SOCIAL DISTANCE ANALYZER

"Whatever You Want To Say, Say It From 6 Feet Away"

A Project Report submitted

for the partial fulfilment of the degree of

Bachelor of Technology
In

Information Technology

By

Swarnim Yadav	**1805213059**
Priya Adarsh	**1805213041**
Alankrita Singh	**1805213007**

Under the guidance of

Prof. Girish Chandra **Dr. Tulika Narang**

Department of Computer Science and Engineering

Institute of Engineering and Technology, Lucknow

Dr. A.P.J. Abdul Kalam Technical University, Lucknow, Uttar Pradesh

Contents

Declaration

We hereby declare that this submission is our own work and that, to the best of our belief and knowledge, it contains no material previously published or written by another person or material which to a substantial error has been accepted for the award of any degree or diploma of university or other institute of higher learning, except where the acknowledgement has been made in the text. The project has not been submitted by us at any other institute for requirement of any other degree.

Submitted by: - Date:

1. Swarnim Yadav
   1805213059
   Information Technology

2. Priya Adarsh
   1805213041
   Information Technology

3. Alankrita Singh
   1805213007
   Information Technology

Certificate

This is to certify that the project report entitled "**Social Distance Analyser**" presented by **Swarnim Yadav, Priya Adarsh, Alankrita Singh** in the partial fulfilment for the award of Bachelor of Technology in Computer Science and Engineering, is a record of work carried out by them under my supervision and guidance at the Department of Computer Science and Engineering at Institute of Engineering and Technology, Lucknow. It is also certified that this project has not been submitted at any other Institute for the award of any other degrees to the best of my knowledge.

Prof. Girish Chandra

Department of Computer Science and Engineering

Institute of Engineering and Technology, Lucknow

Dr. Tulika Narang

Department of Computer Science and Engineering

Institute of Engineering and Technology, Lucknow

Acknowledgement

On the very outset of this report, we would like to extend our sincere & heartfelt gratitude towards all luminaries who have helped us directly or indirectly in this project.

We are ineffably thankful and pay our sincere gratitude to our supervisors **Prof. Girish Chandra**, CSE department and **Dr. Tulika Narang**, CSE department for their helpful information, keen interest, valuable guidance, constructive criticism, and insightful advice at various stages of our training period.

We would like to thank **Dr. D. S. Yadav**, the head of department, **Dr. Promila Bahadur**, the project coordinator and the project monitoring committee members for delivering the guidelines and organizing the online presentations composedly with time and ease.

Finally, we would like to wind up by paying our heartfelt thanks to our supportive family and friends for motivating us and putting out their idea for the project.

Abstract

In this report, a model is proposed that aims to track social distancing among people by calculating the physical distance among every pair of pedestrians. Based on this calculated distance, the model classifies them as violator that in turn helps to mitigate the impact of the pandemic in the area of interest. The proposed model uses a deep learning object detection algorithm, YOLOv3 that helps in detecting the pedestrians and the detection results helps to evaluate the distance between the pedestrians. The distance between any pair of pedestrians in the display is estimated by using Euclidean distance and the pair which is non-compliant with the set parameters is indicated with a red frame and the pair who is on the verge of violating the parameters of social distancing is indicated with a yellow frame and the pair which is well within the set parameters of the same is indicated with a green frame. The proposed model was corroborated on a pre-recorded video. The result exhibits that the model can be used for monitoring whether the social distancing is being followed or not. The model can be further developed for real time monitoring of the pedestrians by providing camera URL.

List of Figures

List Of Tables

List Of Abbreviations

YOLO	You only look once
ROI	Region of Interest
SDA	Social Distance Analyzer
AI	Artificial Intelligence
DL	Deep Learning

CHAPTER-1

Introduction

In today's era, digital technologies such as surveillance of population, contact tracking and evaluation of interventions based on mobility data, case identification etc are being utilized to support the public-health response to COVID-19 worldwide [1]. In future, the public health is likely to shift towards these digital technologies and these will be used to strengthen pandemic management and help in preparing for COVID-19 and other contagious diseases.

In the time of outbreaks such as COVID-19, research and development on new methods and technologies or updating existing methods is done to empower the core capacities [2]. A deeper understanding of the epidemic suggests that in order to reduce the risk of virus spread can be minimized by increasing the physical distance or avoiding the physical contact among the people. Effective technologies and comprehensive tools can be used to implement social distancing. If social distancing is enforced at the initial stages, it can perform a vital role in overcoming the spread of the virus and preventing the peak of the pandemic.

Artificial Intelligence, a core technology that came into existence during the fourth industrial revolution, plays an important non-medical intervention in overcoming the current global health crisis. AI helps in building next-generation epidemic preparedness, and in moving towards a flexible recovery. In the area of computer vision and machine learning, different algorithms have been developed for object

detection. The same methods can also be used to detect the physical distance between people.

The following points summarize the main components of this approach:

1. For object detection, there has been a shift towards using deep learning technologies. the same technique is used for the purpose of human detection.
2. A social distancing detection model is developed that will detect the distance between the pedestrians to analyse safe distance.
3. Evaluating of the result of classification by analysing real-time video streams by providing the camera [3].

The main purpose of this project is to track the social distance by providing a deep learning platform. To accomplish this objective of social distance monitoring, we will be using a framework, YOLOv3 to identify humans in video sequences. The proposed system is favourable because it is automated and helps in reducing the manpower required to physically inspect the areas where the people may not be following social distancing norms.

Fig 1.1 Social distance

CHAPTER-2

Literature Review

This chapter mentions about the sources from where the ideas and references have been drawn to come up with this project.

2.1 RELATED WORKS

2.1.1 Works on social distance analyser

Numerous amounts of work have been done in the area of detection for objects and person using various techniques of deep learning. Real-time object detection algorithms that use model of Convolutional Neural Networks (CNN) like You Only Look Once (YOLO) and Region based Convolutional Neural Networks (RCNN) for detecting multiple classes in several regions was developed by G.V. Shalini et al[2].

Latest improvements in the area of deep learning facilitates the object detection in a more effective way. Researchers frequently utilize these methods to track physical distancing among pedestrians in the dynamic frames.

2.2 STATE OF THE ART

2.2.1 Real time object detection using deep learning and OpenCV

The work proposed by Chandan G, Ayush Jain, Harsh Jain, Mohana, aimed at achieving detection of objects in real time. A real time video that was taken from a webcam was given as an input for the object detection. The input video was pre- processed. The output of pre-processing was then sent to the module for detection where a series of operations were carried out for generating the result. The work will help in the future aspect for real time object detection [5].

2.2.2 Person detection for social distancing and safety violation alert

The work by Afiq Harith Ahamad, Norliza Zaini, Mohd Fuad Abdul Latip, proposed a way for the implementation of two features using python language and OpenCV library. The first feature was to detect social distancing violations and the second feature was to detect violations if a person enters in restricted areas. When different tests were performed on this approach, the results suggested that the object detection model was not having high accuracy and was facing issues in detecting people correctly in the public areas. The work helped us in getting to know about the YOLOv3 algorithm and with this; it also helped us in defining the region of interest parameter [6].

2.2.3 Deep learning based safe social distancing and face mask detection in public areas

The work by Shashi Yadav proposed a system that made use of raspberry pi4, computer vision and Mobile Net V2 that helped the local police officials in the surveillance of public areas. In this system the camera feed real time videos and detects whether a pedestrian is equipped with a mask or not. The work will help in implementing our proposed model using raspberry pi in future [7].

2.2.4 Object detection algorithms for video surveillance

The work proposed by Apoorva Raghunandan; Mohana; Pakala Raghav; H.V. Ravish Aradhya, implements object detection, colour detection and skin detection in a video running in real time. It also comprises of background and foreground image processing model for object detection in real time. The work helped in determing the method to video as an input for the detection algorithm [8].

CHAPTER-3

Methodology

This chapter describes the steps that have been taken to construct and implement the model and analyse the final result.

3.1 WORKFLOW

In this work, we have used a Deep Learning model for object detection to detect people along with an algorithm for classifying the social distance based on the provided input video. Initially, we have used an open-source object detection algorithm, YOLO for the detection of the pedestrians in the input video frame. Detector may detect several objects like pedestrians, trees, cars, etc but we have used only the pedestrian class and ignored the rest classes in this proposed model. The bounding boxes are fitted for each pedestrian that is detected and the data of these detected pedestrian is used for measuring the distance between them.

The tasks were divided in the following categories-

1.	Environment Configuration
2.	Data Acquisition
3.	Applying YOLO algorithm
4.	Generating Result

Table 1 Task Categorization

3.1.1 **Environment Configuration**

The program is developed using python3 language. The model is implemented in Jupyter notebook.

3.1.2 **Data Acquisition**

Pre-recorded videos were downloaded from the internet in order to train the model. We have also recorded some videos with the help of smart phone to test the model.

3.1.3 **Applying YOLO Algorithm**

On providing the input, the YOLO algorithm divides the in input into grids, say SxS. The features are extracted from each grid. The output contains the bounding boxes along with the confidence scores for the classes that are predicted in the bounding boxes. The model identifies the people using the obtained bounding box information.

The distance between each pair of bounding boxes is determined by measuring the Euclidean distance between the bounding box's centroid. In order to estimate the social distance violations between the people, an approximate threshold limit has been determined. The threshold limit has been set to check whether there has been a breach in the minimum social distance threshold.

The steps taken in YOLO algorithm are –

- Video is taken as an input
- The input frame is divided into grids
- Image classification in applied on each grid.
- Bounding boxes and class probabilities are predicted to identify the type of object.

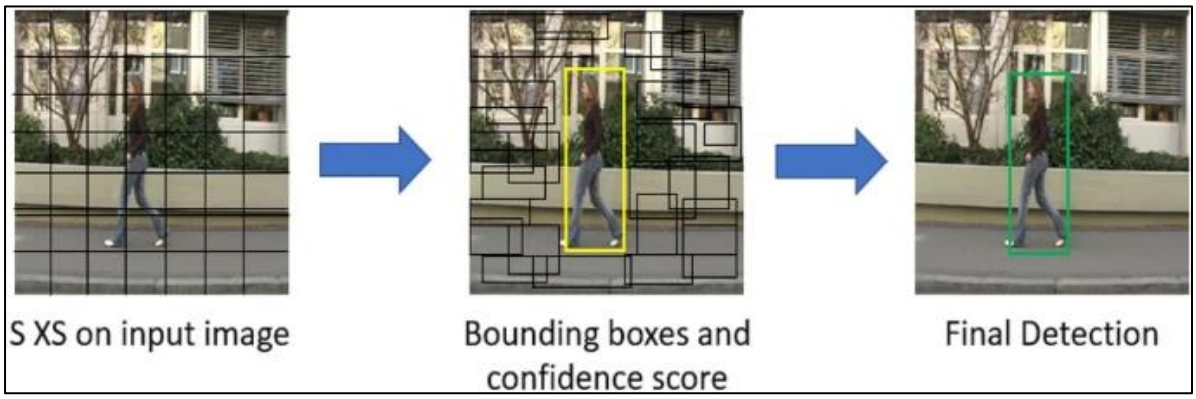


Fig. 3.1 YOLO algorithm depiction

3.1.4 **Generating Result**

To get the output videos, we have run the code in Window PowerShell.

3.2 TECHNOLOGY STACK

To implement this application, following technologies have been used

3.2.1 **Python**

We have used Python version3 as the centre programming language for this project as it has support for Artificial Intelligence and various supporting libraries.

3.2.2 **OpenCV**

OpenCV is an open–source library which is used for machine learning, image processing and computer vision. By using this, we can easily detect objects, humans, etc. We have used cv2 to grab the video file if reference path is provided otherwise, to grab reference to the camera.

3.2.3 **NumPy**

This library consists of multi-dimensional arrays and matrix data structures. With the help of Numpy, we are able to achieve array operations in less space complexity. In this project, we have used Numpy library to extract the centroid details.

3.2.4 **Argparse**

With the help of argparse, we can create a command-line environment in order to improve the interaction. It generates help methods and issues if wrong arguments are provided.

3.2.5 **Scipy**

With the help of this library, we have imported the distance computations.

Program flowchart for social distancing detection in each video frame:

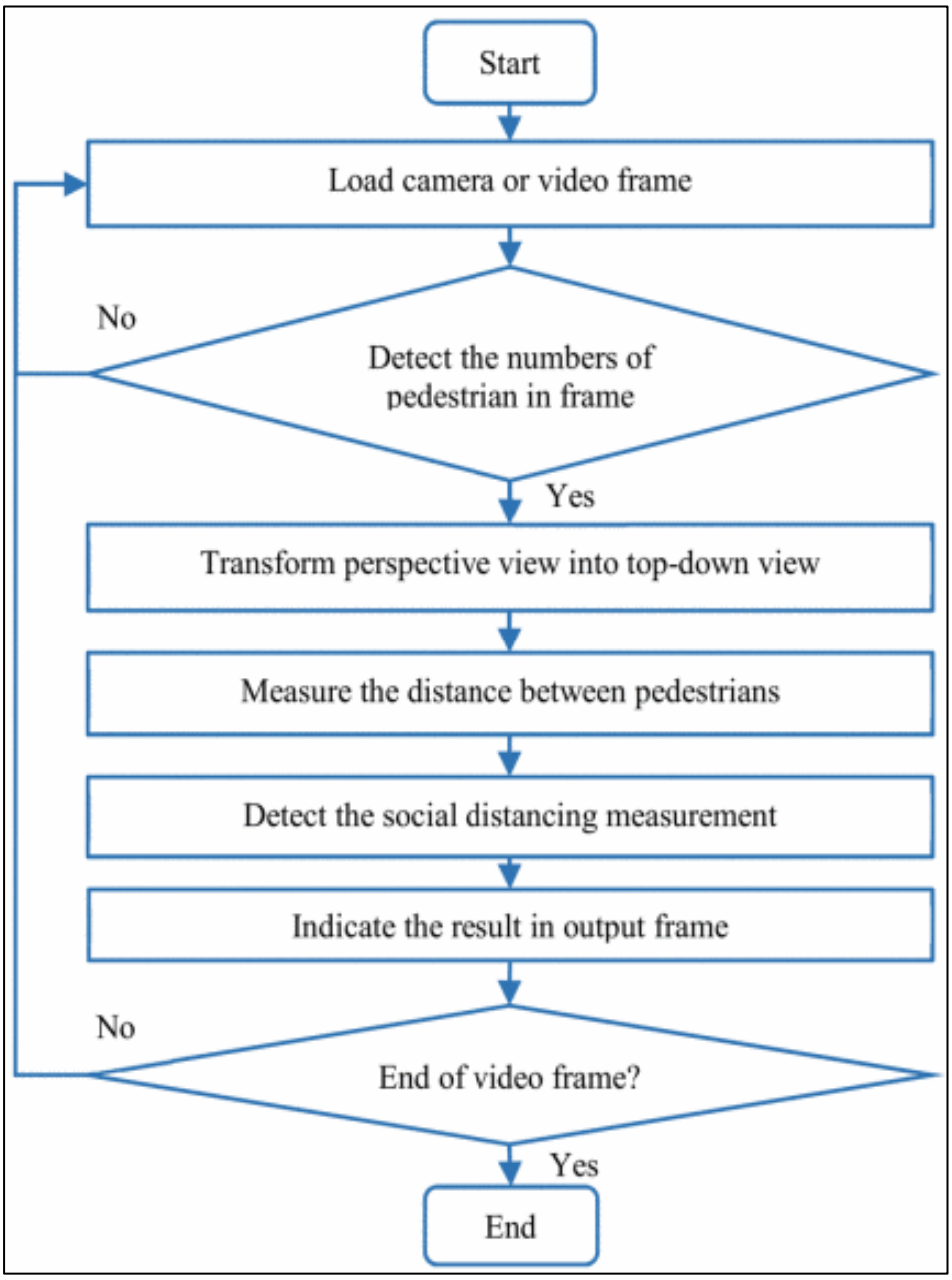


Fig. 3.2 Flowchart of SDA

CHAPTER-4

Experimental Results

In this Chapter, we have provided the detailed description of each output snapshot and code snippet along with all the use cases of the project.

Fig 4.1 Threshold limit

- The above figure shows the snapshot of output video result on the pre-recorded input video. In the bottom right corner of the snippet (**Highlighted with a border**) are the parameters on which we are calculating distance between pedestrians and displaying them with a red frame when they are under the set parameter of safe distance i.e., 80px, and if they are beyond the set parameter of safe distance i.e., 80px they are represented with a green frame.

```
# To count the total number of people (True/False).
People_Counter = True
# Threading ON/OFF. Please refer 'mylib>thread.py'.
Thread = True
# Set the threshold value for total violations limit.
Threshold = 12
# Enter the ip camera url (e.g., url = 'http://191.138.0.100:8040/video');
# Set url = 0 for webcam.
url = '0'
# Turn ON/OFF the email alert feature.
ALERT = False
# Set mail to receive the real-time alerts. E.g., 'xxx@gmail.com'.
MAIL = 'swarnimky88@gmail.com'
# 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
#==============================================================================
#==============================================================================
```

Fig 4.2 Code snippet for threshold value

- The above code snippet has a code for the parameters that has been written in config module which here states about the parameters of threshold that has been set to 12 here which signifies that atmost 12 violations can be tolerated with a parameter of maximum distance of 80px which has been written at the bottom in the above snippet.

```
    # 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)
```

Fig 4.3 Code snippet for parameters

- The above code snippet accepts the arguments for safe distance in the main python file that has been sent from config module as MAX_DISTANCE. This code snippet also accepts the threshold value whose value can be set by the user and is also sent from config module as a argument to the above function.

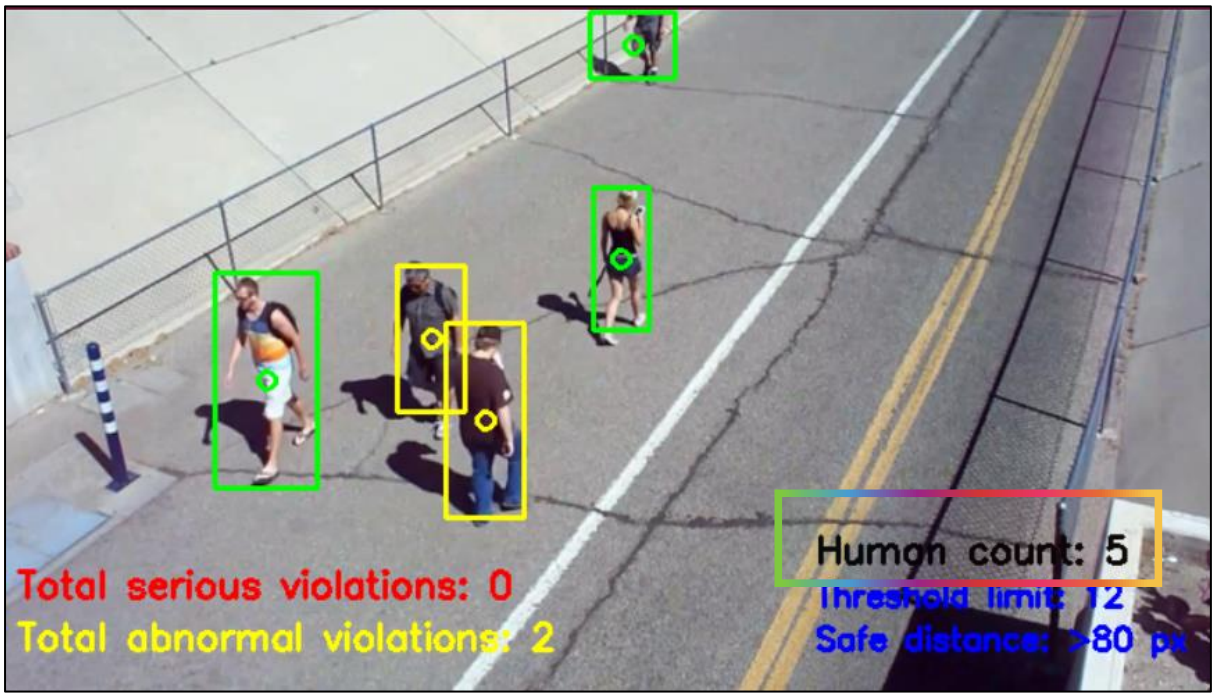
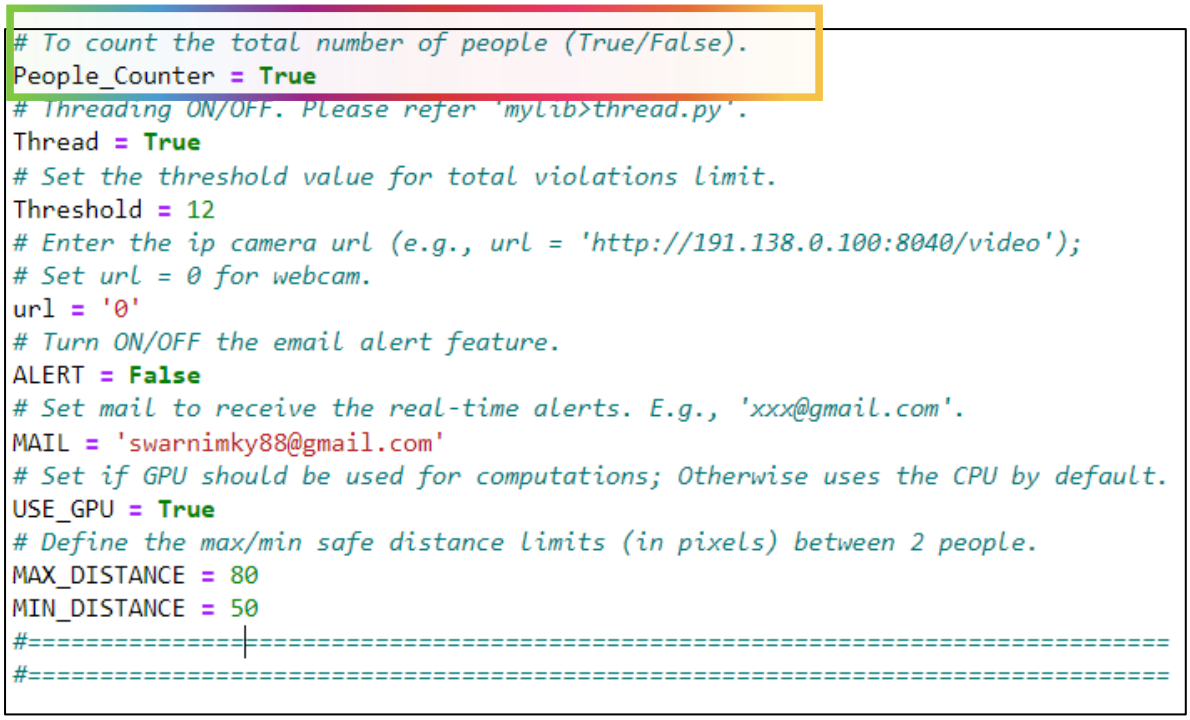


Fig 4.4 UseCase for human count

- As one of the usecase of this project is to calculate the human count in the video. In the above snapshot of the video as we can see total human count is 5 which is perfectly accurate as per the video snapshot provided above. This feature can be used to count no. of people in a gathering, to avoid stampede, to be in compliance of the curfew act, to take attendance of no. of people and many more such cases.



```
# To count the total number of people (True/False).
People_Counter = True
# Threading ON/OFF. Please refer 'mylib>thread.py'.
Thread = True
# Set the threshold value for total violations limit.
Threshold = 12
# Enter the ip camera url (e.g., url = 'http://191.138.0.100:8040/video');
# Set url = 0 for webcam.
url = '0'
# Turn ON/OFF the email alert feature.
ALERT = False
# Set mail to receive the real-time alerts. E.g., 'xxx@gmail.com'.
MAIL = 'swarnimky88@gmail.com'
# 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
#===============================================================================
#===============================================================================
```

Fig 4.5 Code snippet for turning on human count feature

- The Human count feature is completely optional and can be turn off/on in the config module under People counter variable as true or false. The has value has been imported to the detection python file under mylib folder.

```python
# 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")
```

Fig 4.6.1 Count snippet for human count function

```python
                # 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
```

Fig 4.6.2 Code snippet for human count function

- The above code is written under mylib -> detection python file to detect people and count the no. of people.

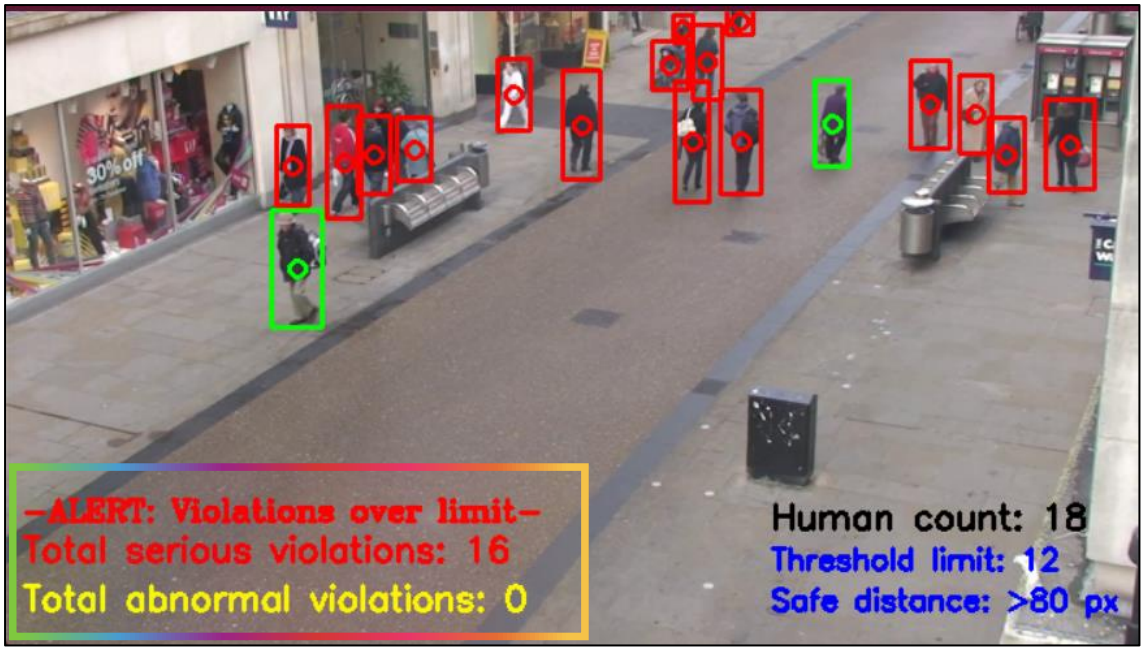


Fig 4.7 Alert function and abnormal violations

- The above snippet of the output shows the serious violations (as red frame) , abnormal violations(as yellow frame)and the safe zone(as green frame).The set parameters of the serious and abnormal violations are mentioned in the bottom left corner of the output snippet.

```
    # 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)
```

Fig 4.8 Code snippet for calculation of violations

- The above code snippet states the code for the total no. of social distancing violations on the output frame.

Fig 4.9 Overlapping camera view

- One of the use cases of our project is to identify the violations on a prerecorded video to defeat the aspect of overlapping video angles. We enhance the project by restricting the area of visualization (Region of interest) which is set by the user itself as per its discretion.

Fig 4.10 Region of interest

- To achieve the restricted field of visualization (Region of interest) which is set by the user, the user draws four points(red) which is termed as region of interest (ROI) here.

Fig 4.11 Unit length for ROI

- In the above snapshot the three blue dots which are parallel to the lines drawn for the region of interest indicates unit length in the real world.

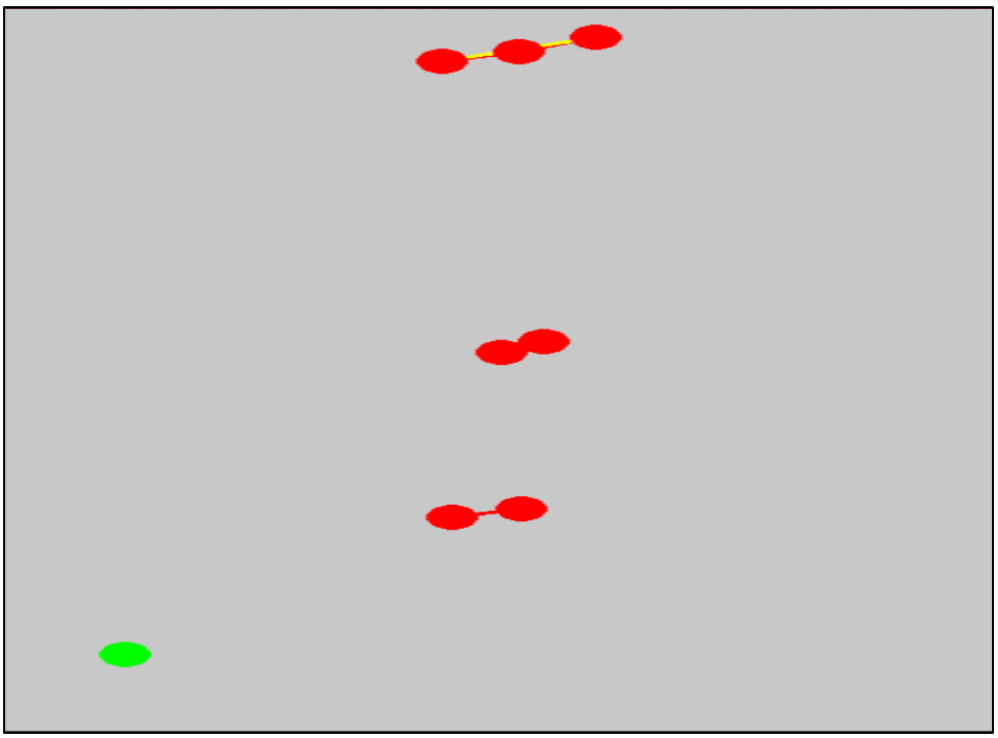


Fig 4.12 Bird eye view

- The above snapshot represents the bird eye view which is actually the output video frame received by providing the ROI and unit length points.

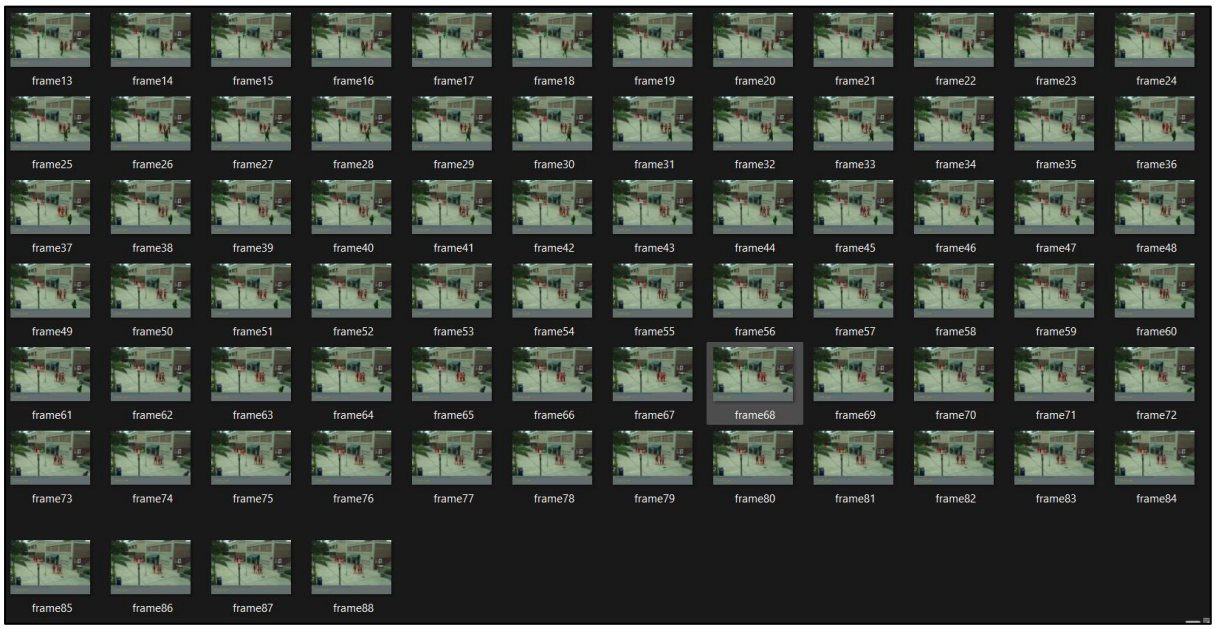


Fig 4.13 Output frames per second

- The above snapshot is of the output folder of the project which captures and store the output frames per second, and it continues to store the output frames until the process is killed.

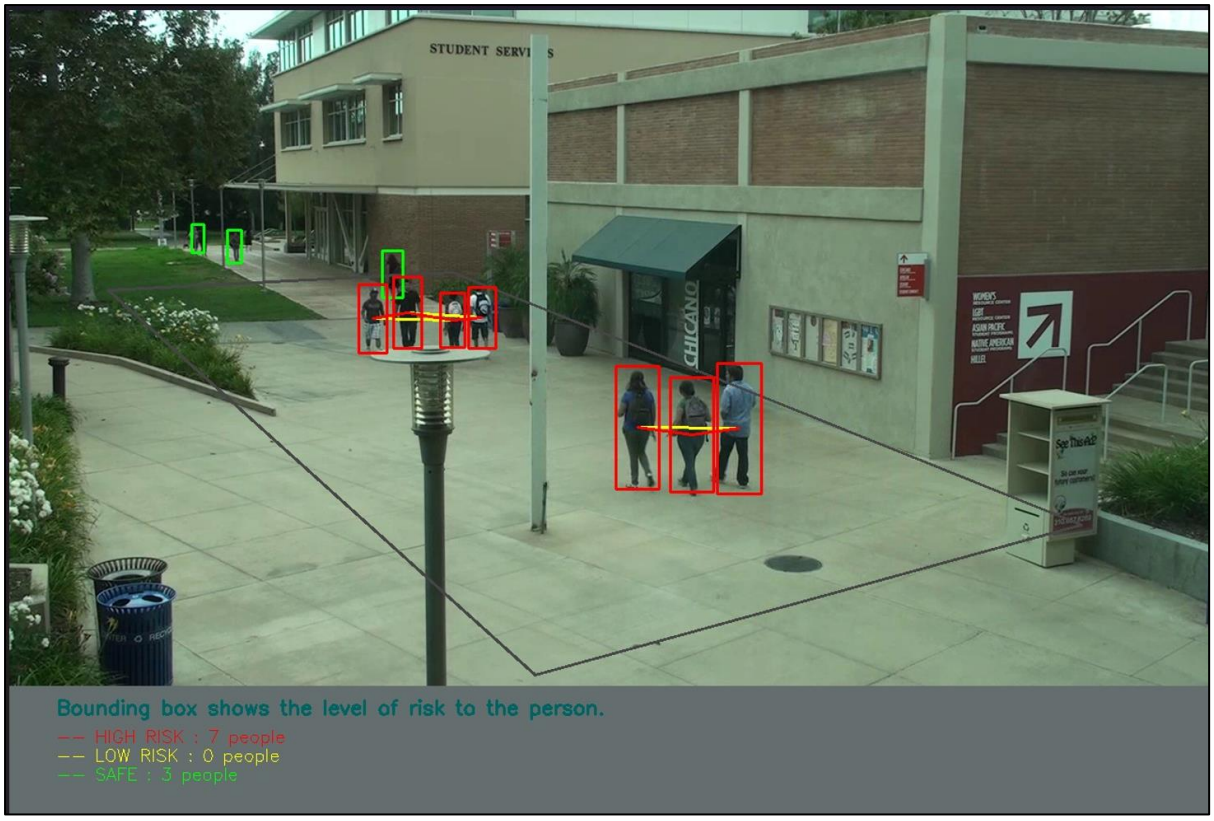


Fig 4.14 Safe and unsafe distance

- In the above snapshot is captured from the output folder of project, the line determines that the person 1 and person 3 are at a reasonably safe distance from each other. The red lines represent that person 1, person 2 and person 2, person 3 are at an unsafe distance from each other.

```
# To count the total number of people (True/False).
People_Counter = True
# Threading ON/OFF. Please refer 'mylib>thread.py'.
Thread = True
# Set the threshold value for total violations limit.
Threshold = 12
# Enter the ip camera url (e.g., url = 'http://191.138.0.100:8040/video');
# Set url = 0 for webcam.
url = '0'
# Turn ON/OFF the email alert feature.
ALERT = False
# Set mail to receive the real-time alerts. E.g., 'xxx@gmail.com'.
MAIL = 'swarnimky88@gmail.com'
# 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
#================================================================================
#================================================================================
```

Fig 4.15 Code snippet for email alert feature

- The above code snippet is taken from the config file in which we have provided the option for turning on the alert function to true/false, whose value is further exported to the mailer file and used as per the requirements.

```python
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 = "swarnimky88@gmail.com"
        # 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('swarnimky88.gmail.com', self.PORT)

    def send(self, mail):
        self.server = smtplib.SMTP_SSL('swarnimky88.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()
```

Fig 4.16 Email alert function

- In the above code snippet, the script initiates the email alert function. This contains two functions ,one which contain the sender email along with its authorization provided 2 step verification is not turned on .

```python
#------------------------------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
```

Fig 4.17 Email receiver function

- The above code snippet has a function which ensures that the administration will receive an email in case if the violations are beyond the set parameters of threshold.

```
# To count the total number of people (True/False).
People_Counter = True
# Threading ON/OFF. Please refer 'mylib>thread.py'.
Thread = True
# Set the threshold value for total violations limit.
Threshold = 12
# Enter the ip camera url (e.g., url = 'http://191.138.0.100:8040/video');
# Set url = 0 for webcam.
url = '0'
# Turn ON/OFF the email alert feature.
ALERT = False
# Set mail to receive the real-time alerts. E.g., 'xxx@gmail.com'.
MAIL = 'swarnimky88@gmail.com'
# 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
#===============================================================================
#===============================================================================
```

Fig 4.18 Code snippet for providing camera URL

- As a future aspect of our project, we aim to bring much more of the scalability. As the result of which we aspire this project to be a real time analyser this we have put forward our step and provided the camera URL variable in which we can provide our camera URL for the inference.

```
# 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"])
```

Fig 4.19 Code snippet for grabbing camera URL/Video fie

- In the above code snippet, we have used if-else statement in which if the video path reference is not supplied then the code will grab the reference to the camera otherwise it will grab a reference to the video file.

```
# To count the total number of people (True/False).
People_Counter = True
# Threading ON/OFF. Please refer 'mylib>thread.py'.
Thread = True
# Set the threshold value for total violations limit.
Threshold = 12
# Enter the ip camera url (e.g., url = 'http://191.138.0.100:8040/video');
# Set url = 0 for webcam.
url = '0'
# Turn ON/OFF the email alert feature.
ALERT = False
# Set mail to receive the real-time alerts. E.g., 'xxx@gmail.com'.
MAIL = 'swarnimky88@gmail.com'
# 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
#===============+=======================================================================
#=======================================================================================
```

Fig 4.20 Threading parameter

- The above code snippet provides user the flexibility to turn on the threading, if user experience lag or delay in the stream of the output video frame, by assigning value to thread parameter as true or false.

```
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
```

Fig 4.21 Threading function

- The above code snippet initiates threading class, which read the frames as soon as they are available and discard any unprocessed frames. This approach removes OpenCV's internal buffer and reduces the frame lag. This function stores the frame in queue instead of the buffer and fetch frames from the queue one by one.

Fig 4.22 Analysis in college campus

- The above snapshot was taken around January 2022 in our college campus during the third wave of covid-19. At this time, we were in our initial phase of the project and we took the opportunity to capture this video and later we performed an analysis on this video so that we can closely relate and associate with the situation of how serious and deadly surrounding we create for ourselves as well as for others.

CHAPTER-5

CONCLUSION

This project presents a deep learning based intelligent system that will track people and identify whether social distancing is being followed as per the predefined norms. We have used the pretrained YOLOv3 exemplar for the detection of human. Our model of detection gives the information of violation through bounding box, containing centroid coordinates information. The specific algorithm implemented on bounding boxes, distinguished between the safe and unsafe conditions, respectively, marking as green and red the bounding box for detected persons. We have used Euclidean distance to measure the distances between two pairs of detected bounding boxes through their centroid. To classify the social distance violations, we have used numerous parameters, such as max_distance between a pair of pedestrians, threshold value, according to which model assigns serious and abnormal violation flags and trigger an alert function if threshold value is hit. We have furthermore added various usecases to our model such as human count which is useful for preventing stampede, taking attendance, tackling contagious diseases, alert mail which is sent to admin whenever the serious violation has crosses the parameter of threshold value, modifying ROI as a usecase is useful whenever we have two overlapping video angles input , we can manually set out our region of interest and run our analyser on that particular ROI only. The proposed technique has achieved promising results in identifying people walking too close and violating social distancing norms.

FUTURE WORKS

The proposed model may be executed in a distributed video surveillance system. It will definitely be an effective solution for the administration to analyse the compliance of pedestrians with physical distancing. In the coming times, this model can also be put into effect on mobile cameras. For eg.,established on drone system and it can simply operate and capture fast and dynamic actions of the captured objects from various angles.

In our project we have made space for real time monitoring of the pedestrians by providing camera URL variable in config file, in which if we provide secure url for the camera the main function would run inference on the reference of the camera otherwise it would run inference on the reference of the stored video file.

References

[1] Jobie Budd, Benjamin S. Miller, Erin M. Manning, Vasileios Lampos, Mengdie Zhuang, Michael Edelstein, Geraint Rees, Vincent C. Emery, Molly M.Stevens, NeilKeegan, Michael J. Short, Deenan Pillay, Ed Manley, Ingemar J. Cox, David Heymann, Anne M. Johnson & Rachel A. McKendry, "Digital technologies in the public-health response to COVID-19"

https://www.nature.com/articles/s41591-020-1011-4#:~:text=Digital%20technologies%20are%20being%20harnessed,and%20communication%20with%20the%20public

[Accessed 7 August 2020]

[2] G V Shalini, "Social Distancing Analyzer Using Computer Vision and Deep Learning"

https://iopscience.iop.org/article/10.1088/1742-6596/1916/1/012039/pdf

[Accessed 23 February 2022]

[3] Yew Cheong Hou; Mohd Zafri Baharuddin; Salman Yussof; Sumayyah Dzulkifly, "Social Distancing Detection with Deep Learning Model"

https://ieeexplore.ieee.org/abstract/document/9243478

[Accessed 11 November 2020]

[4] Riya Chaudhri; Lokesh M. Girpunje; Arpita Patra; Aniket Sagar "A review on Social Distance and Face mask detection"

https://www.jetir.org/view?paper=JETIR2104023
[Accessed 8 April 2021]

[5] G Chandan; Ayush Jain; Harsh Jain; Mohana, "Real time object detection and tracking using deep learning and openCV"

https://ieeexplore.ieee.org/document/8597266?denied=
[Accessed 2018]

[6] Afiq harith Ahamad; Norliza Zaini; M F Abdul Latip, "Person Detection for Social Distancing and Safety Violation Alert based on Segmented ROI"

https://www.researchgate.net/publication/347021016_Person_Detection_for_Social_Distancing_and_Safety_Violation_Alert_based_on_Segmented_ROI
[Accessed August 2020]

[7] Shashi Yadav, "Deep Learning based Safe Social Distancing and Face Mask Detection in Public Areas for COVID-19 Safety Guidelines Adherence"

https://www.researchgate.net/publication/343346690_Deep_Learning_based_Safe_Social_Distancing_and_Face_Mask_Detection_in_Public_Areas_for_COVID-19_Safety_Guidelines_Adherence
[Accessed July 2020]

[8] Apoorva Raghunandan; Mohana; Pakala Raghav; H. V. Ravish Aradhya, “Object Detection Algorithms for Video Surveillance Applications”

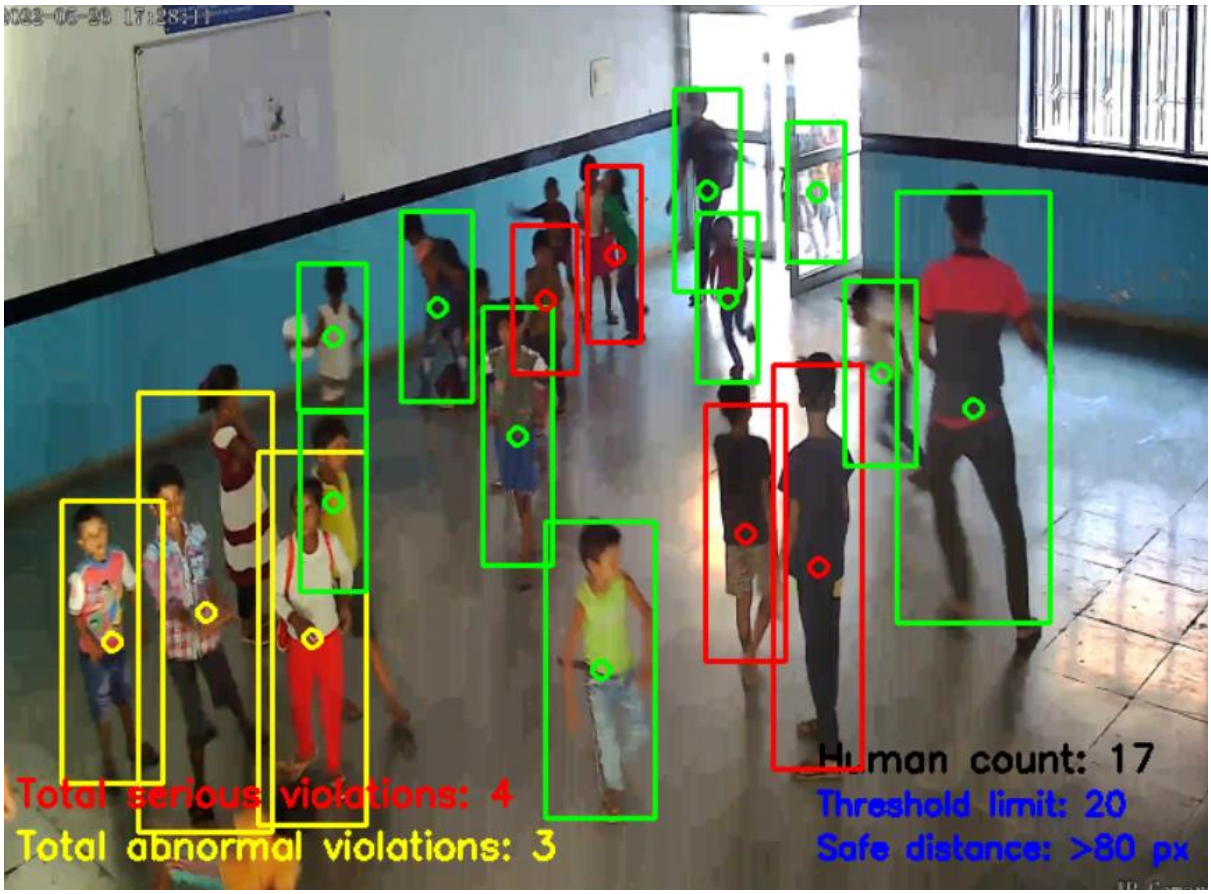
https://ieeexplore.ieee.org/document/8524461
[Accessed 8 November 2018]

Annexure

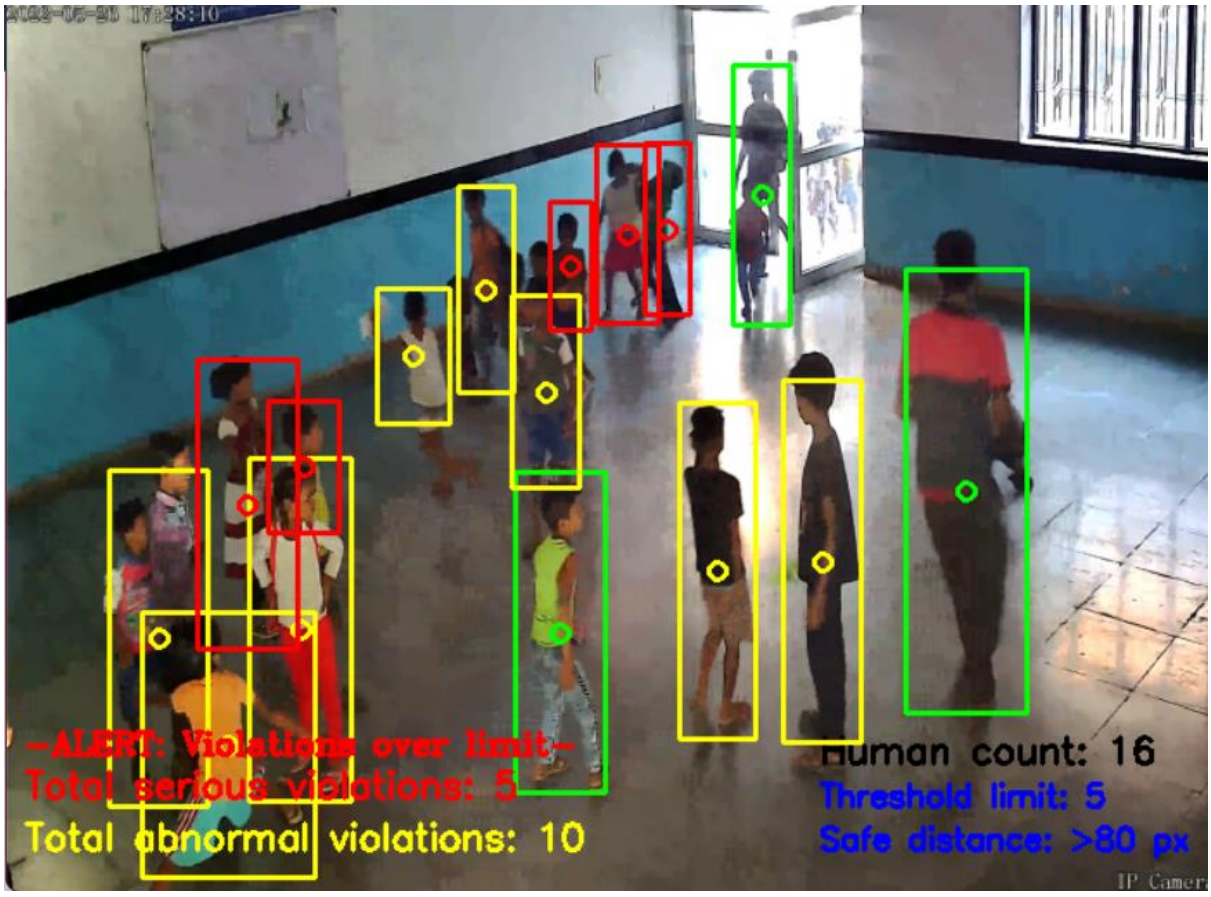
Other Experimental Results

This section consists of the snapshots of the experimental results obtained from the video captured in the campus of Institute of Engineering and Technology, Lucknow.


Total serious violations: 12
Total abnormal violations: 0
Human count: 17
Threshold limit: 20
Safe distance: >80 px


Total serious violations: 4
Total abnormal violations: 3
Human count: 17
Threshold limit: 20
Safe distance: >80 px


Total serious violations: 4
Total abnormal violations: 6
Human count: 14
Threshold limit: 20
Safe distance: >80 px


-ALERT: Violations over limit-
Total serious violations: 5
Total abnormal violations: 10
Human count: 16
Threshold limit: 5
Safe distance: >80 px