A

PROJECT REPORT

On

SOCIAL DISTANCING DETECTION

Submitted in partial fulfillment of the requirement for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted By

MOHAMMED SHARIQ SAADUDDIN AHMED	**18R91A05E2**
VADE SUSHMITHA REDDY	**18C21A0543**
JUJJURI SHANMUKH GOUD	**18R91A05D1**

Under the Guidance

Of

DR . SARANGAM KODATI

PROFESSOR

Department of CSE

Department of Computer Science and Engineering

TEEGALA KRISHNA REDDY ENGINEERING COLLEGE

Medbowli, Meerpet, Saroornagar, Hyderabad – 500097

(Affiliated to JNTUH, Approved by AICTE, Accredited by NBA & NAAC 'A')

(2018-2022)

CERTIFICATE

This is to certify that the Mini Project report on **"SOCIAL DISTANCING DETECTION",** is a bonafide work carried out by **MOHAMMED SHARIQ SAADUDDIN AHMED, VADE SUSHMITHA REDDY, JUJJURI SHANMUKH GOUD** bearing Roll No. **18R91A05E2, 18C21A0543, 18R91A05D1** in the partial fulfillment for the award of B.Tech degree in Computer Science and Engineering, Teegala Krishna Reddy Engineering College (TKEM-R9), Hyderabad, affiliated to Jawaharlal Nehru Technological University, Hyderabad under our guidance and supervision.

The results embodied in the project work have not been submitted to any other University or Institute for the award of any degree or diploma.

INTERNAL EXAMINER

Dr. Sarangam Kodati

Professor

……..

HEAD OF DEPARTMENT

Dr.CH.V.Phani Krishna

Professor

……..

EXTERNAL EXAMINER

……..……

PRINCIPAL

Dr. K. Venkata MuraliMohan

……

DECLARATION

We hereby declare that the Mini Project report entitled entitled **"SOCIAL DISTANCING DETECTION"** is done under the guidance of **Dr. Sarangam Kodati, Professor,** Department of Computer Science and Engineering, Teegala Krishna Reddy Engineering College, is submitted in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering**.

This is a record of bonafide work carried out by us in **Teegala Krishna Reddy Engineering College** and the results embodied in this technical seminar have not been reproduced or copied from any source. The results embodied in this technical seminar report have not been submitted to any other University or Institute for the award of any other degree or diploma.

Submitted by

MOHAMMED SHARIQ SAADUDDIN AHMED	**18R91A05E2**
VADE SUSHMITHA REDDY	**18C21A0543**
JUJJURI SHANMUKH GOUD	**18R91A05D1**

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompanies the successful completion of any task would be incomplete without the mention of the people who made it possible and whose encouragement and guidance have crowned our efforts with success.

We extend my deep sense of gratitude to **Dr.K.Venkata Murali Mohan**, **Principal** Teegala Krishna Reddy Engineering College, Meerpet, for permitting me to undertake this project.

We are indebted to **Dr.CH.V.Phani Krishna, Professor & Head of the Department**, Computer Science and Engineering, Teegala Krishna Reddy Engineering College, Meerpet for his support and guidance throughout our project.

We are indebted to the project coordinators **Dr. N.Vadivelan, Professor**, and **Dr.Rajaram Jotothu**, Professor, Computer Science and Engineering, Teegala Krishna Reddy Engineering College, Meerpet for his support and guidance throughout our project.

We are indebted to our guide **Dr. Sarangam Kodati, Professor**, Computer Science Engineering, Teegala Krishna Reddy Engineering College, Meerpet for his support and guidance throughout our project.

Finally, we express thanks to one and all that have helped me in successfully completing this technical seminar. Further I would like to thank my family and friends for their moral support and encouragement.

Submitted by

MOHAMMED SHARIQ SAADUDDIN AHMED	**18R91A05E2**
VADE SUSHMITHA REDDY	**18C21A0543**
JUJJURI SHANMUKH GOUD	**18R91A05D1**

CONTENTS

LIST OF DIAGRAMS

ABSTRACT

Social Distancing is one such terminology that has gained popularity over the past few months, thanks to COVID-19. People are forced to maintain a sufficient amount of distance between each other to prevent the spread of this deadly virus. Amidst this crisis, we decided to build a simple Social Distancing Detector that could monitor the practice of social distancing in a crowd. This project uses Deep Learning based YOLOv3 Pre trained model for object Detection, OpenCV python library for image processing and Centroid Tracking Algorithm For object tracking. The distance between people can be estimated and any noncompliant pair of people in the display will be indicated with a red frame and red line. The proposed method was validated on a pre-recorded video of pedestrians walking on the street. The result shows that the proposed method is able to determine the social distancing measures between multiple people in the video.

1. INTRODUCTION

1.1 INTRODUCTION

COVID-19 originated from Wuhan, China, has affected many countries worldwide since December 2019. On March 11, 2020, the World Health Organization (WHO) announced it a pandemic diseases as the virus spread through 114 countries, caused 4000 deaths and 118,000 active cases On October 7, 2020, they reported more than 35,537,491 confirmed COVID-19 cases, including 1,042,798 deaths. The latest number of infected people due to pandemic is shown in. Many healthcare organizations, scientists, and medical professionals are searching for proper vaccines and medicines to overcome this deadly virus, although no progress is reported to-date. To stop the virus spread, the global community is looking for alternate ways. The virus mainly spreads in those people; who are in close contact with each other (within 6 feet) for a long period. The virus spreads when an infected person sneezes, coughs, or talks, the droplets from their nose or mouth disperse through the air and affect nearby peoples. The droplets also transfer into the lungs through the respiratory system, where it starts killing lung cells. Recent studies show that individuals with no symptoms but are infected with the virus also play a part in the virus spread .Therefore, it is necessary to maintain at least 6 feet distance from others, even if people do not have any symptoms.

1.2 PROBLEM STATEMENT

The risks of virus spread can be minimized by avoiding physical contact among people. To overcome the virus' spread, by minimizing the physical contacts of humans, such as the masses at public places (e.g., shopping malls, parks, schools, universities, airports, workplaces), evading crowd gatherings, and maintaining an adequate distance between people. Social distancing is essential, particularly for those people who are at higher risk of serious illness from COVID-19. By decreasing the risk of virus transmission from an infected person to a healthy, the virus' spread and disease severity can be significantly reduced.

1.3 OBJECTIVE OF PROJECT

The objective is to reduce transmission, reducing the size of the epidemic peak, and spreading cases over a longer time to relieve pressure on the healthcare.

We all can done this by:

- People Detection using machine Learning.
- Social Distancing Detection using Image Processing.
- Development of web application for remote video streaming.
- Sending alert messages using twilio API.

2. LITERATURE SURVEY

2.1 LITERATURE SUREY

After the rise of the COVID-19 pandemic since late December 2019, Social distancing is deemed to be an utmost reliable practice to prevent the contagious virus transmission and opted as standard practice on January 23, 2020. During one month, the number of cases rises exceptionally, with two thousand to four thousand new confirmed cases reported per day in the first week of February 2020. Later, there has been a sign of relief for the first time for five successive days up to March 23, 2020, with no new confirmed cases (N. H. C. of the Peoples Republic of China, 2020). This is because of the social distance practice initiated in China and, latterly, adopted by worldwide to control COVID-19. Ainslie et al. (2020) investigated the relationship between the region's economic situation and the social distancing strictness. The study revealed that moderate stages of exercise could be allowed for evading a large outbreak. So far, many countries have used technology-based solutions (Punn, Sonbhadra, & Agarwal, 2020a) to overcome the pandemic loss. Several developed countries are employing GPS technology to monitor the movements of the infected and suspected individuals. Nguyen et al. (2020) provides a survey of different emerging technologies, including Wi-fi, Bluetooth, smartphones, and GPS, positioning (localization), computer vision, and deep learning that can play a crucial role in several practical social distancing scenarios. Some researchers utilize drones and other surveillance cameras to detect crowd gatherings (Harvey and LaPlace, 2019, Robakowska et al., 2017).

Until now researchers have done considerable work for detection (Iqbal, Ahmad, Bin, Khan, & Rodrigues, 2020; Patrick et al., 2020; Yash Chaudhary & Mehta, 2020), some provides an smart healthcare system for pandemic using Internet of Medical Things (Chakraborty, 2021; Chakraborty et al., 2021). Prem et al. (2020) studied the social distancing impacts on the spread of the COVID-19 outbreak. The studies concluded that the early and immediate practice of social distancing could gradually reduce the peak of the virus attack. As we all know, that although social distancing is crucial for flattening the infection curve, it is an economically unpleasant step. In Adolph, Amano, Bang-Jensen, Fullman, and Wilkerson (2020), Adolph et al. highlighted the United States of America's condition during the pandemic. Due to a lack of general support by decision-makers, it was not implemented at an initial stage, starting harm to public health. However, social distancing influenced economic productivity; even then, numerous scholars sought alternatives that overcame the loss.

Researchers provide effective solutions for social distance measuring using surveillance videos along with computer vision, machine learning, and deep learning-based approaches. Punn et al. (2020b) proposed a framework using the YOLOv3 model to detect humans and the Deepsort approach to track the detected people using bounding boxes and assigned IDs information. They used an open image data set (OID) repository, a frontal view data set. The authors also compared results with faster-RCNN and SSD.

Ramadass et al. (2020) developed an autonomous drone-based model for social distance monitoring. They trained the YOLOv3 model with the custom data set. The data set is composed of frontal and side view images of limited people. The work is also extended for the monitoring of facial masks. The drone camera and the YOLOv3 algorithm

help identify the social distance and monitor people from the side or frontal view in public wearing masks.Pouw, Toschi, van Schadewijk, and Corbetta (2020) suggested an efficient graph-based monitoring framework for physical distancing and crowd management.

Sathyamoorthy, Patel, Savle, Paul, and Manocha (2020) performed human detection in a crowded situation. The model is designed for individuals who do not obey a social distance restriction, i.e., 6 feet of space between them. The authors used a mobile robot with an RGB-D camera and a 2D lidar to make collision-free navigation in mass gatherings.

From the literature, we concluded that the researcher had done a considerable amount of work for monitoring of social distance in public environments. But, most of the work is focused on the frontal or side view camera perspective. Therefore, in this work, we presented an overhead view social distance monitoring framework that offers a better field of view and overcomes the issues of occlusion, thereby playing a key role in social distance monitoring to compute the distance between peoples.

2.2 EXISTING SYSTEM

- It was difficult for the machine to determine who is maintaining social distancing and the real-world applications were limited.
- People are not detected from all angles by the current technology.
- A physical person should be there to observe whether or not the individuals are practicing social distancing.
- Sometimes human errors also occurs so results are not accurate.

2.3 PROPOSED SYSTEM:

- **Real-Time alert:** If selected, we send an email alert in real-time.
- **Threading:**

  Threading removes OpenCV's internal buffer (which basically stores the new frames yet to be processed until your system processes the old frames) and thus reduces the lag/increases fps.

- **People counter:**

  If enabled, we simply count the total number of people: set People Counter = True in the config.

- **Desired violations limits:**

  You can also set your desired minimum and maximum violations limits.

2.3.1 ADVANTAGES

- Limit opportunities to come in contact with contaminated person.
- Provides alerts if specific distance not maintained.
- Provides safety.
- Easy to operate.

2.4 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- Economical feasibility
- Technical feasibility
- Social feasibility

2.4.1 ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

2.4.2 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system

must have a modest requirement, as only minimal or null changes are required for implementing this system.

2.4.3 SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

3. ANALYSIS

3.1 REQUIREMENTS

3.1.1 Functional Requirements:

- Apply **object detection** to detect all people (and *only* people) in a video stream
- **Compute the pairwise distances** between all detected people
- Based on these distances, check to see if any two people are less than *N* pixels apart.
- **OpenCV:** OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library.
- **YOLOv3:** YOLOv3 (You Only Look Once, Version 3) is a real-time object detection algorithm that identifies specific objects in videos, live feeds, or images.

3.1.2 Non-Functional Requirements:

The major non-functional Requirements of the system are as follows

- Performance
- Accuracy
- User interface

Performance

Requests should be processed within no time.

User Interface

A menu interface has been provided to the client to be user friendly.

Accuracy

The quality or state of being correct or precise.

3.2 SYSTEM REQUIREMENTS

3.2.1 Hardware Requirements

Processor : Intel core I3

Ram : 4 GB

Hard Disk : 40 GB

GPU : 2GB

Webcam

3.2.2 Software Requirements

Operating system : Windows 10

Programming : Python

IDE : Pycharm

Frameworks : Flask, Tensorflow, Keras

Datasets : Admissions Dataset

Library : Open CV & YoloV3

4. DESIGN

4.1 SYSTEM ARCHITECTURE:

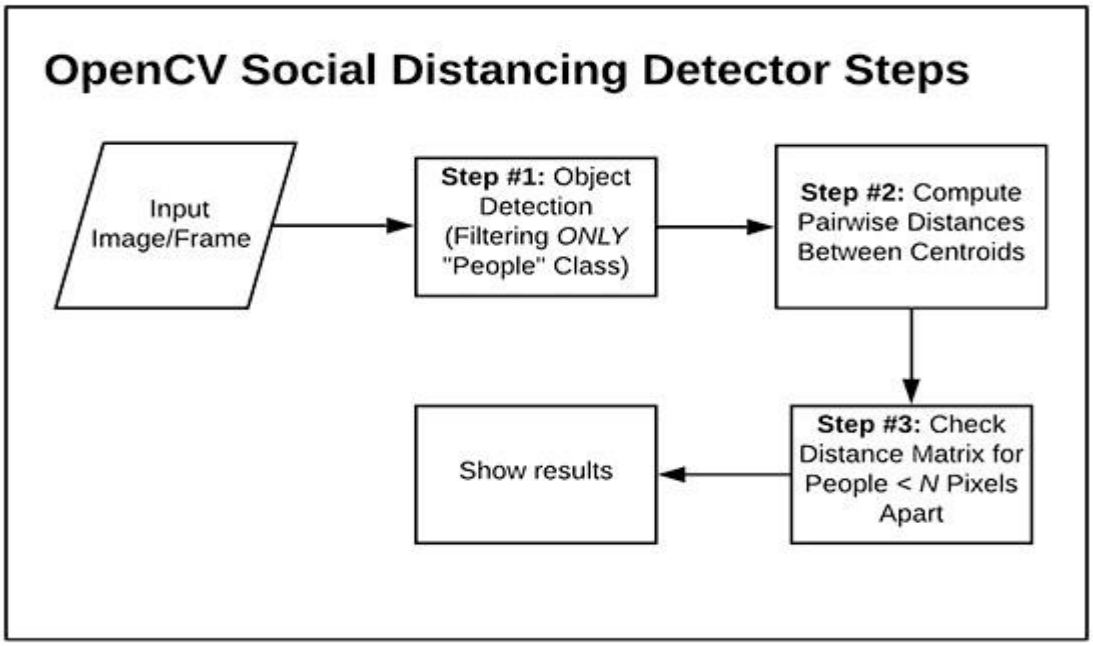

Fig 4.1 SYSTEM ARCHITECTURE

4.2 UML DIAGRAMS

UML stands for Unified Modelling Language. UML is a standardized general-purpose modelling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Metamodel and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modelling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modelling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modelling of large and complex systems.

The UML is a very important part of developing object oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

GOALS: The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modelling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modelling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.

4.2.1 USE CASE DIAGRAM

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

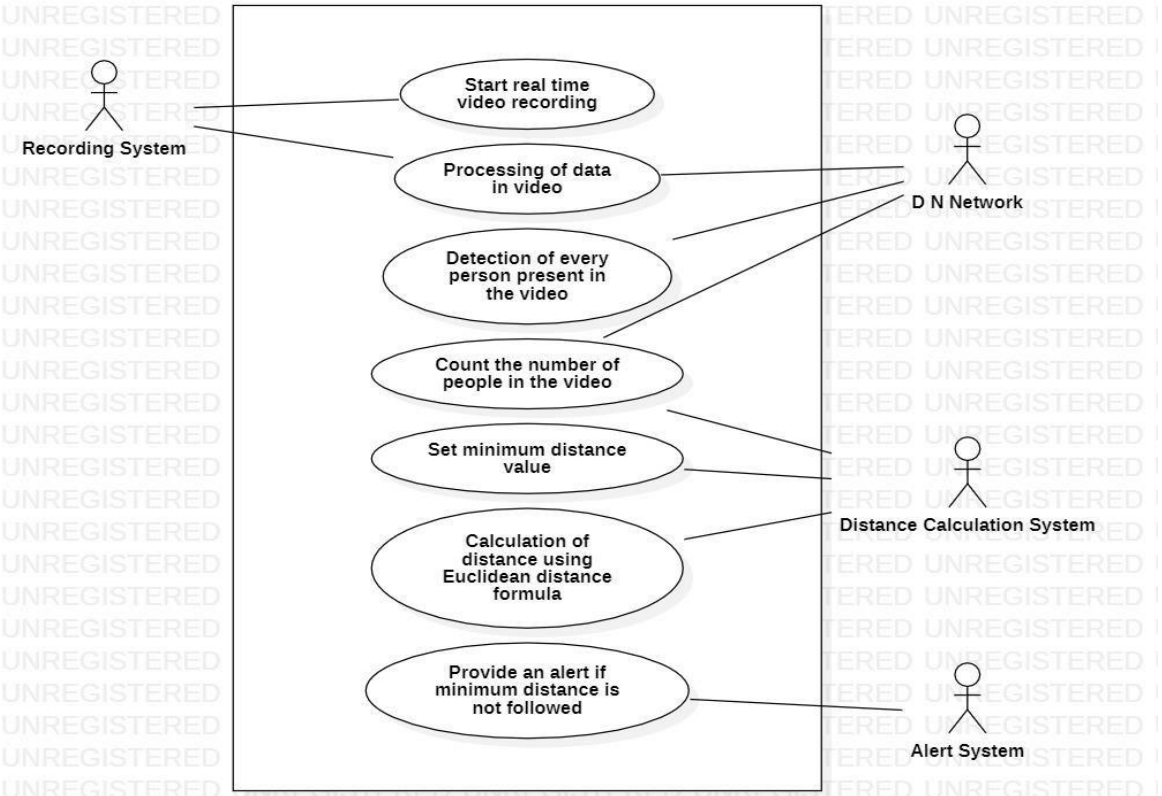

Fig 4.2.1 USE CASE DIAGRAM

4.2.2 SEQUENCE DIAGRAM

A sequence diagram in Unified Modelling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

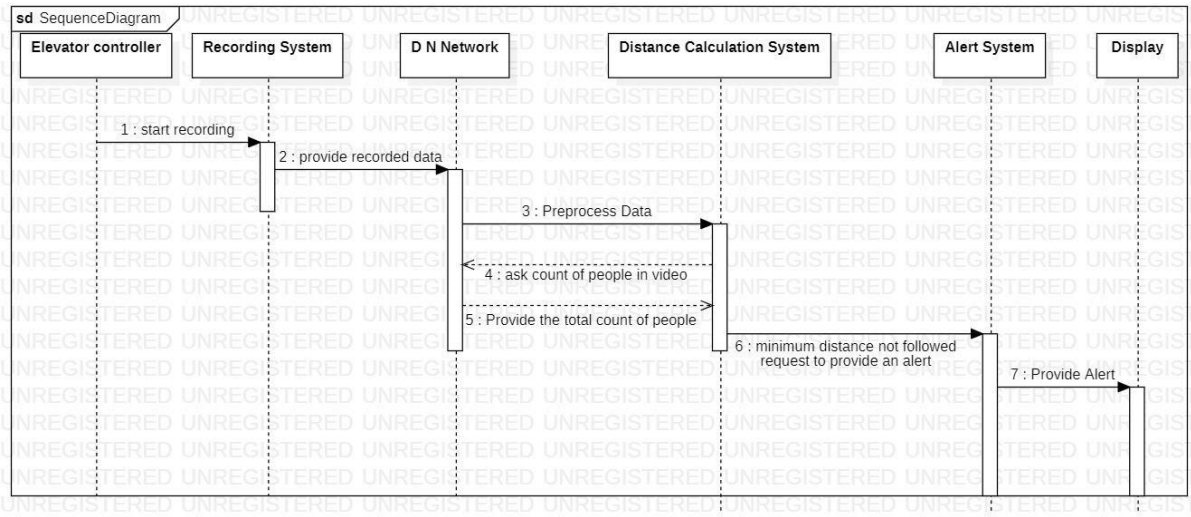

FIG 4.2.2 SEQUENCE DIAGRAM

4.2.3 ACTIVITY DIAGRAM

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

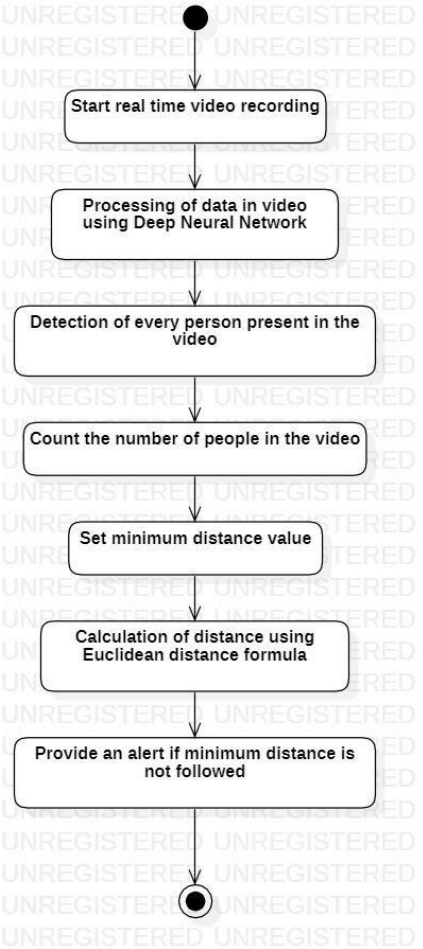

FIG 4.2.3 ACTIVITY DIAGRAM

4.2.4 CLASS DIAGRAM

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

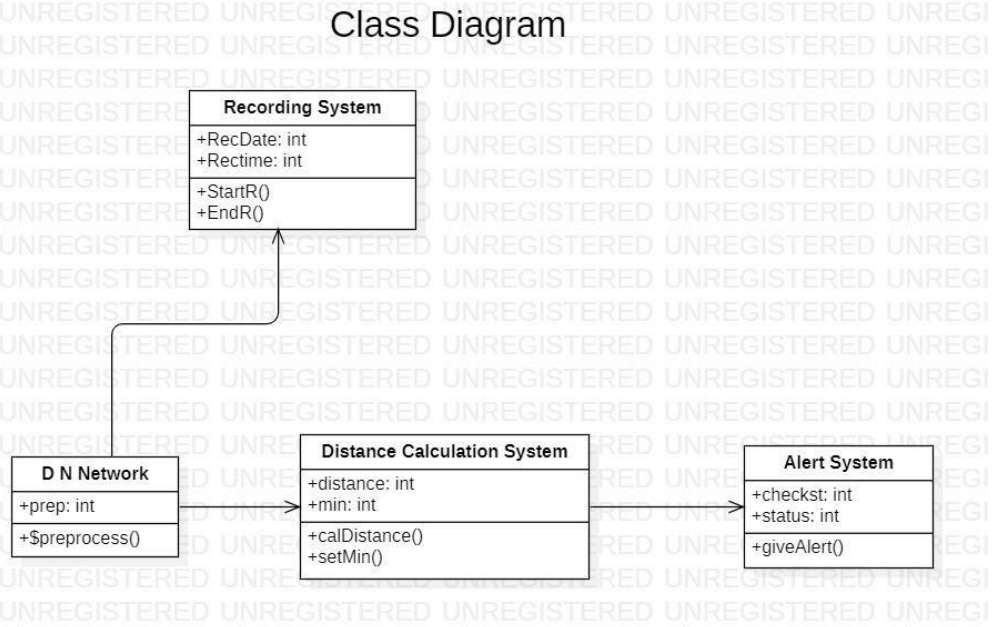

FIG 4.2.4 CLASS DIAGRAM

4.3 METHODOLOGY

This social distancing detection tool was developed to detect the safety distance between people in public spaces. The deep CNN method and computer vision techniques are employed in this work. Initially, an open-source object detection network based on the YOLOv3 [13] algorithm was used to detect the pedestrian in the video frame. From the detection result, only pedestrian class was used and other object classes are ignored in this application. Hence, the bounding box best fits for each detected pedestrian can be drawn in the image, and these data of detected pedestrians will be used for the distance measurement.

For camera setup, the camera is captured at fixed angle as the video frame, and the video frame was treated as perspective view are transformed into a two-dimensional top-down view for more accurate estimation of distance measurement. In this methodology, it is assumed that the pedestrians in the video frame are walking on the same flat plane. Four filmed plane points are selected from frame and then transformed into the top-down view. The location for each

pedestrian can be estimated based on the top-down view. The distance between pedestrians can be measured and scaled. Depending on the preset minimum distance, any distance less than the acceptable distance between any two individuals will be indicated with red lines that serve as precautionary warnings. The work was implemented using the Python programming language. The pipeline of the methodology for the social distancing detection tool is shown in Figure 2.

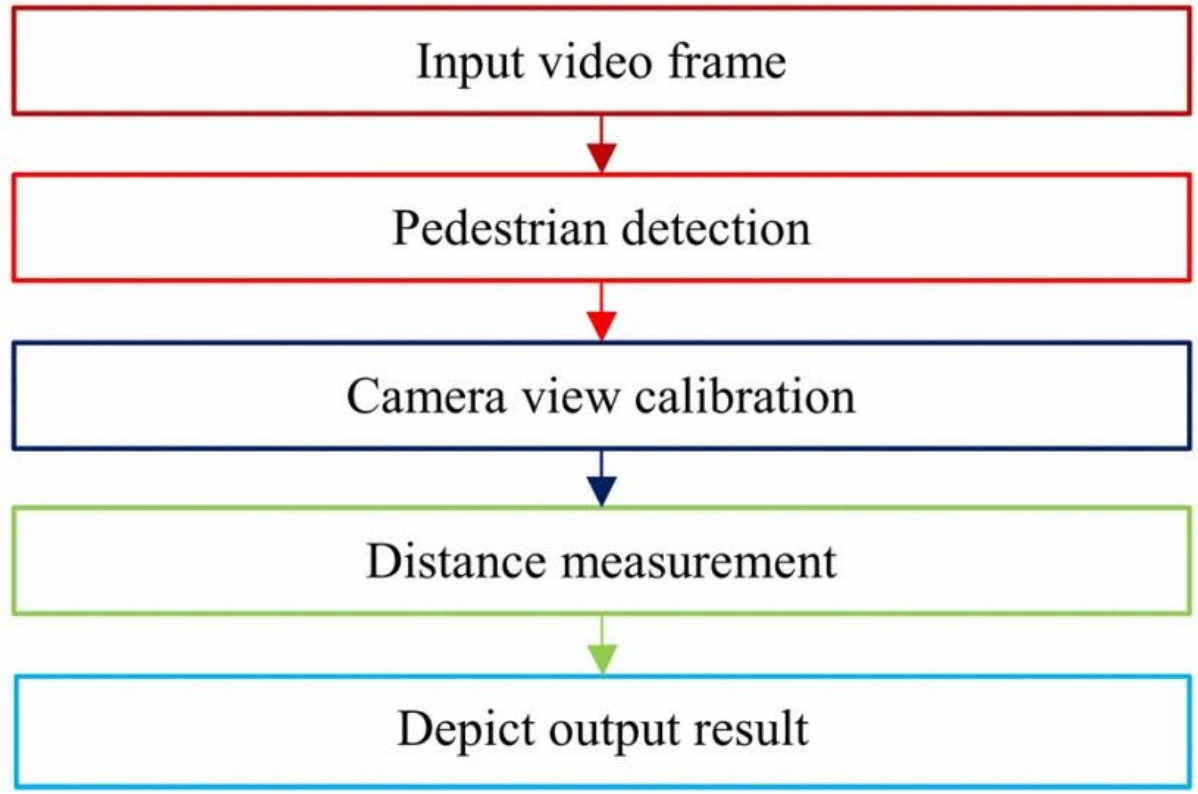

Figure 2

Pipeline for social distancing detection.

4.3.1Pedestrian Detection

Deep CNN model was the object detection approach was proposed that mitigated the computational complexity issues by formulating the detection with a single regression problem [11]. When it comes to deep learning-based obj ect detection, the YOLO model is considered one of the state-of-the art obj ect detectors which can be demonstrated to provide significant speed advantages will suitable for real-time application. In this work, the YOLO model was adopted for pedestrian detection is shown in Figure 3. The YOLO algorithm was considered as an obj ect detection taking a given input image and simultaneously learning bounding box coordinates (tx, ty, tw, th), obj ect confidence and corresponding class label probabilities (P1, P2, …, Pc). The YOLO trained on the COCO dataset which consists of 80 labels including human or pedestrian classs. In this work, the only box coordinates, object confidence and

pedestrain object class from detection result in the YOLO model were used for pedestrian detection.

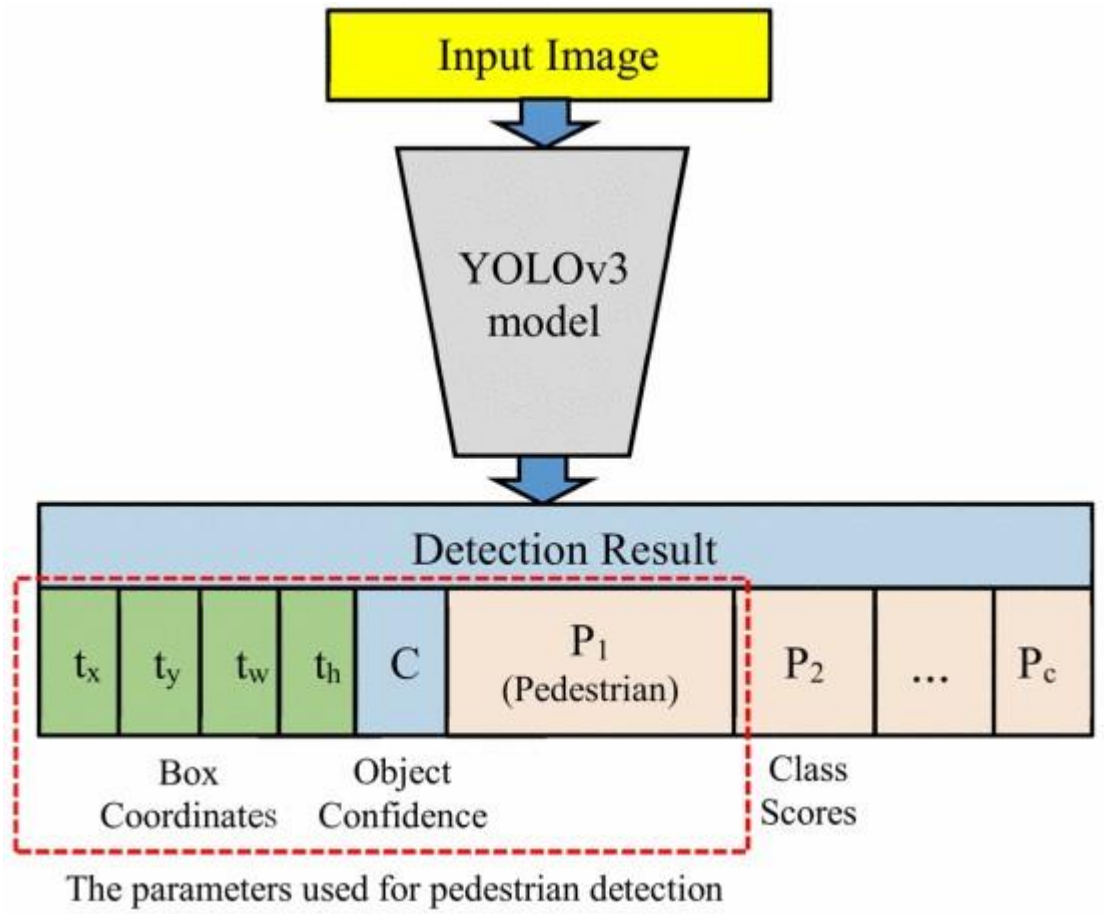

Figure 3

YOLO model applied for pedestrian detection.

4.3.2 Camera View Calibration

The region of interest (ROI) of an image focuses on the pedestrian walking street was transformed into a top-down 2D view that contains 480×480 pixels as shown in Figure 4. Camera view calibration is applied which works by computing the transformation of the perspective view into a top-down view. In OpenCV, the perspective transformation is a simple camera calibration method which involves selecting four points in the perspective view and mapping them to the corners of a rectangle in the 2D image view. Hence, every person is assumed to be standing on the same level flat plane. The actual distance between pedestrians corresponds to the number of pixels in the top-down view can be estimated.

4.3.3 Distance Measurement

In this step of the pipeline, the location of the bounding box for each person (x, y, w, h) in the perspective view is detected and transformed into a top-down view. For each pedestrian, the position in the top-down view is estimated based on the bottom-center

point of the bounding box. The distance between every pedestrian pair can be computed from the top-down view and the distances is scaled by the scaling factor estimated from camera view calibration. Given the position of two pedestrians in an image
as (x1, y1) and (x2,y2) respectively, the distance between the two pedestrians, d., can be computed as:

$$d= \sqrt{(x2-x1)2+(y2-y1)2} \quad (1)$$

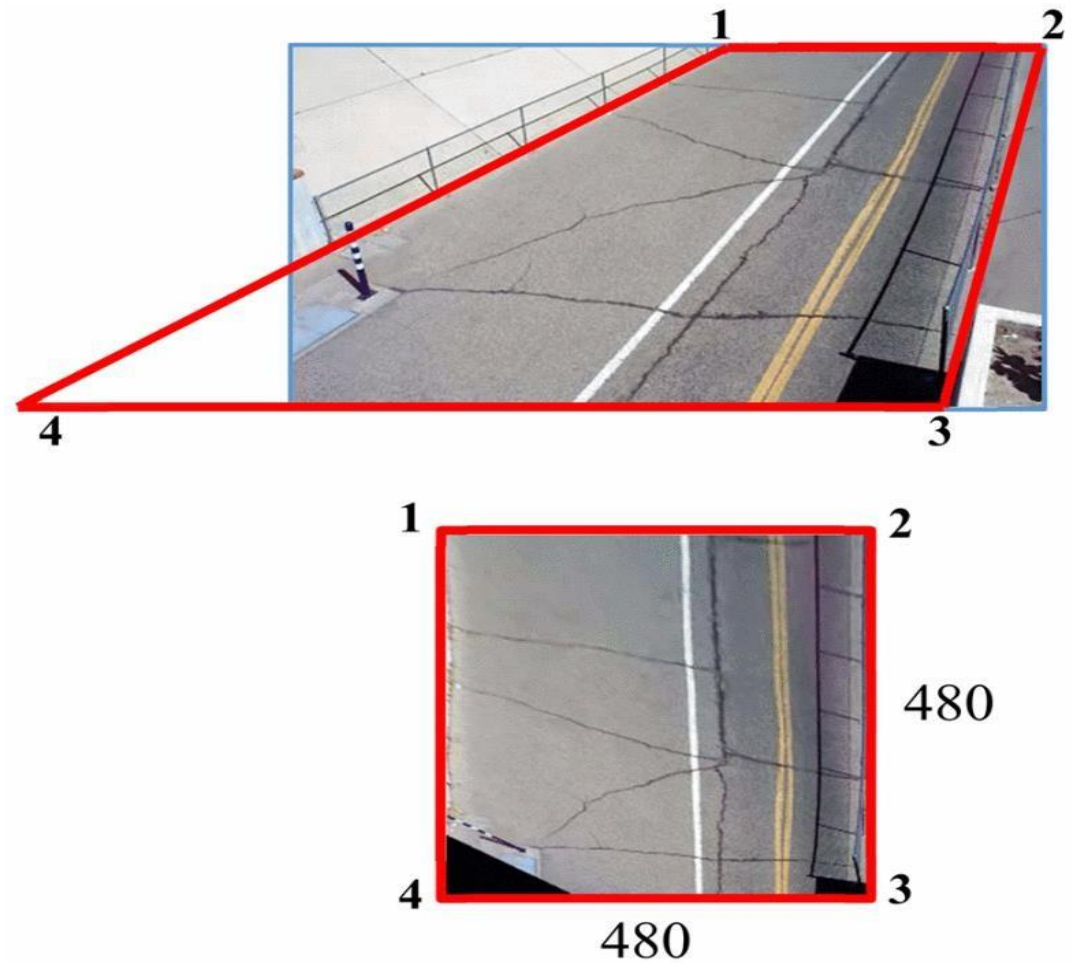

Figure 4

Sample of original perspective view (top) and top down view after calibration (bottom).

The pair of pedestrians whose distance is below the minimum acceptable distance, t., is marked in red, and the rest is marked in green. A red line is also drawn between the pair of individuals whose distance is below the pre-defined threshold. The bounding box's color threshold operation, c, can be defined as:

$$c=\{redgreend<td\geq t(2)$$

4.4 SOURCE CODE

4.4.1 MODEL GENERATION

```
# base path to YOLO directory

MODEL_PATH = "yolo"

# initialize minimum probability to filter weak detections along with

# the threshold when applying non-maxima suppression

MIN_CONF = 0.3

NMS_THRESH = 0.3


#================================================================
==============

#================================\CONFIG./=======================
=============

""" Below are your desired config. options to set for real-time inference """

# To count the total number of people (True/False).

People_Counter = True

# Threading ON/OFF. Please refer 'mylib>thread.py'.

Thread = False

# Set the threshold value for total violations limit.

Threshold = 15

# Enter the ip camera url (e.g., url = 'http://191.138.0.100:8040/video');

# Set url = 0 for webcam.

url = ''

# Turn ON/OFF the email alert feature.
```

```
ALERT = False

# Set mail to receive the real-time alerts. E.g., 'xxx@gmail.com'.

MAIL = ''

# Set if GPU should be used for computations; Otherwise uses the CPU by default.

USE_GPU = True

# Define the max/min safe distance limits (in pixels) between 2 people.

MAX_DISTANCE = 80

MIN_DISTANCE = 50
```

```
# import the necessary packages from .config import

NMS_THRESH, MIN_CONF, People_Counter import numpy as

np import cv2


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

        # grab the dimensions of the frame and  initialize the list of

        # results

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

results = []


        # construct a blob from the input frame and then perform a forward

        # pass of the YOLO object detector, giving us our bounding boxes

        #       and       associated       probabilities       blob       =

        cv2.dnn.blobFromImage(frame, 1 / 255.0, (416, 416),

                swapRB=True, crop=False)
```

```
        net.setInput(blob) layerOutputs
        = net.forward(ln)

        # initialize our lists of detected bounding boxes, centroids, and
        # confidences, respectively
boxes = []      centroids = []
confidences = []

        # loop over each of the layer outputs
for output in layerOutputs:             # loop
over each of the detections             for
detection in output:
                        # extract the class ID and confidence (i.e., probability)
                        # of the current object
detection                       scores = detection[5:]
                classID = np.argmax(scores)
confidence = scores[classID]

                        # filter detections by (1) ensuring that the object
                        # detected was a person and (2) that the minimum
                        # confidence is met
                        if classID == personIdx and confidence > MIN_CONF:
                                # scale the bounding box coordinates back relative to
                                # the size of the image, keeping in mind that YOLO
```

```
                    # actually returns the center (x, y)-coordinates of
                    # the bounding box followed by the boxes' width and
                    # height
                    box = detection[0:4] * np.array([W, H, W, H])
                    (centerX, centerY, width, height) = box.astype("int")

                    # use the center (x, y)-coordinates to derive the
top                 # and and left corner of the bounding box
                x = int(centerX - (width / 2))                  y =
int(centerY - (height / 2))

                    # update our list of bounding box coordinates,
                # centroids, and confidences
boxes.append([x, y, int(width), int(height)])
centroids.append((centerX, centerY))
confidences.append(float(confidence))

        # apply non-maxima suppression to suppress weak, overlapping
        # bounding boxes    idxs    =    cv2.dnn.NMSBoxes(boxes,    confidences,
MIN_CONF, NMS_THRESH)
        #print('Total people count:', len(idxs))
        # compute the total people counter if
        People_Counter:
```

```
                human_count = "Human count: {}".format(len(idxs))
                cv2.putText(frame, human_count, (470, frame.shape[0] - 75),
cv2.FONT_HERSHEY_SIMPLEX, 0.70, (0, 0, 0), 2)

        # ensure at least one detection exists
if len(idxs) > 0:
                # loop over the indexes we are keeping
        for i in idxs.flatten():
                        # extract the bounding box coordinates
                        (x, y) = (boxes[i][0], boxes[i][1])
                        (w, h) = (boxes[i][2], boxes[i][3])

                        # update our results list to consist of the person
                        # prediction probability, bounding box coordinates,
                        # and the centroid
                        r = (confidences[i], (x, y, x + w, y + h), centroids[i])
                        results.append(r)

        # return the list of results
return results
```

```
import smtplib, ssl

class Mailer:
```

```
    """
    This script initiaties the email alert function.
    """                 def
__init__(self):
        # Enter your email below. This email will be used to send alerts.
        #          E.g.,          "email@gmail.com"
self.EMAIL = ""
        # Enter the email password below. Note that the password varies if you have secured
# 2 step verification turned on. You can refer the links below and create an application
specific password.
        # Google mail has a guide here: https://myaccount.google.com/lesssecureapps
        # For 2 step verified accounts: https://support.google.com/accounts/answer/185833
        # Example: aoiwhdoaldmwopau         self.PASS = ""
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)
        # message to be sent
        SUBJECT = 'ALERT!'
        TEXT = f'Social distancing violations exceeded!'
message = 'Subject: {}\n\n{}'.format(SUBJECT, TEXT)
# sending the mail         self.server.sendmail(self.EMAIL, mail,
message)         self.server.quit()
```

```
import cv2, threading, queue


class ThreadingClass:
  #   initiate   threading   class
def __init__(self, name):
    self.cap = cv2.VideoCapture(name)
          # define an empty queue and thread
    self.q   =   queue.Queue()              t   =
threading.Thread(target=self._reader)
    t.daemon = True
    t.start()


  # read the frames as soon as they are available, discard any unprocessed frames;
# this approach removes OpenCV's internal buffer and reduces the frame lag
def _reader(self):     while True:
      (ret, frame) = self.cap.read() # read the frames and ---
      if not ret:         break
if not self.q.empty():
try:

         self.q.get_nowait()
except queue.Empty:
         pass       self.q.put(frame) # --- store them in a queue (instead of
the buffer)
```

```
 def read(self):
   return self.q.get() # fetch frames from the queue one by one
```

4.4.2 LOADING MODEL

```
from mylib import config, thread from
mylib.mailer import Mailer from
mylib.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, schedule

#----------------------------Parse req. arguments-----------------------------#
ap = argparse.ArgumentParser() ap.add_argument("-i", "--input",
type=str, default="",   help="path to (optional) input video file")
ap.add_argument("-o", "--output", type=str, default="",       help="path
to (optional) output video file") ap.add_argument("-d", "--display",
type=int, default=1,    help="whether or not output frame should be
displayed") args = vars(ap.parse_args())
#------------------------------------------------------------------------------#

# load the COCO class labels our YOLO model was trained on labelsPath
= os.path.sep.join([config.MODEL_PATH, "coco.names"])
```

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

# derive the paths to the YOLO weights and model configuration weightsPath
= os.path.sep.join([config.MODEL_PATH, "yolov3.weights"]) configPath =
os.path.sep.join([config.MODEL_PATH, "yolov3.cfg"])

# load our YOLO object detector trained on COCO dataset (80 classes) net
= cv2.dnn.readNetFromDarknet(configPath, weightsPath)

# check if we are going to use GPU if
config.USE_GPU:
        # set CUDA as the preferable backend and target
        print("")          print("[INFO] Looking for GPU")
net.setPreferableBackend(cv2.dnn.DNN_BACKEND_CUDA)
net.setPreferableTarget(cv2.dnn.DNN_TARGET_CUDA)
# determine only the *output* layer names that we need from YOLO
ln = net.getLayerNames() ln = [ln[i[0] - 1] for i in
net.getUnconnectedOutLayers()]

# if a video path was not supplied, grab a reference to the camera if
not args.get("input", False):
        print("[INFO] Starting the live stream..")      vs =
cv2.VideoCapture(config.url)            if config.Thread:
```

```
        cap = thread.ThreadingClass(config.url)
time.sleep(2.0)

# otherwise, grab a reference to the video file else:
        print("[INFO] Starting the video..")
vs = cv2.VideoCapture(args["input"])
if config.Thread:
                cap = thread.ThreadingClass(args["input"])

writer = None # start
the FPS counter fps =
FPS().start()

# loop over the frames from the video stream
while True:
        # read the next frame from the file
if config.Thread:
                frame = cap.read()

        else:
                (grabbed, frame) = vs.read()
                # if the frame was not grabbed, then we have reached the end of the stream
                if not grabbed:
                        break
```

```
        # resize the frame and then detect people (and only people) in
it      frame = imutils.resize(frame, width=700)    results =
detect_people(frame, net, ln,
                personIdx=LABELS.index("person"))

        # initialize the set of indexes that violate the max/min social distance limits
        serious     =     set()
abnormal = set()

        # ensure there are *at least* two people detections (required in
        # order to compute our pairwise distance maps)
        if len(results) >= 2:
                # extract all centroids from the results and compute the
                # Euclidean distances between all pairs of the centroids
                centroids = np.array([r[2] for r in results])
                D = dist.cdist(centroids, centroids, metric="euclidean")

                # loop over the upper triangular of the distance matrix
        for i in range(0, D.shape[0]):                  for j in
range(i + 1, D.shape[1]):
                                # check to see if the distance between any two
                                # centroid pairs is less than the configured number of pixels
                                if D[i, j] < config.MIN_DISTANCE:
```

```
                    # update our violation set with the indexes of the centroid pairs
                                    serious.add(i)
                        serious.add(j)
                    # update our abnormal set if the centroid distance is below max distance limit
                                    if (D[i, j] < config.MAX_DISTANCE) and not serious:
                                        abnormal.add(i)
                                        abnormal.add(j)

        # loop over the results          for  (i,  (prob,  bbox,
centroid)) in enumerate(results):
                # extract the bounding box and centroid coordinates, then
                # initialize the color of the annotation
                (startX, startY, endX, endY) = bbox
                (cX, cY) = centroid
                color = (0, 255, 0)

        # if the index pair exists within the violation/abnormal sets, then update the color
                if i in serious:
                        color  =  (0,  0,  255)
        elif i in abnormal:
                        color = (0, 255, 255) #orange = (0, 165, 255)

                # draw (1) a bounding box around the person and (2) the
```

```
                # centroid coordinates of the person,
cv2.rectangle(frame, (startX, startY), (endX, endY), color, 2)
cv2.circle(frame, (cX, cY), 5, color, 2)

        # draw some of the parameters
        Safe_Distance = "Safe distance: >{} px".format(config.MAX_DISTANCE)
cv2.putText(frame, Safe_Distance, (470, frame.shape[0] - 25),
                cv2.FONT_HERSHEY_SIMPLEX, 0.60, (255, 0, 0), 2)
        Threshold = "Threshold limit: {}".format(config.Threshold)
cv2.putText(frame, Threshold, (470, frame.shape[0] - 50),
                cv2.FONT_HERSHEY_SIMPLEX, 0.60, (255, 0, 0), 2)

   # draw the total number of social distancing violations on the output frame text
        = "Total serious violations: {}".format(len(serious)) cv2.putText(frame,
        text, (10, 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, (10, frame.shape[0] - 25),
                cv2.FONT_HERSHEY_SIMPLEX, 0.70, (0, 255, 255), 2)

#------------------------------Alert function----------------------------------#
        if len(serious) >= config.Threshold:
        cv2.putText(frame, "-ALERT: Violations over limit-", (10, frame.shape[0] - 80),
```

```
                    cv2.FONT_HERSHEY_COMPLEX, 0.60, (0, 0, 255), 2)
        if config.ALERT:
                    print("")
                    print('[INFO] Sending mail...')
        Mailer().send(config.MAIL)
print('[INFO] Mail sent')
                    #config.ALERT = False
#------------------------------------------------------------------------------#
        # check to see if the output frame should be displayed to our screen
if args["display"] > 0:
                    # show the output frame                cv2.imshow("Real-Time
Monitoring/Analysis Window", frame)                 key = cv2.waitKey(1) &
0xFF

                    # if the `q` key was pressed, break from the loop
                    if key == ord("q"):
                              break
   # update the FPS counter
        fps.update()

        # if an output video file path has been supplied and the video
        # writer has not been initialized, do so now   if
args["output"] != "" and writer is None:                # initialize our
```

```
video writer                fourcc = cv2.VideoWriter_fourcc(*"MJPG")
        writer = cv2.VideoWriter(args["output"], fourcc, 25,
                        (frame.shape[1], frame.shape[0]), True)

        # if the video writer is not None, write the frame to the output video file
if writer is not None:
                writer.write(frame)

# stop the timer and display FPS information fps.stop()
print("============================")
print("[INFO] Elasped time: {:.2f}".format(fps.elapsed())) print("[INFO]
Approx. FPS: {:.2f}".format(fps.fps()))
# close any open windows cv2.destroyAllWindows()
```

5. IMPLEMENTATION AND TESTING

5.1 Introduction to Python

Python is an interpreted high-level general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), as an object-oriented and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

Guido van Rossum began working on Python in the late 1980s, as a successor to the ABC programming language, and first released it in 1991 40 as Python 0.9.0. Python 2.0 was released in 2000 and introduced new features, such as list comprehensions and a garbage collection system using reference counting. Python 3.0 was released in 2008 and was a major revision of the language that is not completely backward-compatible and much Python 2 code does not run unmodified on Python 3. Python 2 was discontinued with version 2.7.18 in 2020.

Python consistently ranks as one of the most popular programming languages.

5.1.1 Introduction to PyCharm

PyCharm is an integrated development environment (IDE) used in computer programming, specifically for the Python language. It is developed by the Czech company JetBrains (formerly known as IntelliJ). It provides code analysis, a graphical debugger, an integrated unit tester, integration with version control systems(VCSes), and supports web development with Django as well as data science with Anaconda. PyCharm is cross-platform, with Windows, macOS and Linux versions. The Community Edition is released under the Apache License, and there is also Professional Edition with extra features – released under a proprietary license.

Features

- Coding assistance and analysis, with code completion, syntax and error highlighting, linter integration, and quick fixes.
- Project and code navigation: specialized project views, file structure views and quick jumping between files, classes, methods and usages.

- Python refactoring: includes rename, extract method, introduce variable, introduce constant, pull up, push down and others.
- Support for web frameworks: Django, web2py and Flask [professional edition only].
- Integrated Python debugger.
- Integrated unit testing, with line-by-line code coverage 42.
- Google App Engine Python development [professional edition only]
- Version control integration: unified user interface for Mercurial, Git, Subversion, Perforce and CVS with change lists and merge
- Support for scientific tools like matplotlib, numpy and scipy [professional edition only] It competes mainly with a number of other Python-oriented IDEs, including Eclipse's PyDev, and the more broadly focused Komodo IDE.

5.2 PYTHON LIBRARIES

5.2.1 OpenCV

OpenCV (Open Source Computer Vision Library: http://opencv.org) is an open-source library that includes several hundreds of computer vision algorithms. The document describes the socalled OpenCV 2.x API, which is essentially a C++ API, as opposed to the C-based OpenCV 1.x API (C API is deprecated and not tested with "C" compiler since OpenCV 2.4 releases)

OpenCV has a modular structure, which means that the package includes several shared or static libraries. The following modules are available:

- **Core functionality (core)** – a compact module defining basic data structures, including the dense multi-dimensional array Mat and basic functions used by all other modules.
- **Image Processing (imgproc) –** an image processing module that includes linear and nonlinear image filtering, geometrical image transformations (resize, affine and perspective warping, generic table-based remapping), color space conversion, histograms, and so on.
- **Video Analysis (video) –** a video analysis module that includes motion estimation, background subtraction, and object tracking algorithms.

- **Camera Calibration and 3D Reconstruction (calib3d) –** basic multiple-view geometry algorithms, single and stereo camera calibration, object pose estimation, stereo correspondence algorithms, and elements of 3D reconstruction.
- **2D Features Framework (features2d) –** salient feature detectors, descriptors, and descriptor matchers.
- **Object Detection (objdetect) –** detection of objects and instances of the predefined classes (for example, faces, eyes, mugs, people, cars, and so on).
- **High-level GUI (highgui) –** aneasy-to-use interface to simple UI capabilities.
- **Video I/O (videoio) –** an easy-to-use interface to video capturing and video codecs.

5.2.2 OS Module

It is possible to automatically perform many operating system tasks. The OS module in Python provides functions for creating and removing a directory (folder), fetching its contents, changing and identifying the current directory, etc. You first need to import the os module to interact with the underlying operating system. So, import it using the import os statement before using its functions.

5.2.3 Tensorflow

TensorFlow is a free and open-source software library for machine learning. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks. Tensorflow is a symbolic math library based on dataflow and differentiable programming. It is used for both research and production at Google. TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache License 2.0 in 2015.

TensorFlow is Google Brain's second-generation system. Version 1.0.0 was released on February 11, 2017. While the reference implementation runs on single devices, TensorFlow can run on multiple CPUs and GPUs (with optional CUDA and SYCL extensions for generalpurpose computing on graphics processing units). TensorFlow is available on 64bit Linux, macOS, Windows, and mobile computing platforms including Android and iOS.

Its flexible architecture allows for the easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices.

TensorFlow computations are expressed as stateful dataflow graphs. The name TensorFlow derives from the operations that such neural networks perform on multidimensional data arrays, which are referred to as tensors. During the Google I/O Conference in June 2016, Jeff Dean stated that 1,500 repositories on GitHub mentioned TensorFlow, of which only 5 were from Google.

In December 2017, developers from Google, Cisco, RedHat, CoreOS, and CaiCloud introduced Kubeflow at a conference. Kubeflow allows operation and deployment of TensorFlow on Kubernetes.

In March 2018, Google announced TensorFlow.js version 1.0 for machine learning in JavaScript.

In Jan 2019, Google announced TensorFlow 2.0 It became officially available in Sep 2019.

In May 2019, Google announced TensorFlow Graphics for deep learning in computer graphics.

5.2.4 Keras

keras is a minimalist Python library for deep learning that can run on top of Theano or TensorFlow.

It was developed to make implementing deep learning models as fast and easy as possible for research and development.

It runs on Python 2.7 or 3.5 and can seamlessly execute on GPUs and CPUs given the underlying frameworks. It is released under the permissive MIT license.

Keras was developed and maintained by François Chollet, a Google engineer using four guiding principles:

- **Modularity:** A model can be understood as a sequence or a graph alone. All the concerns of a deep learning model are discrete components that can be combined in arbitrary ways.
- **Minimalism:** The library provides just enough to achieve an outcome, no frills and maximizing readability.

- **Extensibility:** New components are intentionally easy to add and use within the framework, intended for researchers to trial and explore new ideas.
- **Python:** No separate model files with custom file formats. Everything is native Python.

5.2.5 Pytesseract

Python-tesseract is an optical character recognition (OCR) tool for python. That is, it will recognize and "read" the text embedded in images.

Python-tesseract is a wrapper for Google's Tesseract-OCR Engine. It is also useful as a standalone invocation script to tesseract, as it can read all image types supported by the Pillow and Leptonica imaging libraries, including jpeg, png, gif, bmp, tiff, and others. Additionally, if used as a script, Python-tesseract will print the recognized text instead of writing it to a file

Tesseract can be used directly via command line, or (for programmers) by using an API to extract printed text from images. It supports a wide variety of languages. Tesseract doesn't have a built-in GUI, but there are several available from the 3rdParty page. External tools, wrappers and training projects for Tesseract are listed under AddOns.

Tesseract can be used in your own project, under the terms of the Apache License 2.0.It has a fully featured API, and can be compiled for a variety of targets including Android and the iPhone. See the 3rdParty and AddOns pages for samples of what has been done with it.

5.3 Jupyter Notebook

Jupyter Notebook (formerly Ipython Notebooks) is a web-based interactive computational environment for creating Jupyter notebook documents. The "notebook" term can colloquially make reference to many different entities, mainly the Jupyter web application, Jupyter Python web server, or Jupyter document format depending on context. A Jupyter Notebook document is a JSON document, following a versioned schema, containing an ordered list of input/output cells which can contain code, text (using Markdown), mathematics, plots and rich media, usually ending with the ".ipynb" extension.

A Jupyter Notebook can be converted to a number of open standard output format through "Download As" in the web interface, via the nbconvert library or "jupyter nbconvert" command line interface in a shell. To simplify isualization of Jupyter notebook documents on the web,

the nbconvert library is provided as a service through NbViewer which can take a URL to any publicly available notebook document, convert it to HTML on the fly and display it to the user.

Jupyter Notebook can connect to many kernels to allow programming in different languages. By default Jupyter Notebook ships with the Ipython kernel. As of the 2.3 release (October 2014), it was 49 Jupyter-compatible kernels for many programming languages, including Python, R, Julia and Haskell.

5.4 Google Firebase

Google Firebase is a Google-backed application development software that enables developers to develop iOS, Android and Web apps. Firebase provides tools for tracking analytics, reporting and fixing app crashes, creating marketing and product experiment.

Firebase offers a number of services, including:

- **Analytics –** Google Analytics for Firebase offers free, unlimited reporting on as many as 500 separate events. Analytics presents data about user behavior in iOS and Android apps, enabling better decision-making about improving performance and app marketing.
- **Authentication –** Firebase Authentication makes it easy for developers to build secure authentication systems and enhances the sign-in and onboarding experience for users. This feature offers a complete identity solution, supporting email and password accounts, phone auth, as well as Google, Facebook, GitHub, Twitter login and more.
- **Cloud messaging** – Firebase Cloud Messaging (FCM) is a cross-platform messaging tool that lets companies reliably receive and deliver messages on iOS, Android and the web at no cost.
- **Realtime database –** the Firebase Realtime Database is a cloud-hosted NoSQL database that enables data to be stored and synced between users in real time. The data is synced across all clients in real time and is still available when an app goes offline.
- **Crashlytics** – Firebase Crashlytics is a real-time crash reporter that helps developers track, prioritize and fix stability issues that reduce the quality of their apps. With crashlytics, developers spend less time organizing and troubleshooting crashes and more time building features for their apps.

- **Performance –** Firebase Performance Monitoring service gives developers insight into the performance characteristics of their iOS and 52 Android apps to help them determine where and when the performance of their apps can be improved.
- **Test lab** – Firebase Test Lab is a cloud-based app-testing infrastructure. With one operation, developers can test their iOS or Android apps across a variety of devices and device configurations. They can see the results, including videos, screenshots and logs, in the Firebase console.

Firebase Realtime Database

Create a new project on Firebase – let's name it BookStoreProject. Once it has been set up, create a Realtime Database by selecting the Create Database option.

When you click on Create Database, you have to specify the location of the database and the security rules. Two rules are available:

- Locked mode, which denies all reads and writes to the database, and
- Test mode, which allows read and write access for a default 30 days (after which all read and writes are denied unless the security rules get updated).

Since we will be using the database for read, write, and edit, we choose test mode. Once that is done, the database is all ready for our usage!

5.5 OUTPUT SCREENS

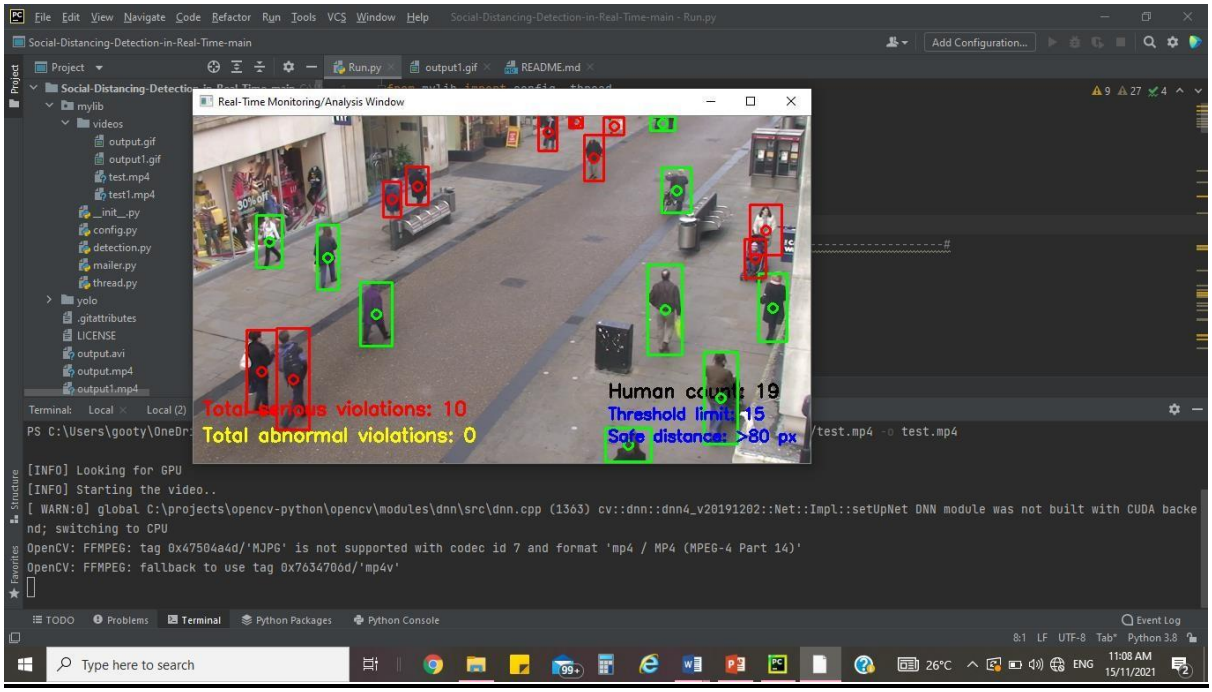

FIG 5.5.1 SHOWING A PERSONS WITHOUT ABNORMAL VIOLATIONS

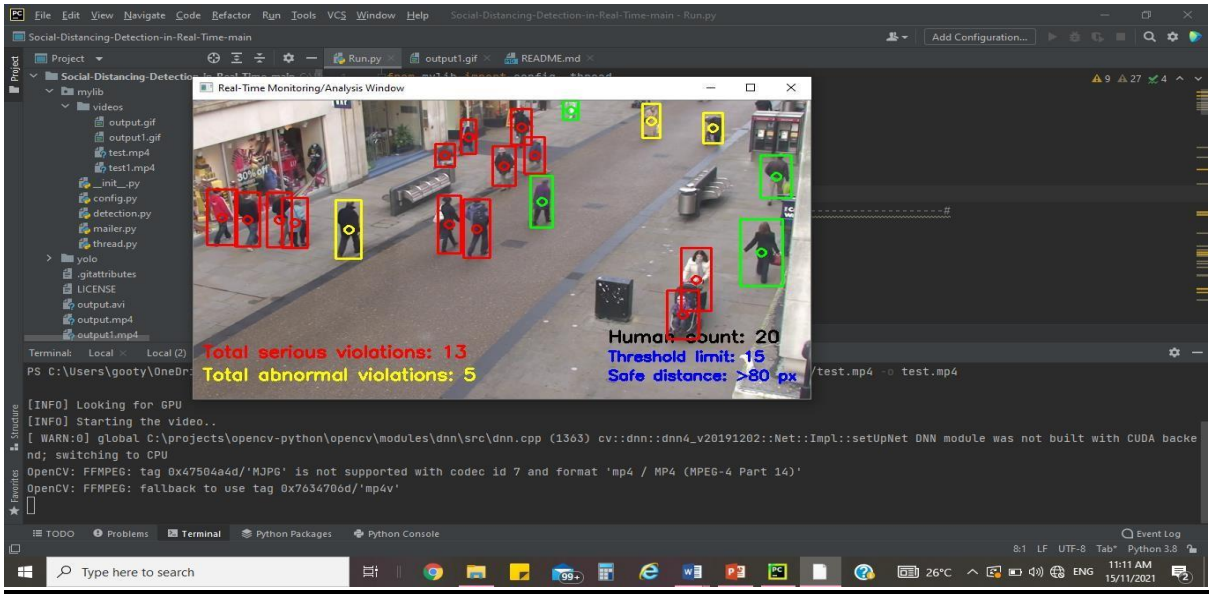

FIG 5.5.2 SHOWING A PERSONS WITH ABNORMAL VIOLATIONS

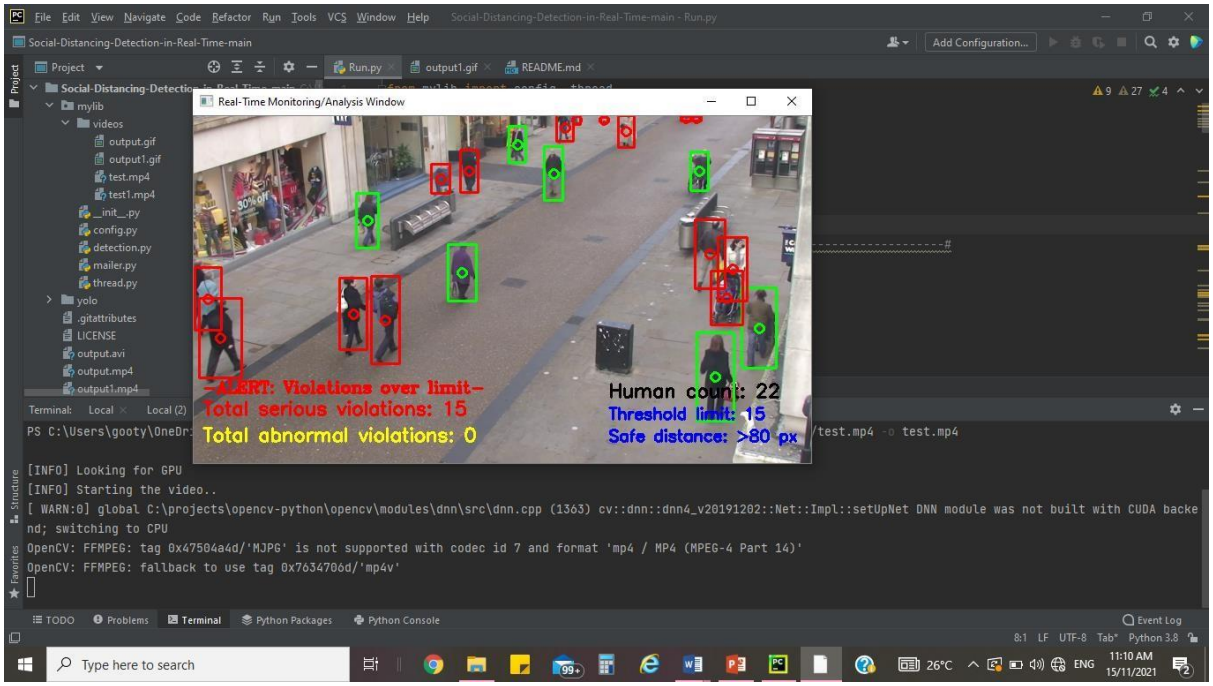

FIG 5.5.3 SHOWING A ALERT VIOLATIONS OVER LIMIT

5.5.1RESULT AND DISCUSSION

The video shows the pedestrian walking on a public street. In this work, the video frame is fixed at a specified angle to the street. The perspectives view of the video frame is transformed into a top-down view for more accurate estimation of distance measurement. Figure 6 shows the social distancing detection in a video frame and the results of the top-down view. The sequences are depicted from top to bottom. The points represent each pedestrian for social distancing detection. The red points represent the pedestrians whose distance with another pedestrian is below the acceptable threshold and the green points represent the pedestrians who keep a safe distance from other pedestrians. However, there are also a number of detection errors are shown in Figure 7. These errors are possibly due to the pedestrians walking too near to another pedestrian until they are overlaid on the camera view. The precision of the distance measurement between pedestrians is also affected by the pedestrian detection algorithm. The YOLO algorithm is also able to detect the half body of the pedestrian as an object by showing the bounding box, the position of the pedestrian corresponds the middle-point of bottom line is estimated based on the bounding box will less precise. To overcome the detection errors, the proposed methodology had been improved by adding a quadrilateral box to observe the appointed region in an image as shown in Figure 8. Hence, only the pedestrians walking within the specified space will be counted for people density measurement.

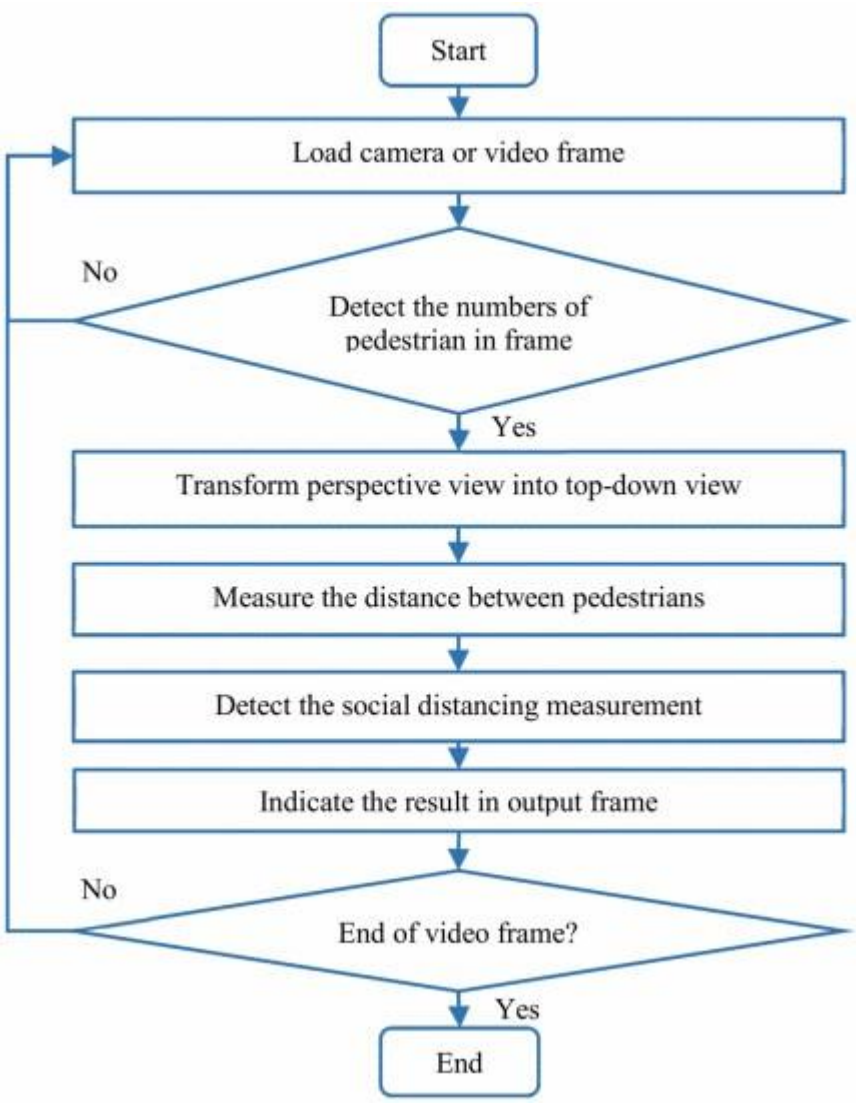

Figure 5 Program flowchart of social distancing detection for each video frame.

6. CONCLUSION

A methodology of social distancing detection tool using a deep learnin g model is proposed. By using computer vision, the distance between people can be estimated and any noncompliant pair of people will be indicated with a red frame and a red line. The proposed method was validated using a video showing pedestrians walking on a street. The visualization results showed that the proposed method is capable to determine the social distancing measures between people which can be further developed for use in other environment such as office, restaurant, and school. Furthermore, the work can be further improved by optimizing the pedestrian detection algorithm, integrating other detection algorithms such as mask detection and human body temperature detection, improving the computing power of the hardware, and calibrating the camera perspective view.

7. REFERENCES

- Adlhoch, C. (2020). https://www.ecdc.europa.eu/sites/default/files/documents/covid-19social-distancing-measuresg-guide-second-update.pdf.
- Adolph C., Amano K., Bang-Jensen B., Fullman N., Wilkerson J. *medRxiv.* 2020 [Google Scholar]
- Ahmad M., Ahmed I., Ullah K., Khan I., Adnan A. *2018 9th IEEE annual ubiquitous computing, electronics mobile communication conference (UEMCON)* 2018. pp. 746–752. [CrossRef] [Google Scholar]
- Ahmad M., Ahmed I., Ullah K., Khan I., Khattak A., Adnan A. *International Journal of Advanced Computer Science and Applications.* 2019;10 doi: 10.14569/IJACSA.2019.0100367. [CrossRef] [Google Scholar]
- Ahmad M., Ahmed I., Khan F.A., Qayum F., Aljuaid H. *International Journal of Distributed Sensor Networks.* 2020;16 1550147720934738. [Google Scholar]
- Ahmed I., Adnan A. 2017. Cluster computing; pp. 1–22. [Google Scholar]
- Ahmed I., Ahmad A., Piccialli F., Sangaiah A.K., Jeon G. *IEEE Internet of Things Journal.* 2018;5:1598–1605. [Google Scholar]
- Ahmed I., Ahmad M., Nawaz M., Haseeb K., Khan S., Jeon G. *Computer Communications.* 2019;147:188–197. [Google Scholar]
- Ahmed I., Din S., Jeon G., Piccialli F. *IEEE Internet of Things Journal.* 2019 [Google Scholar]
- Ahmed I., Ahmad M., Adnan A., Ahmad A., Khan M. *International Journal of Machine Learning and Cybernetics.* 2019:1–12. [Google Scholar]