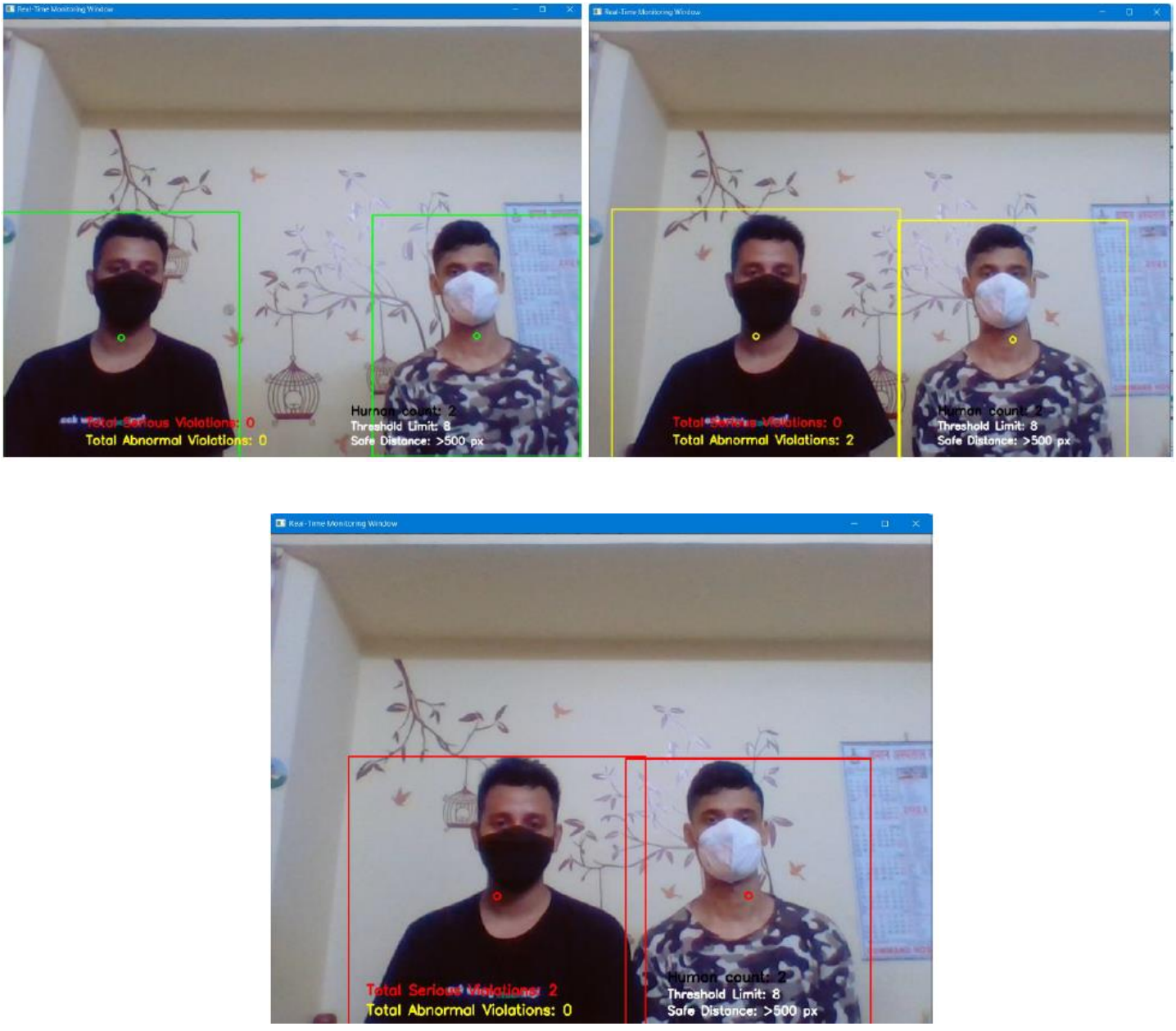

Fig. 6.3 Output screenshot for a webcam input. Here the three frames indicate three phases of detection which is safe, Abnormal and serious violation.

6.3 ALERT EMAIL RECEIVED

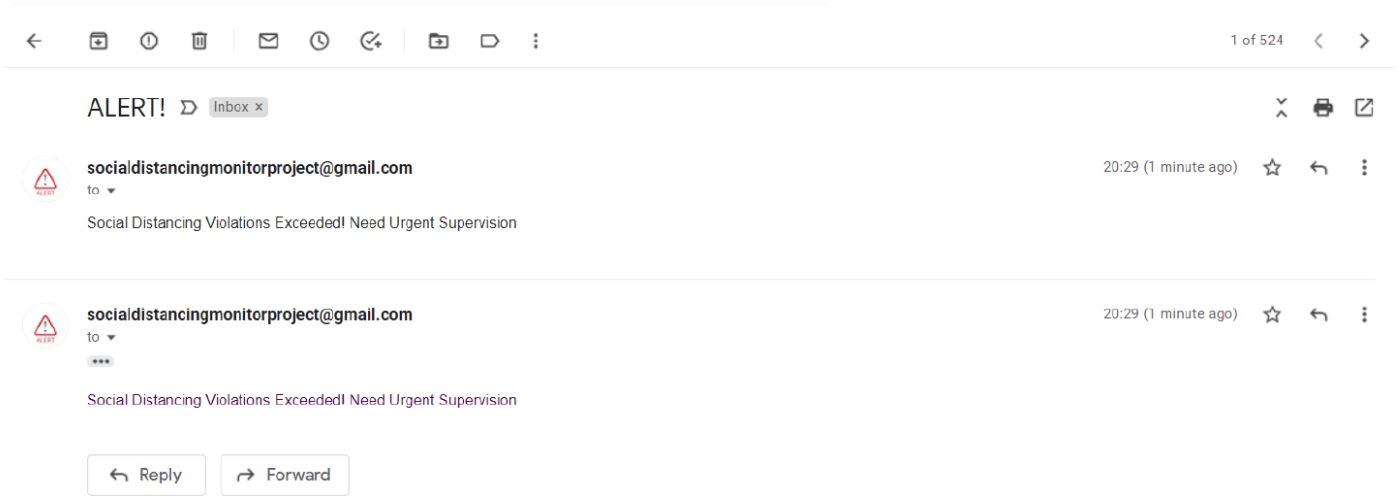

Fig. 6.4 Screenshot of email received when violation count exceed the threshold.

CHAPTER 7

CONCLUSION

The implementation of the Social Distance Monitor (SDM) marks a pivotal advancement in the realm of public health surveillance and intervention strategies, particularly in response to the challenges posed by the COVID-19 pandemic. By leveraging artificial intelligence (AI), computer vision, and deep learning technologies, the SDM offers a sophisticated solution for monitoring and enforcing social distancing protocols in real-time.

The SDM's ability to accurately identify and track individuals, assess spatial relationships, and measure distances with precision underscores its efficacy in promoting public safety. Its adaptability to diverse environments and crowd densities, coupled with advanced AI-driven analytics, ensures robust performance and enables proactive intervention when violations occur.

Beyond its immediate application in combating COVID-19, the SDM's versatility extends to various scenarios, including the management of other airborne diseases like Ebola, enforcing social distancing during curfews, and preventing close proximity among students during examinations. This adaptability underscores its potential as a multifaceted tool for safeguarding public health across different contexts.

Moreover, the real-time alert system integrated into the SDM enables prompt intervention by designated authorities, facilitating timely responses to instances of non-compliance. Customizable notification mechanisms further enhance its utility by enabling efficient communication and coordination among stakeholders.

In essence, the integration of AI, computer vision, and deep learning technologies in the SDM represents a significant step forward in enhancing public health surveillance capabilities and promoting adherence to social distancing measures. By empowering communities and authorities with real-time monitoring, proactive alerts, and data-driven insights, the SDM emerges as a crucial asset in safeguarding public health and well-being amidst evolving challenges and uncertainties.

CHAPTER 8

FUTURE SCOPE

The Social Distance Monitor (SDM) presents a promising future scope with its innovative integration of AI, computer vision, and deep learning technologies. Building upon its success in addressing the challenges of the COVID-19 pandemic, the SDM can evolve and expand its applications to meet various emerging needs in public health surveillance and beyond. Here are some potential future directions for the SDM:

- **Enhanced Feature Set**: Continuously refining the capabilities of the SDM to incorporate new features and functionalities can broaden its utility. This could include improvements in object detection algorithms, enhanced accuracy in distance measurement, and the ability to detect and track multiple types of objects beyond humans, such as vehicles or objects in industrial settings.

- **Integration with Wearable Technology**: Exploring the integration of the SDM with wearable devices equipped with sensors can offer additional layers of monitoring and feedback. Wearables could provide real-time feedback to individuals about their proximity to others, helping them maintain safe distances and promoting self-awareness of social distancing practices.

- **Geofencing and Location-Based Monitoring**: Implementing geofencing capabilities within the SDM could enable targeted monitoring and enforcement of social distancing protocols in specific geographical areas or high-risk zones. This could be particularly useful in settings such as airports, public transportation hubs, or crowded event venues.

- **IoT Integration and Data Analytics**: Leveraging the Internet of Things (IoT) to connect the SDM with other smart devices and sensors can enrich data collection and analysis capabilities. By aggregating data from multiple sources, such as environmental sensors or mobile devices, the SDM can provide deeper insights into patterns of social behavior and facilitate more informed decision-making by authorities.

- **Global Deployment and Standardization**: Scaling up the deployment of the SDM on a global scale and establishing standardized protocols for implementation and data sharing can promote consistency and interoperability across different regions and jurisdictions. This could facilitate collaboration between countries and enable more effective cross-border management of public health crises.

- **Integration with Public Health Systems**: Integrating the SDM with existing public health systems and infrastructure can streamline data exchange and coordination between healthcare authorities and monitoring agencies. This would enable seamless integration of social distancing monitoring data with other health indicators, enhancing overall situational awareness and response capabilities.

- **Continuous Research and Innovation**: Continued investment in research and development to advance the capabilities of the SDM and explore new applications in diverse domains will be essential. This includes ongoing collaboration between academia, industry, and government agencies to drive innovation and address evolving challenges in public health and safety.

REFERENCES

[1] Scientific Brief: SARS-CoV-2 Transmission | CDC https://www.cdc.gov/coronavirus/2019-ncov/science/science-briefs/sars-cov-2-transmission.html

[2] China Had First Coronavirus Case in November 2019 Itself - The Wire Science https://science.thewire.in/health/china-had-first-coronavirus-case-in-november-2019-itself/

[3] COVID-19 pandemic – Wikipedia https://en.wikipedia.org/wiki/COVID-19_pandemic

[4] Scientific Brief: SARS-CoV-2 Transmission | CDC https://www.cdc.gov/coronavirus/2019-ncov/science/science-briefs/sars-cov-2-transmission.html

[5] What is social distancing and how can it slow the spread of COVID-19? | Hub (jhu.edu) https://hub.jhu.edu/2020/03/13/what-is-social-distancing

[6] Social distancing – Wikipedia https://en.wikipedia.org/wiki/Social_distancing

[7] Leviticus 13 AKJV - And the LORD spake unto Moses and - Bible Gateway https://www.biblegateway.com/passage/?search=Leviticus+13&version=AKJV

[8] How they flattened the curve during the 1918 Spanish Flu (nationalgeographic.com) https://www.nationalgeographic.com/history/article/how-cities-flattened-curve-1918-spanish-flupandemic-coronavirus

[9] A deep learning-based social distance monitoring framework for COVID-19 (nih.gov). By Imran Ahmed, Misbah Ahmad, Joel J.P.C. Rodrigues, Gwanggil Jeon, and Sadia Din https://arxiv.org/abs/1804.02767

[10] [1506.02640] You Only Look Once: Unified, Real-Time Object Detection (arxiv.org). By Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi https://arxiv.org/abs/1506.02640

[11] [1804.02767] YOLOv3: An Incremental Improvement (arxiv.org). By Joseph Redmon, Ali Farhadi https://arxiv.org/abs/1804.02767