SOCIAL DISTANCE DETECTION

A
Project Report

Submitted in partial fulfilment of the
Requirements for the award of the Degree of

BACHELOR OF ENGINEERING

IN

INFORMATION TECHNOLOGY

By

K. Raj Kumar

1602-17-737-027

A. Vishwa Prabhath

1602-17-737-058

Under the guidance of

R. DHARMA REDDY

ASSISTANT PROFESSOR

Department of Information Technology

Vasavi College of Engineering (Autonomous)

ACCREDITED BY NAAC WITH 'A++' GRADE
(Affiliated to Osmania University)

Ibrahimbagh, Hyderabad-31

Vasavi College of Engineering (Autonomous)

ACCREDITED BY NAAC WITH 'A++' GRADE

(Affiliated to Osmania University)

Hyderabad-500 031

Department of Information Technology

DECLARATION BY THE CANDIDATE

I, **K. Raj Kumar,** bearing hall ticket number, **1602-17-737-027**, hereby declare that the project report entitled **Social Distance Detection** under the guidance of **Dharma reddy,** Assistant Professor, Department of Information Technology, Vasavi College of Engineering, Hyderabad, is submitted in partial fulfilment of the requirement for the award of the degree of **Bachelor of Engineering** in **Information Technology**

This is a record of bonafide work carried out by me and the results embodied in this project report have not been submitted to any other university or institute for the award of any other degree or diploma.

K. Raj Kumar

1602-17-737-027

Vasavi College of Engineering (Autonomous)

ACCREDITED BY NAAC WITH 'A++' GRADE

(Affiliated to Osmania University)

Hyderabad-500 031

Department of Information Technology

DECLARATION BY THE CANDIDATE

I, **A. Vishwa Prabhath** bearing hall ticket number, **1602-17-737-058**, hereby declare that the project report entitled **Social Distance Detection** under the guidance of **Dharma reddy** ,Assistant Professor, Department of Information Technology, Vasavi College of Engineering, Hyderabad, is submitted in partial fulfilment of the requirement for the award of the degree of **Bachelor of Engineering** in **Information Technology**.

This is a record of bonafide work carried out by me and the results embodied in this project report have not been submitted to any other university or institute for the award of any other degree or diploma.

A. Vishwa Prabhath

1602-17-737-058

Vasavi College of Engineering (Autonomous)

ACCREDITED BY NAAC WITH 'A++' GRADE

(Affiliated to Osmania University)

Hyderabad-500 031

Department of Information Technology

BONAFIDE CERTIFICATE

This is to certify that the project entitled **Social Distance Detection** being submitted by **K. Raj Kumar, A. Vishwa Prabhath** bearing 1602-17-737-027,1602-17-737-058 in partial fulfilment of the requirements for the award of the degree of Bachelor of Engineering in Information Technology is a record of bonafide work carried out by him/her under my guidance.

Dharma Reddy
Assistant Professor
Internal Guide

Dr. K. Ram Mohan rao
HOD , IT

ACKNOWLEDGEMENT

It is my privilege to acknowledge with deep sense of gratitude and devotion for keen personal interest and invaluable guidance rendered by our Project Guide **R.Dharma Reddy,** Assistant Professor, Department of Information Technology, Vasavi College of Engineering

I take the opportunity to express my thanks to **Dr. K. Ram Mohan Rao** Professor & Head of IT Department, VCE for his valuable suggestions and moral support. I am grateful to my Principal, Dr S.V. Ramana, Vasavi College of Engineering, for his cooperation and encouragement. Finally, I also thank all the staff members, faculty of Dept. of IT, VCE, and my friends, who with their valuable suggestions and support, directly or indirectly helped me in completing this project work.

ABSTRACT

Social distancing, also called "physical distancing," means keeping a safe space between yourself and other people who are not from your household. To practice social or physical distancing, stay at least 6 feet (about 2 arm lengths) from other people who are not from your household in both indoor and outdoor spaces. COVID-19 spreads mainly among people who are in close contact (within about 6 feet) for a prolonged period. Spread happens when an infected person coughs, sneezes, or talks, and droplets from their mouth or nose are launched into the air and land in the mouths or noses of people nearby. The droplets can also be inhaled into the lungs. Recent studies indicate that people who are infected but do not have symptoms likely also play a role in the spread of COVID-19. Since people can spread the virus before they know they are sick, it is important to stay at least 6 feet away from others when possible, even if you—or they—do not have any symptoms. Social distancing is especially important for people who are at higher risk for severe illness from COVID-19.

Methodology for social distancing detection using deep learning to evaluate the distance between people to mitigate the impact of this coronavirus pandemic. The detection tool was developed to alert people to maintain a safe distance with each other by evaluating a video feed. The video frame from the camera was used as input, and the open-source object detection pre-trained model based on the YOLOv3 algorithm was employed for pedestrian detection. Later, the video frame was transformed into top-down view for distance measurement from the 2D plane. 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.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

Coronaviruses (CoV) are a large family of viruses that cause illness ranging from the common cold to more severe diseases such as Middle East Respiratory Syndrome (MERS-CoV) and Severe Acute Respiratory Syndrome (SARS-CoV). A novel coronavirus (nCoV) is a new strain that has not been previously identified in humans.

1.1 Overview

Coronavirus disease (COVID-19) is an infectious disease caused by a newly discovered coronavirus.Most people infected with the COVID-19 virus will experience mild to moderate respiratory illness and recover without requiring special treatment. Older people, and those with underlying medical problems like cardiovascular disease, diabetes, chronic respiratory disease, and cancer are more likely to develop serious illness.The best way to prevent and slow down transmission is to be well informed about the COVID-19 virus, the disease it causes and how it spreads. Protect yourself and others from infection by washing your hands or using an alcohol based rub frequently and not touching your face. The COVID-19 virus spreads primarily through droplets of saliva or discharge from the nose when an infected person coughs or sneezes, so it's important that you also practice respiratory etiquette (for example, by coughing into a flexed elbow).

When the novel coronavirus (Covid-19) pandemic emerges, the spread of the virus has left public keep anxiety if they do not have any effective cure. The World Health Organisation (WHO) has declared Covid-19 as a pandemic due to the increase in the number of cases reported around the world . To contain the pandemic, many countries have implemented a lockdown where the government enforced that the citizens to stay at home during this critical period. The public health bodies such as the Centres for Disease Control and Prevention (CDC) had to make it clear that the most effective way to slow down the spread of Covid-19 is by avoiding close contact with other people . To flatten the curve on the Covid-19 pandemic, the citizens around the world are practicing physical distancing.

To implement social distancing, group activities and congregations such as travel, meetings, gatherings, workshops, praying had been banned during the quarantine period. The people are encouraged to use phone and email to manage and conduct events as much as pos-

sible to minimise the person-to-person contact. To further contain the spread of the virus, people are also informed to perform hygiene measures such as frequently washing hands, wearing mask and avoiding close contact with people who are ill. However, there is a difference between knowing what to do to reduce the transmission of the virus and putting them into practice.

In order to understand how to prevent the transmission of SARS-CoV-2 at work it is important to first review how the virus is spread between people and in the environment. The WHO interim guidance on mask use in the context of COVID-19 provides an overview of SARS-CoV-2 transmission and implications for infection prevention in the community . According to current knowledge about transmission, SARS-CoV-2 transmission primarily occurs between people when an infected person is in close contact with another person. The extent to which the virus will transmit between co-workers depends on the amount of viable virus being shed and expelled by a person, the type of contact that person has with others, the setting where exposure occurs and what preventative measures are in place. The SARS-CoV-2 virus can spread from the mouth or nose of an infectious person when the person coughs, sneezes, sings, breathes heavily or talks. Close contact with an infected person can result in inhalation of, or inoculation with, the virus through the mouth, nose or eyes. There is limited evidence of transmission through fomites(objects or materials that may be contaminated with viable virus, such as work equipment or surfaces) in the immediate environment around the infected person. Such transmission may occur through touching the fomites followed by touching the mouth, nose or eyes. Transmission occurs in settings outside of medical facilities, most often in indoor, crowded, and inadequately ventilated spaces, where infected persons spend long periods of time with others. This suggests SARS-CoV-2 transmission is particularly effective in crowded, confined indoor spaces where there is poor or no ventilation .

1.2 Applications

The social distance detector will be useful to monitor the people where a large number of people gathering at one place are more likely to be found. These kind of events can be monitored and will be able to alert the people and mitigate the impact caused by the corona

virus to the minimum. This can be also implemented at work places and offices after they re-start physically.

1.3 Problem Definition

The proposed model must detect the people who are not following social distancing with the use of YOLO-V3 algorithm for detection of people and calculate the distance between them and divide them into zones of red and green for people who are at risk and who are in safe zone respectively .

1.4 Aim of the project

As manual social detection is highly time consuming, and requires lot of human power to detect the distance. The Aim of the project is to implement social distancing detection using deep learning to evaluate the distance between people to mitigate the impact of this coronavirus pandemic.

2. LITERATURE SURVEY

[1] J. Redmon, S. Divvala, R. Girshick, A. Farhadi#, “You only look once: Unified, real-time object detection” the authors proposes 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 . When it comes to deep learning-based object detection, the YOLO model is considered one of the state-of-the art object 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

[2] D.T. Nguyen, W. Li, P.O. Ogunbona, “Human detection from images and videos: A survey” the authors proposes Deep learning has shown a research trend in multi-class object recognition and detection in artificial intelligence and has achieved outstanding performance on challenging datasets. Nguyen et al. presented a comprehensive analysis of state-of-the-art on recent development and challenges of human detection . The survey mainly focuses on human descriptors, machine learning algorithms, occlusion, and real-time detection.

[3] K. He, X. Zhang, S. Ren, J. Sun, “Deep residual learning for image recognition”,the authors proposes Deep CNN is a deep learning algorithm with multilayer perceptron neural networks which contain several convolutional layers, sub-sampling layers, and fully connected layers. Later, the weight in the whole layers in the networks are trained for each object classification based on its dataset. For object detection in image, the CNN model was one of the categories in deep learning which are supervised feature learning methods robust in detecting the object in different scenarios.

3. SYSTEM REQUIREMENT SPECIFICATION

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

3.1 Software Requirements

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

3.1.1 Anaconda / Jupiter note book

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda® distribution that allows you to launch applications and easily manage anaconda packages, environments, and channels without using command-line commands. Navigator can search for packages on Anaconda.org or in a local Anaconda Repository. It is available for Windows, macOS, and Linux.

In order to run, many scientific packages depend on specific versions of other packages. Data scientists often use multiple versions of many packages and use multiple environments to separate these different versions.

The command-line program conda is both a package manager and an environment manager. This helps data scientists ensure that each version of each package has all the dependencies it requires and works correctly.

Navigator is an easy, point-and-click way to work with packages and environments without needing to type conda commands in a terminal window. You can use it to find the

packages you want, install them in an environment, run the packages, and update them – all inside Navigator.

The following applications are available by default in Navigator:

- JupyterLab
- Jupyter Notebook
- Spyder
- PyCharm
- VSCode
- Glueviz
- Orange 3 App
- RStudio

3.1.2 Python

Python is an interpreter, high-level and general-purpose programming language. Python's design philosophy emphasises code readability with its notable use of significant whitespace. Its language constructs and 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), object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

Key advantages of learning Python

- Python is Interactive − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- Python is Object-Oriented − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- Python is a Beginner's Language − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

Characteristics of Python

- Following are important characteristics of Python Programming –
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- It supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

Applications of Python:

- Easy-to-learn − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- Easy-to-read − Python code is more clearly defined and visible to the eyes.
- Easy-to-maintain − Python's source code is fairly easy-to-maintain.
- A broad standard library − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- Interactive Mode − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- Portable − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.

3.1.3 Keras

Keras is one of the leading high-level neural networks APIs. It is written in Python and supports multiple back-end neural network computation engines.

Keras contains numerous implementations of commonly used neural-network building blocks such as layers, objectives, activation functions, optimisers, and a host of tools to make working with image and text data easier to simplify the coding necessary for writing deep neural network code.

Keras was created to be user friendly, modular, easy to extend, and to work with Python. The

API was “designed for human beings, not machines,” and “follows best practices for reducing cognitive load.”

Neural layers, cost functions, optimisers, initialisation schemes, activation functions, and regularisation schemes are all standalone modules that you can combine to create new models. New modules are simple to add, as new classes and functions. Models are defined in Python code, not separate model configuration files.

The Model is the core Keras data structure. There are two main types of models available in Keras: the Sequential model, and the Model class used with the functional API.

3.1.4 Tensor Flow

TensorFlow is a free and open-source library for data flow and differentiable programming across a range of tasks. It is a symbolic math library, and is most suitable for machine learning applications such as neural networks.

Created by the Google Brain team, TensorFlow is a library for numerical computation and large-scale machine learning. TensorFlow bundles together some of machine learning and deep learning (neural networking) models and makes them useful. It uses Python to provide a convenient front-end API for building applications with the framework. The major advantage of using TensorFlow is that it performs all the back-end operations in high performance C++.

How Does Tensor Flow Work?

TensorFlow allows to create data flow-graphs. These are structures that describe how data given moves through a series of processing nodes. Each node represents a mathematical operation, and each connection between nodes is a multidimensional data array called tensor.

The math operations are not performed in Python. The libraries that are available through TensorFlow are written in high-performance C++. This is the reason of increased performance of tensorflow.

GPU on Windows 10:

Getting setup with an installation of TensorFlow GPU can be done in 3 simple steps.

1. Go to anaconda prompt and setup a new virtual conda environment.
2. $ conda create –n tensorflow_gpu pip python=3.7.

   This command will create a new virtual environment with name tensorflow_gpu.
3. $ activate tensorflow_gpu.

   This command will activate the newly created virtual environment.

3.2 Hardware Requirements

Hardware Specification:

As the processing is intensive, a computer system with I5 Intel core processor and graphic card (usually CUDA) is required.

GPU:

A **graphics processing unit** (**GPU**) is a specialized electronic circuit designed to rapidly manipulate and alter memory to accelerate the creation of images in a frame buffer intended for output to a display device. GPUs are used in embedded systems, mobile phones, personal computers, workstations, and game consoles. Modern GPUs are very efficient at manipulating computer graphics and image processing. Their highly parallel structure makes them more efficient than general-purpose central processing units (CPUs) for algorithms that process large blocks of data in parallel.

4. METHODOLOGY

We will be using YOLOv3, trained on COCO dataset for object detection. In general, single-stage detectors like YOLO tend to be less accurate than two-stage detectors (R-CNN) but are significantly faster. YOLO treats object detection as a regression problem, taking a given input image and simultaneously learning bounding box coordinates and corresponding class label probabilities. It is used to return the person prediction probability, bounding box coordinates for the detection, and the centroid of the person.

NMS (Non-maxima suppression) is also used to reduce overlapping bounding boxes to only a single bounding box, thus representing the true detection of the object. Having overlapping boxes is not exactly practical and ideal, especially if we need to count the number of objects in an image. Euclidean distance is then computed between all pairs of the returned centroids. Simply, a centroid is the center of a bounding box. Based on these pairwise distances, we check to see if any two people are less than/close to 'N' pixels apart.

After calculating the distance between two persons, we will keep a threshold value of the distance and if the distance between two persons is less than that threshold values, we mark or classify the person with red rectangle around him as to direct he is in the risk zone and if the calculated distance is more than then minimum and less than the maximum value, we classify that person as in orange zone which is near to danger zone and if the calculated distance is more than the threshold value, we classify it as safe zone and we represent that person in the green rectangle around him.

4.1 System architecture

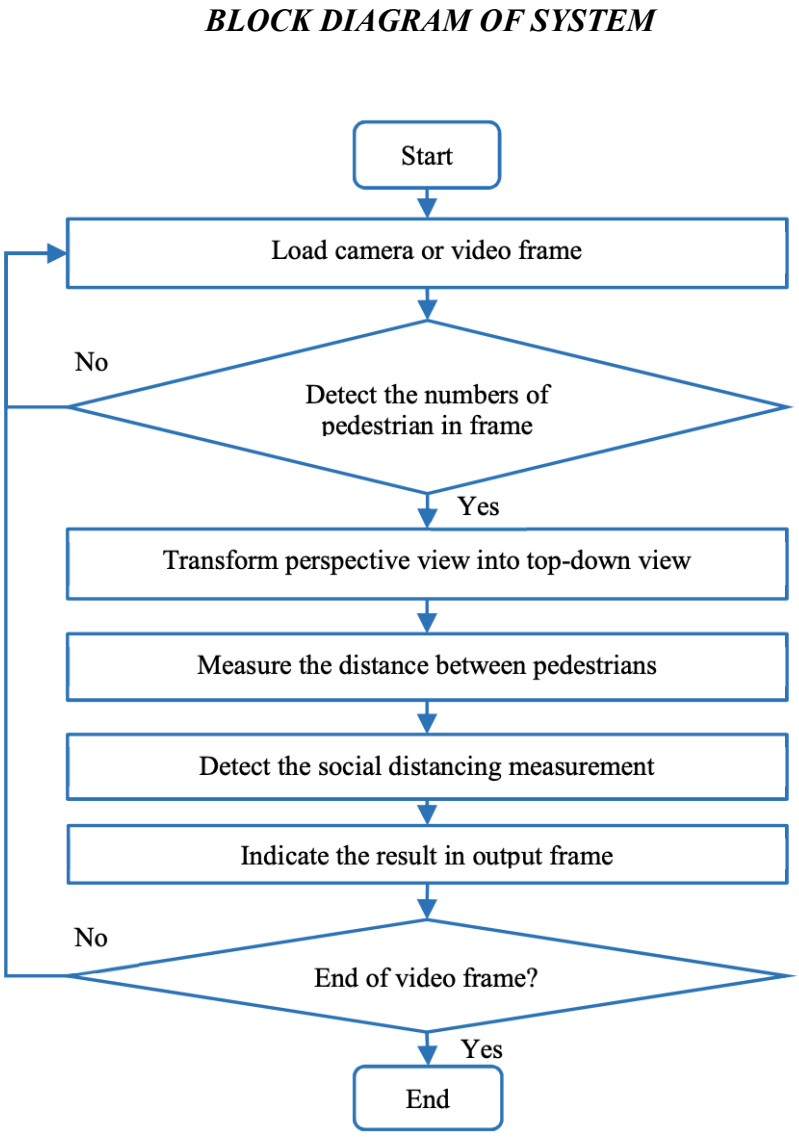


Figure 4.1.1: Flow chart of the proposed system

Figure 4.1.1 describes the main idea of the project is to take the input video or take the video from a webcam and identify the pedestrians from that video and calculate the distance be-

tween them and classify them into red and green zones. The proposed system contains the following modules

• First we need to take the input as video frame or through web cam on with we can perform analysis of detecting pedestrians .

• From the input video taken, we detect the pedestrians using YOLO V3 algorithm of object detection.

• After the detection of pedestrians, we calculate the Euclidean distance between two pedestrians using the centroids of them.

• After calculating the distance, we classify the detected pedestrians into risk zones such as red, orange and green zones.

5. IMPLEMENTATION AND RESULTS

5.1 Input video or access to camera

For our project, the input is either given by a already recorded video or it can be taken from the system's webcam or through any CCTV footage by connecting our project with its IP address for real time operation of Social distance detection.

For the taking of input video we have used the OPENCV for the processing of the video taken .

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products.[4]

Figure 5.1.1 The input video taken to process.

5.2 Detection of pedestrians

For the detection of pedestrians from the input video, we have used the YOLO (You Look Only Once) deep learning algorithm where it detects the pedestrians fro the video given.

YOLO -Version-3

YOLOv3 (You Only Look Once, Version 3) is a real-time object detection algorithm that identifies specific objects in videos, live feeds, or images.Object classification systems are used by Artificial Intelligence (AI) programs to perceive specific objects in a class as subjects of interest. The systems sort objects in images into groups where objects with similar characteristics are placed together, while others are neglected unless programmed to do otherwise.[2]

YOLO is a Convolutional Neural Network (CNN) for doing object detection. CNN's are classifier-based systems that can process input images as structured arrays of data and identify patterns between them. YOLO has the advantage of being much faster than other networks and still maintains accuracy.[1]

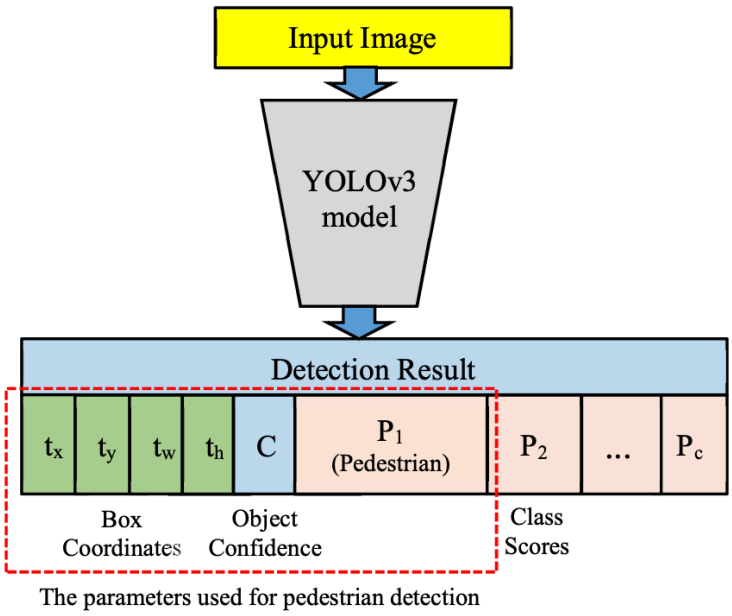


Figure 5.2.1 Architecture of Yolo-V3

The YOLOv3 algorithm first separates an image into a grid. Each grid cell predicts some number of boundary boxes (sometimes referred to as anchor boxes) around objects that score highly with the aforementioned predefined classes. Each boundary box has a respective

confidence score of how accurate it assumes that prediction should be, and detects only one object per bounding box. The boundary boxes are generated by clustering the dimensions of the ground truth boxes from the original dataset to find the most common shapes and sizes.

We have taken the already pre-trained model for YOLO with the data set as COCO. COCO dataset consists of 80 classes to distinguish and 200k tabled images in total.

```
person
bicycle
car
motorbike
aeroplane
bus
train
truck
boat
traffic light
fire hydrant
stop sign
parking meter
bench
bird
cat
dog
horse
sheep
cow
elephant
bear
zebra
giraffe
backpack
umbrella
handbag
tie
suitcase
frisbee
skis
snowboard
sports ball
```

Figure 5.2.2 The classes of COCO Data set

We have loaded the pre trained model to perform the required operations for our project.

```
In [2]: import numpy as np

        import cv2

        import math

        import time

        # Load Yolo Model

        net = cv2.dnn.readNet("yolov3.weights", "yolov3.cfg")

        classes = ["person"]
```

Figure 5.2.3 Loading of YOLO Model

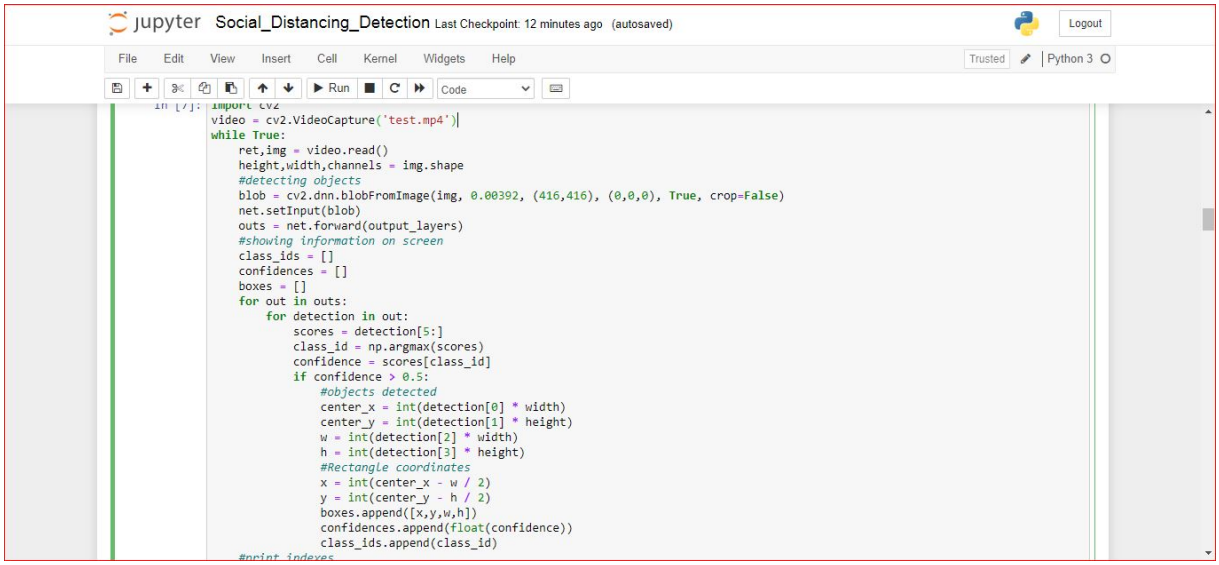


Figure 5.2.4 When YOLO is applied

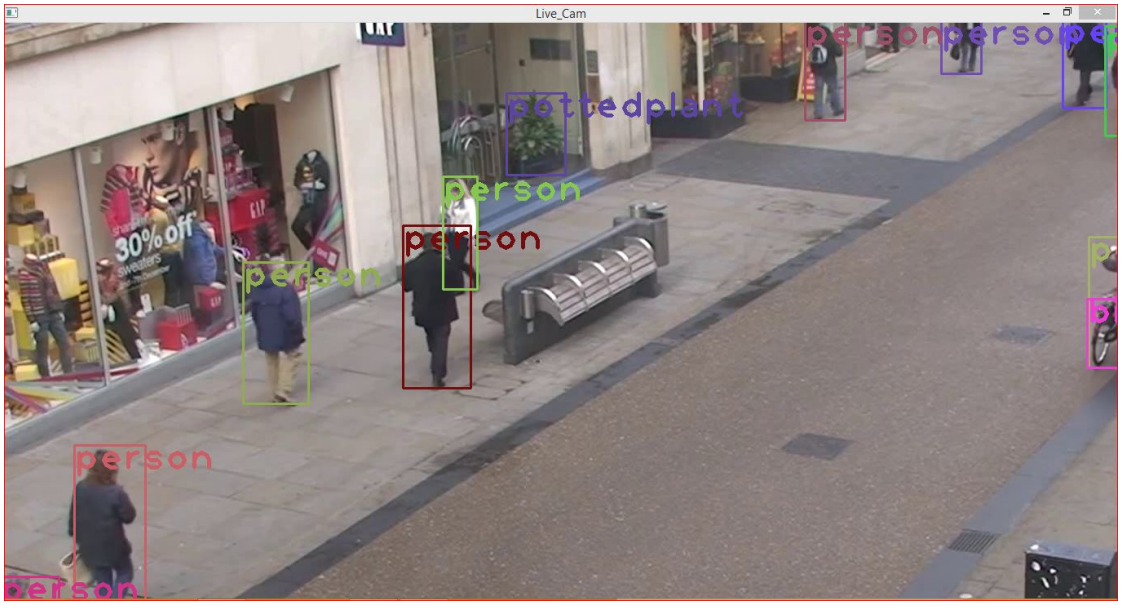


Figure 5.2.5 When YOLO algorithm is applied to detect all the objects

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Where (x1,y1) are co-ordinates of the centroid of the first person and

(x2,y2) are co-ordinates of the centroid of the second person.

But for the project here, we need not classify or detect all the objects in the video frame rather detect only the pedestrians or only detect the humans in the video frame .

For that purpose, we only consider the classes which are labeled as “PERSONS”.

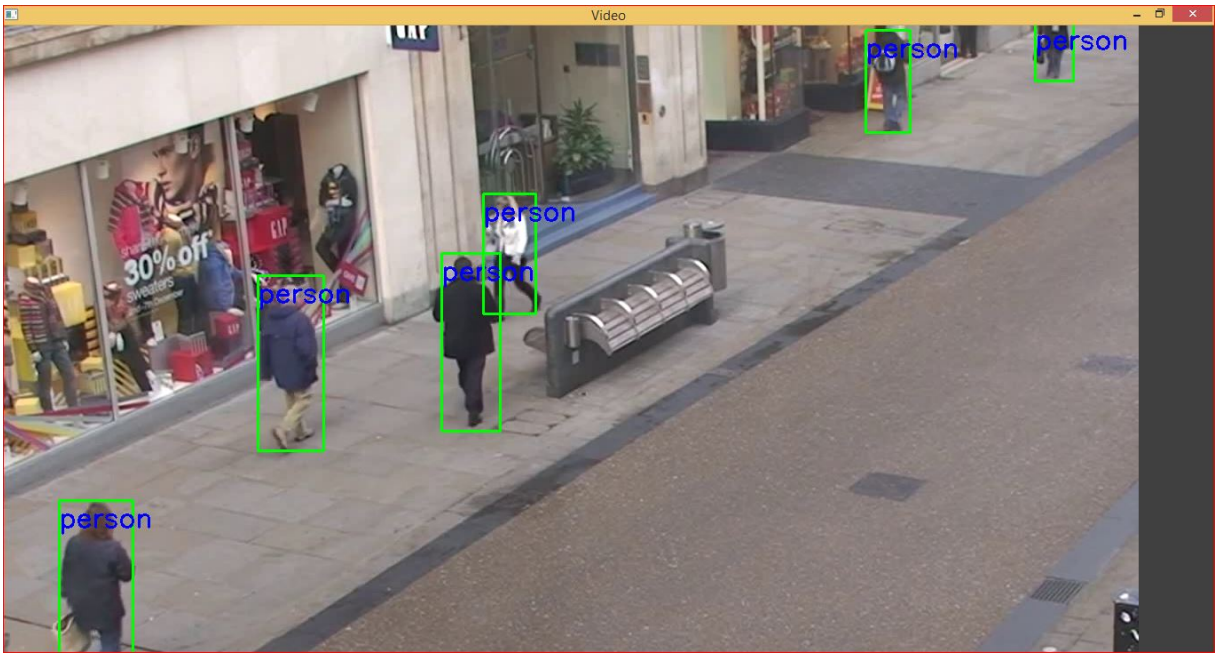


Figure 5.2.6 Output screen when only Persons are detected.

5.3 Distance calculation

After the detection of only pedestrians from the video frame, now we need to calculate the distance between them so that we can categorise them into different risk zones.

For this, we have calculated the Euclidean distance between two pedestrians by taking the centroid of a person from the centroid to another.

Euclidean distance is calculated as the square root of the sum of the squared differences between the two vectors.

Figure 5.3.1 Calculation of distance between two persons .

5.4 Classifying the detected persons into risk zones

After calculating the distance between two persons, we will keep a threshold value of the distance and if the distance between two persons is less than that threshold values, we mark or classify the person with red rectangle around him as to direct he is in the risk zone and if the calculated distance is more than then minimum and less than the maximum value, we classify that person as in orange zone which is near to danger zone and if the calculated distance is more than the threshold value, we classify it as safe zone and we represent that person in the green rectangle around him.

Distance calculated <= T ————————> RED

Distance calculated = In between ——->ORANGE

Distance calculated >= T ————————>GREEN

Figure 5.4.1 Classification persons into zones

After the classification of persons into different zones, we connect each person with another person who is closest to him with a connected line so that if there are more than 2 people at a particular place, we could differentiate among persons as who is in which zone.

Figure 5.4.2 Output screen after classifying persons into different zones

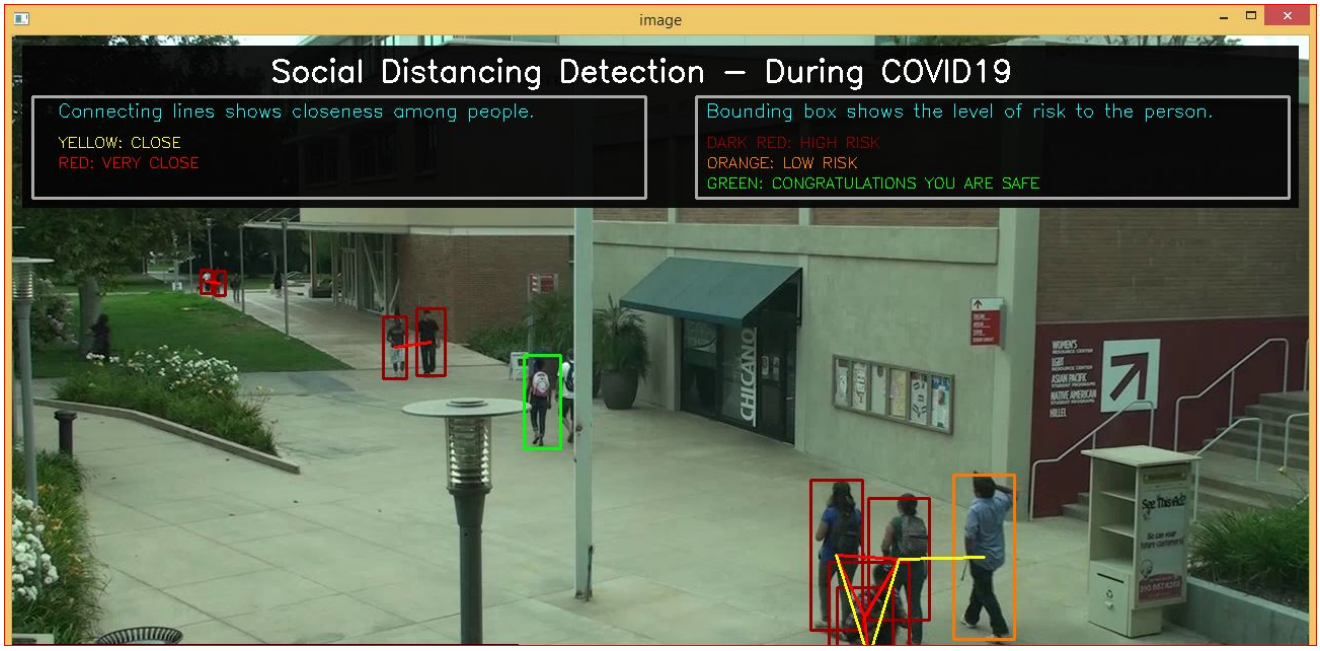


Figure 5.4.3 connecting lines and text in the output

After the successful execution of the connecting lines, we have used CHIME[5] open source package to create an alarm when any red zone people are detected to alert the public who are present in the video frame.

At the end , we have added some additional features like the total number of people in the video frame and number of people in each zone (Red, orange and green).

5.5 Results

Figure 5.5.1 Social distance detection using sample video- 1

Figure 5.5.2 Social distance detection using sample video- 2

From the figures 6.1 and 6.2, we can can see that the input video is taken and from that pedestrians are detected and we find the distance between them and classify them into different risk zones such as red green and orange.

6. CONCLUSION AND FUTURE SCOPE

A project to detect the social distancing among the people by taking the video frame has been implemented and executed successfully. By this, we can minimise the spread of corona virus up to a large extent. This can be also useful when if we face any kind of pandemic in the future as well to stop spreading the diseases from one to one. This can be mainly useful in public places such as malls, offices, and other places.

Although by this project, we reduce lot of human power up to a large extent but still some people are required to continuously watch the video and alert if people are found not being in social distancing. To avoid this, we can attach an external alarm so that whenever there is a person detected in the red zone, the alarm can automatically be working to alert people.

7. REFERENCES

2. J. Redmon, S. Divvala, R. Girshick, A. Farhadi, “You only look once: Unified, real-time object detection”, In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 779-788. 2016.

3. D.T. Nguyen, W. Li, P.O. Ogunbona, “Human detection from images and videos: A survey”, Pattern Recognition, 51:148-75, 2016.

4. K. He, X. Zhang, S. Ren, J. Sun, “Deep residual learning for image recognition”, In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770-778, 2016.

4. https://opencv.org/

5. https://towardsdatascience.com/personalized-your-python-notification-sounds-with-chime-

8. APPENDIX

```
#import the necessary packages

import numpy as np import cv2

import math

import time

# Load Yolo

net = cv2.dnn.readNet("yolov3.weights", "yolov3.cfg") classes = []

with open("coco.names", "r") as f:

classes = [line.strip() for line in f.readlines()] layer_names = net.getLayerNames()

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

np.random.uniform(0, 255, size=(len(classes),3)) cap = cv2.VideoCapture('test.mp4')

while(cap.isOpened()):

    # Capture frame-by-frame

ret, img = cap.read()

if not ret: break

if width is None or height is None: height,width=img.shape[:2] q=width

 #height, width, channels = img.shape

img = img[0:height, 0:width]

height,width = img.shape[:2]

    # Detecting objects 0.00392

blob = cv2.dnn.blobFromImage(img,0.00392, (416, 416), (0,0,0), True, crop=False)

net.setInput(blob)

start = time.time()

outs = net.forward(output_layers) end=time.time()
```

```
# Showing informations on the screen class_ids = []
confidences = []
boxes = []
for out in outs:

for detection in out:
scores = detection[5:] class_id = np.argmax(scores)

confidence = scores[class_id] if classes[class_id]=="person":

        #0.5 is the threshold for confidence
if confidence > 0.5:
# Object detected
#Purpose : Converts center coordinates to

rectangle coordinates
        # x, y = midpoint of box
center_x = int(detection[0] * width) center_y = int(detection[1] * height) # w, h = width,
height of the box
w = int(detection[2] * width)

h = int(detection[3] * height)
# Rectangle coordinates
x = int(center_x - w / 2)
y = int(center_y - h / 2) boxes.append([x, y, w, h]) confidences.append(float(confidence))
class_ids.append(class_id)

indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.5) #print(indexes)
font = cv2.FONT_HERSHEY_SIMPLEX
if len(indexes)>0:

status=list()
idf = indexes.flatten() close_pair = list() s_close_pair = list() center = list()
dist = list()
for i in idf:
```

```
(x, y) = (boxes[i][0], boxes[i][1]) (w, h) = (boxes[i][2], boxes[i][3]) label =
str(classes[class_ids[i]]) if label=='person' :

cv2.rectangle(img, (x,y), (x+w,y+h), (0,255,0), 2) cv2.putText(img,label,(x,y+30),font,1,
(255,0,0),2) center.append([int(x + w / 2), int(y + h / 2)]) status.append(0)

cv2.imshow("Video", img)
if cv2.waitKey(1) & 0xFF == ord('q'):

break cv2.waitKey(1)

fourcc = cv2.VideoWriter_fourcc(*'MJPG')
output = cv2.VideoWriter('output4.mp4',fourcc, 20.0,

(img.shape[1], img.shape[0])) #cv2.imwrite("output1.mp4",img) #img = cv2.flip(img,0)

output.write(img) cap.release()

output.release() cv2.destroyAllWindows()

In [2]:

# import the necessary packages
import numpy as np import cv2
import math
import time

import chime #for notification sound chime.theme('zelda')
# Load Yolo
net = cv2.dnn.readNet("yolov3.weights", "yolov3.cfg") classes = []

with open("coco.names", "r") as f:
classes = [line.strip() for line in f.readlines()]

layer_names = net.getLayerNames()
output_layers = [layer_names[i[0] - 1] for i in
```

```
net.getUnconnectedOutLayers()] np.random.uniform(0, 255, size=(len(classes),3)) #start
video
cap = cv2.VideoCapture('example.mp4')
#cap = cv2.VideoCapture(0)

#cap = cv2.VideoCapture('http://192.168.43.1:8080//video')
```

```
def E_dist(p1, p2):
return ((p1[0] - p2[0]) ** 2 + (p1[1] - p2[1]) ** 2) ** 0.5

def isclose(p1, p2):
c_d = E_dist(p1, p2)
calib = (p1[1] + p2[1]) / 2 if 0 < c_d < 0.15 * calib:

return 1
elif 0 < c_d < 0.2 * calib:

return 2 else:

return 0

height,width=(None,None)
q=0
#Start working on video or camera while(cap.isOpened()):

    # Capture frame-by-frame
ret, img = cap.read()

if not ret: break

if width is None or height is None: height,width=img.shape[:2] q=width

    #height, width, channels = img.shape
img =img[0:height, 0:q]

height,width=img.shape[:2]

    # Detecting objects 0.00392
```

```
blob = cv2.dnn.blobFromImage(img,0.00392, (416, 416), (0,0,0), True, crop=False)

net.setInput(blob)
start = time.time()
outs = net.forward(output_layers) end=time.time()

    # Showing informations on the screen
class_ids = [] confidences = [] boxes = []
for out in outs:

for detection in out:
scores = detection[5:]
class_id = np.argmax(scores) confidence = scores[class_id]
#0.5 is the threshold for confidence

if confidence > 0.5:
# Object detected
#Purpose : Converts center coordinates to

rectangle coordinates
            # x, y = midpoint of box
center_x = int(detection[0] * width) center_y = int(detection[1] * height)

             # w, h = width, height of the box
w = int(detection[2] * width)
h = int(detection[3] * height)
# Rectangle coordinates
x = int(center_x - w / 2)
y = int(center_y - h / 2) boxes.append([x, y, w, h]) confidences.append(float(confidence))
class_ids.append(class_id)

indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.5) #print(indexes)
font = cv2.FONT_HERSHEY_SIMPLEX
if len(indexes)>0:
```

```
status=list()
idf = indexes.flatten() close_pair = list()

s_close_pair = list() center = list() persons = list()
dist = list()

for i in idf:
(x, y) = (boxes[i][0], boxes[i][1]) (w, h) = (boxes[i][2], boxes[i][3]) label =
str(classes[class_ids[i]]) if label == 'person':

center.append([int(x + w / 2), int(y + h / 2)]) persons.append(i)
status.append(0)

for i in range(len(center)):
for j in range(len(center)):

    #compare the closeness of two values
g=isclose(center[i], center[j]) if g ==1:

close_pair.append([center[i],center[j]]) chime.success()
status[i] = 1
status[j] = 1

elif g == 2:
s_close_pair.append([center[i], center[j]])

1.0)

if status[i] != 1: status[i] = 2

if status[j] != 1: status[j] = 2

total_p = len(center) low_risk_p = status.count(2) high_risk_p = status.count(1) safe_p = sta-
tus.count(0)
kk = 0
for i in persons:
```

```
sub_img = img[10:170, 10:width - 10]
black_rect = np.ones(sub_img.shape, dtype=np.uint8)*0 res = cv2.addWeighted(sub_img,
0.77, black_rect,0.23,

img[10:170, 10:width - 10] = res

   # adding text to image
            #(image,text,org( X coordinate value, Y
coordinate value),font,fontScale,color,thikness)
cv2.putText(img, "Social Distancing Detection - During COVID19 ", (255, 45),font, 1, (255,
255, 255), 2)

        #image = cv2.rectangle(image, start_point, end_point,
color, thickness)
170), 2)

cv2.rectangle(img, (20, 60), (625, 160), (170, 170,

cv2.putText(img, "Connecting lines shows closeness among people. ", (45, 80),font, 0.6, (255,
255, 0), 1)

cv2.putText(img, "YELLOW: CLOSE", (45, 110),font, 0.5, (0, 255, 255), 1)

cv2.putText(img, "RED: VERY CLOSE", (45, 130),font, 0.5, (0, 0, 255), 1)

cv2.rectangle(img, (675, 60), (width -20, 160), (170, 170, 170), 2)

cv2.putText(img, "Bounding box shows the level of risk to the person.",(685, 80),font, 0.6,
(255, 255, 0), 1)

cv2.putText(img, "DARK RED: HIGH RISK", (685, 110),font, 0.5, (0, 0, 150), 1)

cv2.putText(img, "ORANGE: LOW RISK", (685, 130),font, 0.5, (0, 120, 255), 1)

cv2.putText(img, "GREEN: SAFE", (685, 150),font, 0.5, (0, 255, 0), 1)
```

```
tot_str = "NUMBER OF PEOPLE: " + str(total_p) high_str = "RED ZONE: " + str(high-
_risk_p) low_str = "ORANGE ZONE: " + str(low_risk_p) safe_str = "GREEN ZONE: " +
str(safe_p) #image ROI

sub_img = img[height - 120:height-20, 0:500] #cv2.imshow("sub_img",sub_img)
black_rect = np.ones(sub_img.shape, dtype=np.uint8) *

0

1.0)

res = cv2.addWeighted(sub_img, 0.8, black_rect, 0.2,

img[height - 120:height-20, 0:500] = res cv2.putText(img, tot_str, (10, height - 75),font, 0.6,

(255, 255, 255), 1)
cv2.putText(img, safe_str, (300, height - 75),font, 0.6, (0, 255, 0), 1)

cv2.putText(img, low_str, (10, height - 50),font, 0.6, (0, 120, 255), 1)

cv2.putText(img, high_str, (300, height - 50),font, 0.6, (0, 0, 150), 1)

150), 2)

(x, y) = (boxes[i][0], boxes[i][1])
(w, h) = (boxes[i][2], boxes[i][3]) #color of the ractangle when is too close if status[kk] == 1:

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

elif status[kk] == 0:
cv2.rectangle(img, (x, y), (x + w, y + h), (0,

255, 0), 2)
else:

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

120, 255), 2)
kk += 1
```

```
for h in close_pair:
cv2.line(img, tuple(h[0]), tuple(h[1]), (0, 0, 255),

2)

for b in s_close_pair:
cv2.line(img, tuple(b[0]), tuple(b[1]), (0, 255, 255),

2)
imgs = cv2.resize(img,(1200,700)) cv2.imshow('image',imgs)
if cv2.waitKey(1) & 0xFF == ord('q'):

break cv2.waitKey(1)

    #FourCC code is passed as
fourcc = cv2.VideoWriter_fourcc(*'MJPG')
output = cv2.VideoWriter('output4.mp4',fourcc, 20.0,

(img.shape[1], img.shape[0])) #cv2.imwrite("output1.mp4",img) #img = cv2.flip(img,0) out-
put.write(img)

cap.release() output.release() cv2.destroyAllWindows()
```

Social Distance Detection

Batch no. A-6

By: K. Raj Kumar(1602-17-737-027)
A. Vishwa Prabhath (1602-17-737-058)

Internal Guide : Dharma reddy (Assistant professor)

Table of contents :

Abstract

- In the present situation, social distancing among the people is much needed for controlling the pandemic .
- It is said that people must follow a minimum of 6 feet social distancing in order to not get affected by the corona virus.
- Hence, if we detect the people who are not following the social distancing, we can warn them and mitigate the risk.

Applications :

- Shopping malls
- Easy for the police to detect social distancing
- Social distancing in offices

Problem Statement

- To develop a model for social distancing detection using deep learning to evaluate the distance between people to mitigate the impact of this coronavirus pandemic.

People with social distancing

People without social distancing

Literature Survey

- Landing AI Creates an AI Tool to Help Customers Monitor Social Distancing in the Workplace [Onlive]. Available at https://landing.ai/landing-ai-creates-an-ai-tool-to-help-customers- monitor-social-distancing-in-the-workplace/ (Access on 4 May 2020).

- R.Girshick,J.Donahue,T.Darrell,J.Malik."Richfeaturehierarchies for accurate object detection and semantic segmentation." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 580-587. 2014.
- J. Redmon, S. Divvala, R. Girshick, A. Farhadi, “You only look once: Unified, real-time object detection”, In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 779-788. 2016.
- K. Simonyan, A. Zisserman, “Very deep convolutional networks for large-scale image recognition”, arXiv preprint arXiv:1409.1556, 2014.

System Requirements

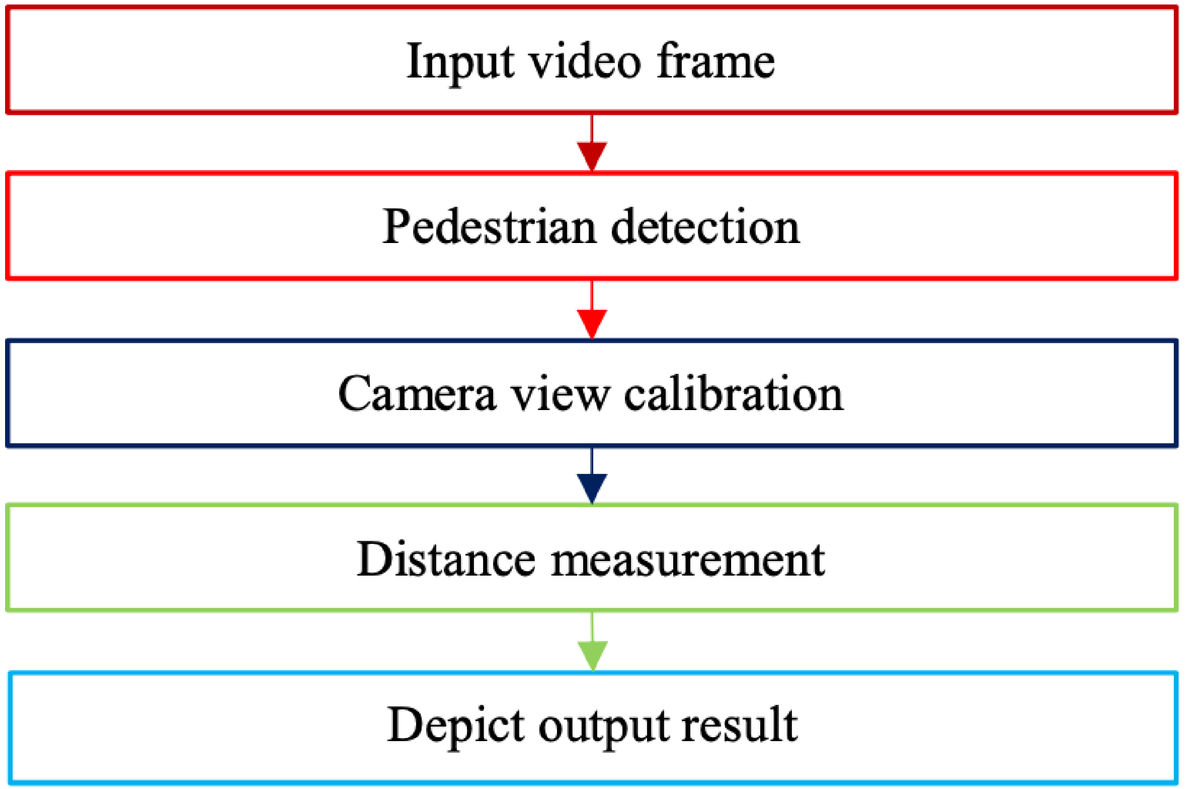
SOFTWARE REQUIREMENTS :

- Programming language : Python
- Operating system : Windows / Mac OS / ubuntu
- IDE : Anaconda Navigator

HARDWARE REQUIRMENTS :

- Ram : Minimum 4GB
- Processor : Minimum i3
- Hard disc : 250 GB

Methodology



Algorithms

- YOLOv3 algorithm was used to detect the pedestrian in the video frame.
- OpenCV for Camera view calibration
- Finding the distance between two persons using the distance formula.

Data sets and inputs :

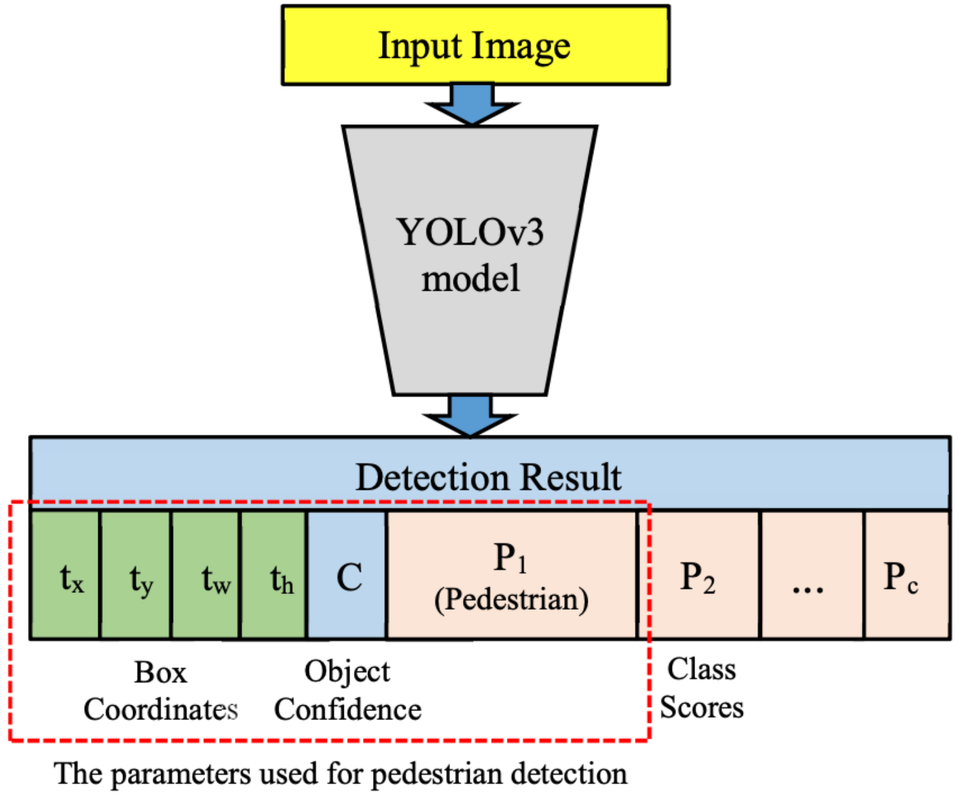
- Coco data set is used for object detection
- Coco data set has 80 categories and has 200k labelled images.
- An video with pedestrians is taken for input.

Like the below mentioned , we have 80 Categories in the COCO data set

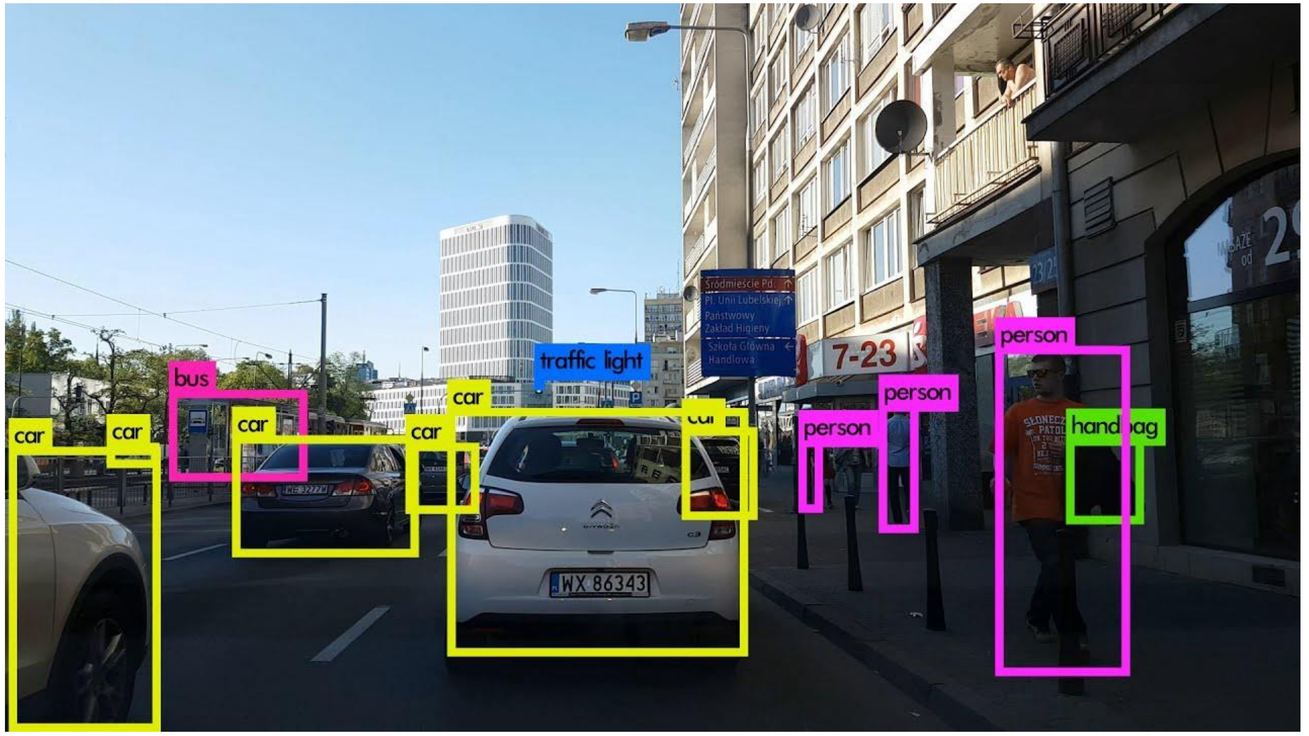
```
person
bicycle
car
motorbike
aeroplane
bus
train
truck
boat
traffic light
fire hydrant
stop sign
parking meter
bench
bird
cat
dog
horse
sheep
cow
elephant
bear
zebra
giraffe
backpack
umbrella
handbag
tie
suitcase
frisbee
skis
snowboard
sports ball
```

YOLO V3

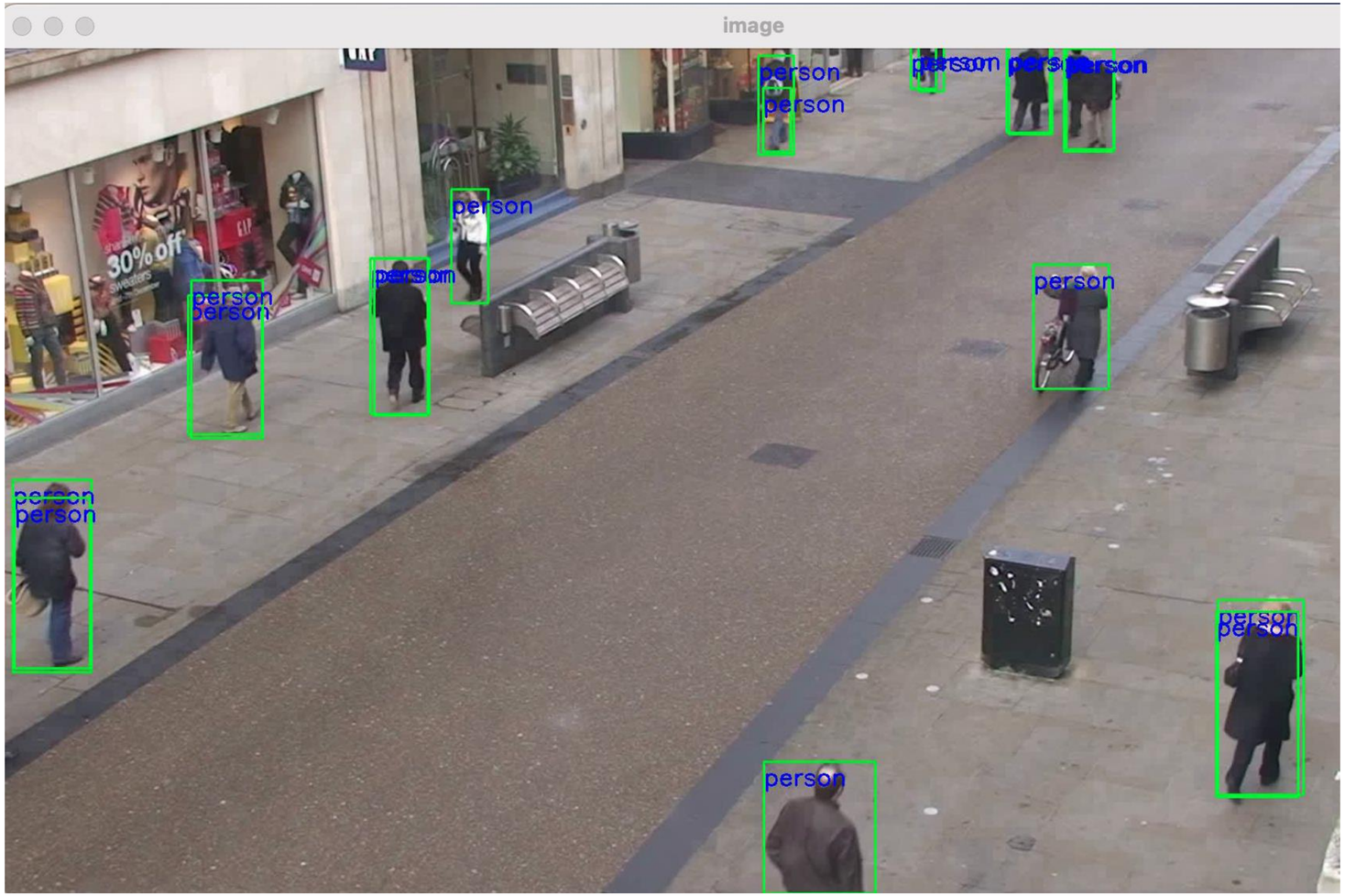
- YOLOv3 is the latest variant of a popular object detection algorithm.The published model recognises 80 different objects in images and videos.
- From the detection result, only pedestrian class was used and other object classes are ignored .



The parameters used for pedestrian detection

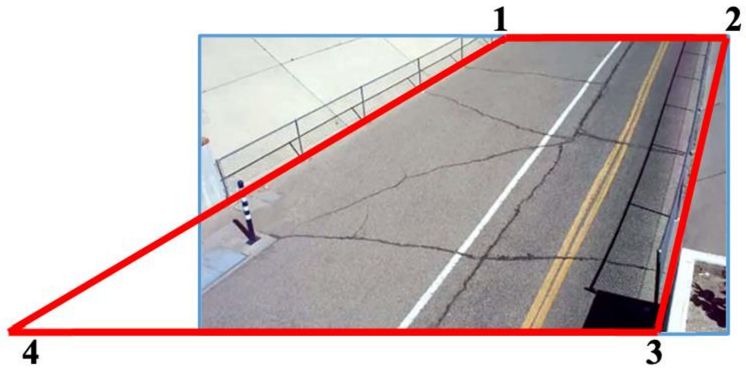


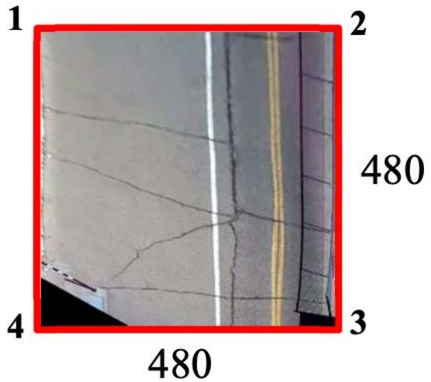
The frames when YOLO is applied and pedestrians are detected



Camara view calibration

- Selecting four points in the perspective view and mapping them to the corners of a rectangle in the 2D image view.



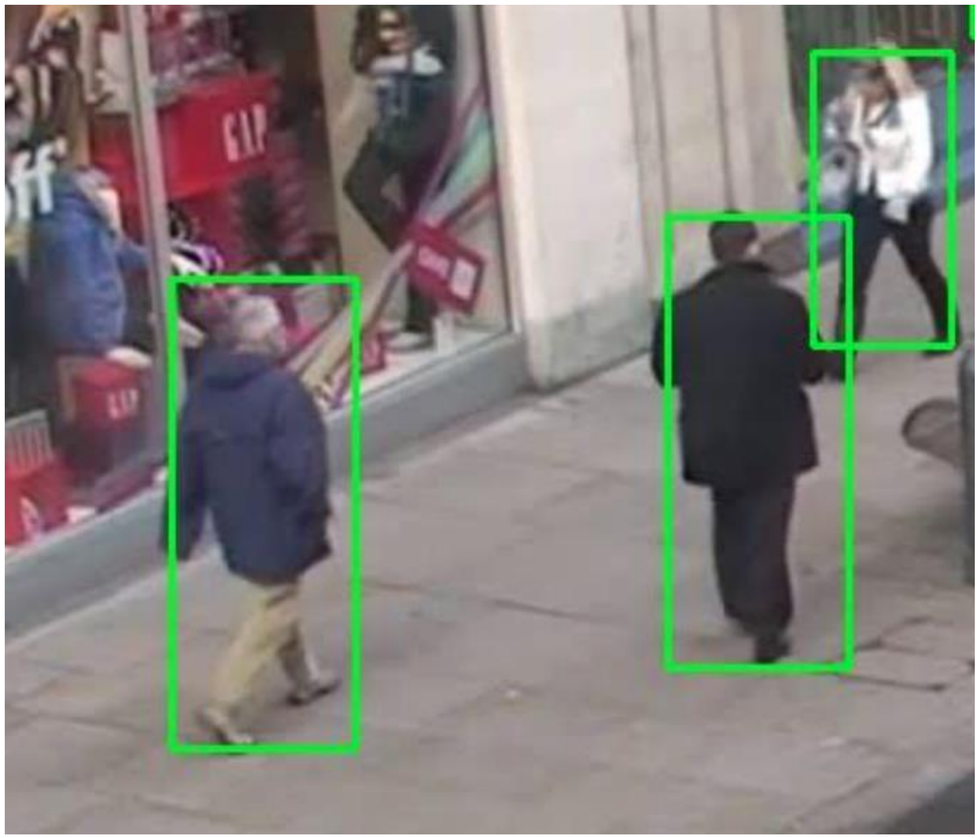


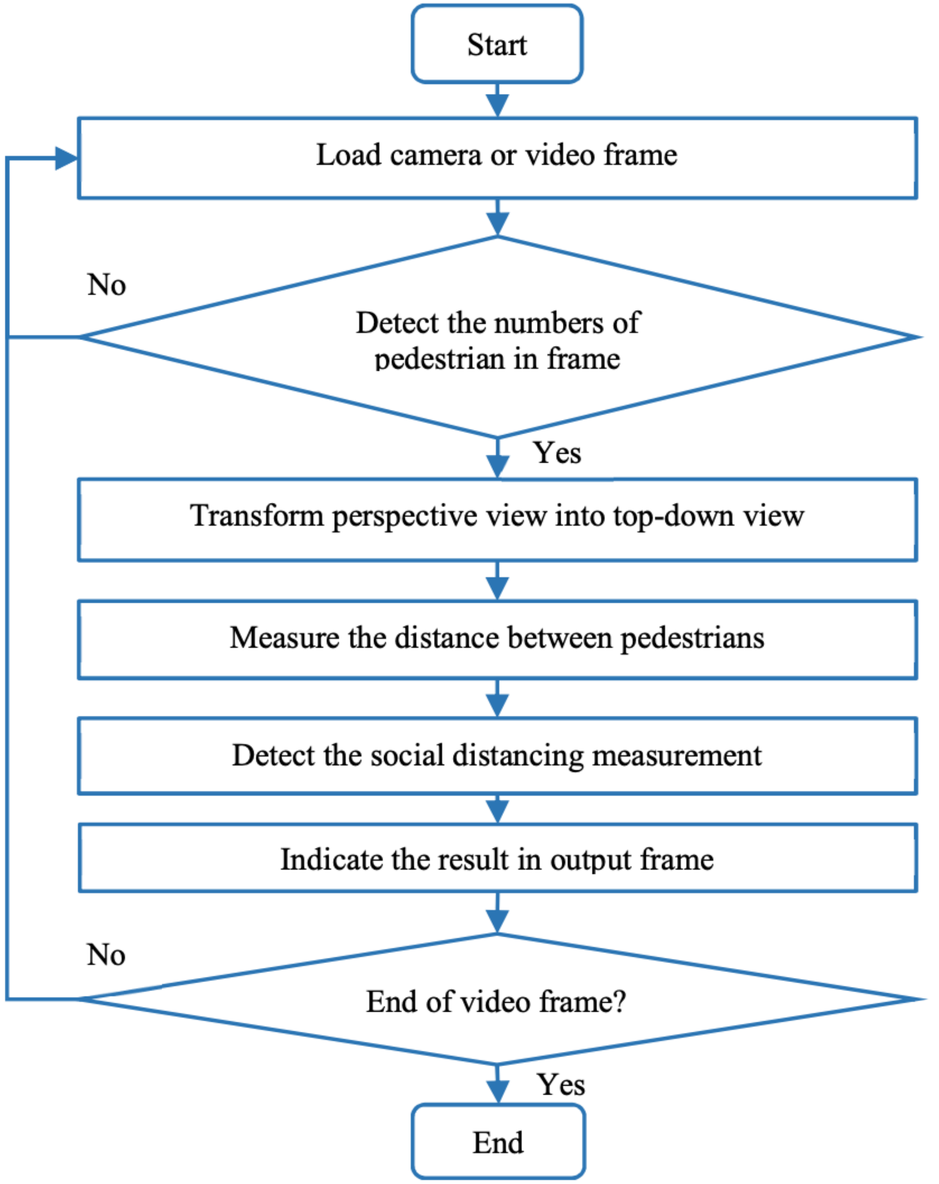
Distance calculation

- The distance between two persons can be calculated as : $d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$
- The pair of pedestrians whose distance is below the minimum acceptable distance, is marked in red, and the rest is marked in green.

$$c = \begin{cases} red & d < t \\ green & d \geq t \end{cases}$$

Risk detection in pedestrians using distance measure





Flow chart of the project

Output


image
Social Distancing Detection – During COVID19
Connecting lines shows closeness among people.
YELLOW: CLOSE
RED: VERY CLOSE
Bounding box shows the level of risk to the person.
ARK RED: HIGH RISK
RANGE: LOW RISK
REEN: CONGRATULATIONS YOU ARE SAFE
30% off
sweaters

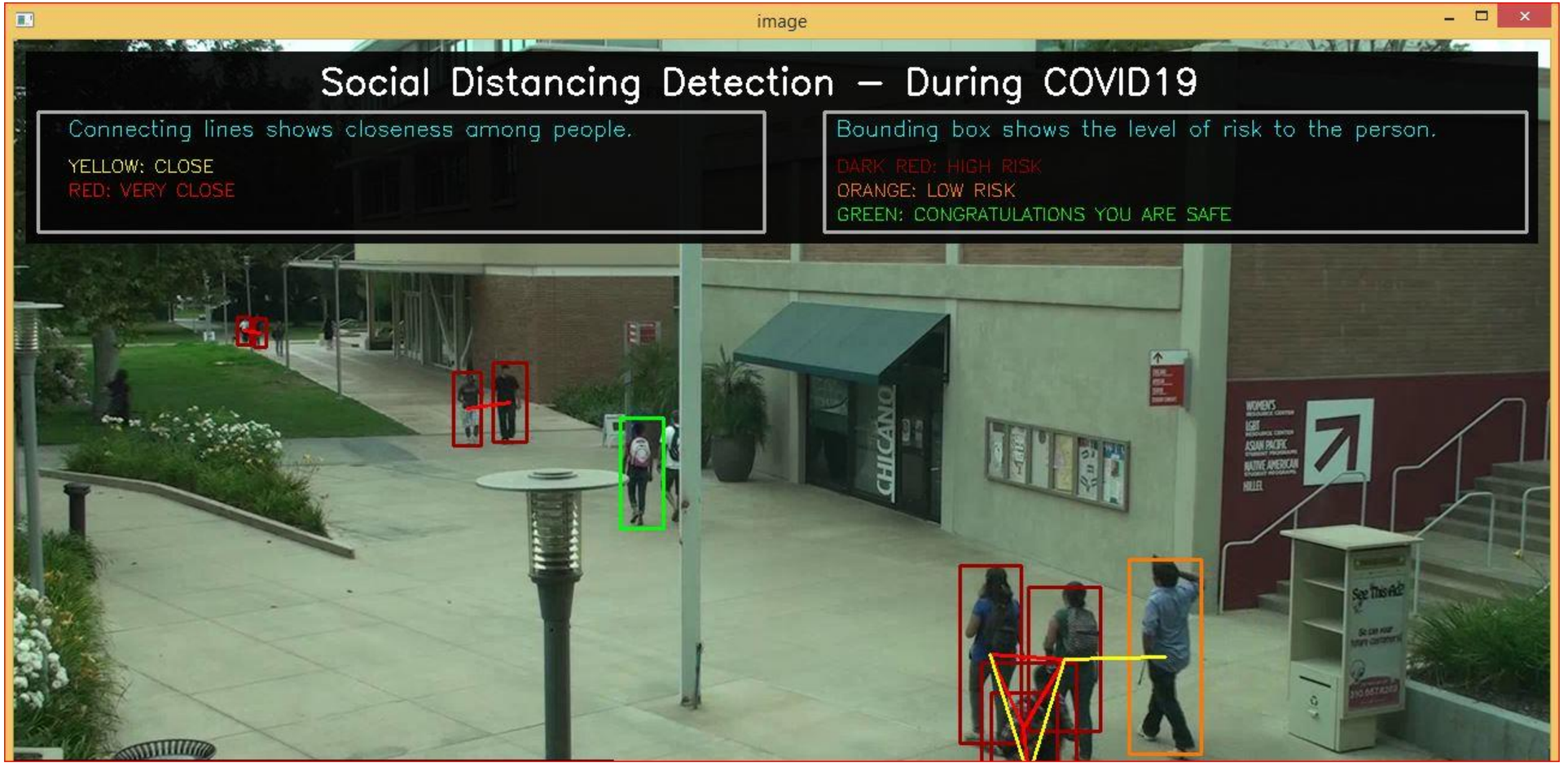


image
Social Distancing Detection – During COVID19
Connecting lines shows closeness among people.
YELLOW: CLOSE
RED: VERY CLOSE
Bounding box shows the level of risk to the person.
DARK RED: HIGH RISK
ORANGE: LOW RISK
GREEN: CONGRATULATIONS YOU ARE SAFE
CHICANO

Conclusion

- As, in real time, manual detection of social distancing is highly time consuming and which needs a lot of human effort, this project provides us an accurate and automatic detection of social distancing.

Future scope :

- In the future, we would like to apply this for real time applications taking real data such as taking input from the CCTV, etc
- With bringing more of this kind of applications, helps people protecting themselves from virus of any kind.

References :

- openCV resources from https://opencv.org/
- D.T. Nguyen, W. Li, P.O. Ogunbona, “Human detection from images and videos: A survey”, Pattern Recognition, 51:148-75, 2016.
- J. Redmon, S. Divvala, R. Girshick, A. Farhadi, “You only look once: Unified, real-time object detection”, In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 779-788. 2016.

Thank you