

SOCIAL DISTANCING ANALYZER

Submitted in partial fulfilment of the requirements for the award of Bachelor of
Engineering degree in Electronics and Communication Engineering

by

PRIYADHARSHINI G (38130172)



DEPARTMENT OF ELECTRONICS AND COMMUNICATION
ENGINEERING
SCHOOL OF ELECTRICAL AND ELECTRONICS

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with Grade “A” by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600 119

APRIL – 2022



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

This is to certify that this Project Report is the bonafide work of **G. Priyadharshini** (38130172) who carried out the project entitled "**SOCIAL DISTANCING ANALYZER**" under our supervision from September 2021 to April 2022.

INTERNAL GUIDE

EXTERNAL GUIDE

Mrs. L. MAGTHELIN THERASE M.E, (Ph.D.)

HEAD OF THE DEPARTMENT

Dr. T. Ravi M.E, Ph.D.,

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I Priyadharshini G (38130172) hereby declare that the Project Report entitled “Social distance analyser” done by us under the guidance of Mrs. L. Magthelin Therase M.E., (Ph.D.) submitted in partial fulfilment of the requirement for the award of Bachelor of Engineering degree in Electronics and Communication Engineering.

DATE :

PLACE :

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to Board of Management of SATHYABAMA for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. N. M. NANDHITHA M.E., Ph.D.**, Dean, School of Electrical and Electronics and **Dr. T. RAVI M.E., Ph.D.**, Head of the Department, Dept. of Electronics and Communication Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my project guide **L. MAGTHELIN THERASE M.E., (Ph.D.)** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the Department of Electronics and Communication Engineering who were helpful in many ways for the completion of the project.

ABSTRACT

This work presents a 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 estimate 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. The developed technique can be further developed as a detection tool in real time application.

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	v
	LIST OF FIGURES	viii
	LIST OF ABBREVIATIONS	viii
1	INTRODUCTION	1
1.1	Pedestrian Detection	3
1.2	Camera View Calibration	4
1.3	Distance Measurement	4
2	LITERATURE SURVEY	5
2.1	Measuring Social Distancing Practices with Geo-located Tweets	5
2.2	A video observational analysis	6
2.3	A proximity-based indoor navigation system tackling the COVID-19 social distancing measures	7
2.4	Person Detection for Social Distancing and Safety Violation alert based on Segmented ROI	8
2.5	Monitoring social distancing under various low light conditions with deep learning and a single	9
2.6	A Comprehensive Survey of Enabling and Emerging Technologies for Social Distancing	10
3	EXISTING SYSTEM	11
3.1	Image Segmentation	11
3.2	Semantic Segmentation	12
3.3	Instance Segmentation	12

4	PROBLEM DESCRIPTION	13
4.1	Problem statement	13
4.2	Problem Objective	13
5	PROPOSED SYSTEM	14
5.1	Inputs and Outputs of the code	14
5.1.1	Mark necessary inputs	14
5.1.2	Input: First 4 points	14
5.1.3	Input: Last 2 points	15
5.1.4	Output: Detected pedestrians in original video	15
5.1.5	Output: Social Distancing Parameters	15
5.2	Working procedure	16
5.3	Future works	17
6	RESULT AND DISCUSSIONS	18
7	CONCLUSION AND FUTURE WORK	20
7.1	Appendix I	21
7.1.1	Hardware Requirements	21
7.1.2	Software Requirements	21
7.2	Appendix II	22
7.2.1	Code	22
7.3	Appendix III	35
7.4	References	37

LIST OF FIGURES

S. No	Name	Page no
1.1	General Architecture	2
1.2	Pedestrian Detection	3
6.1	Processing Diagram	18
7.1	Social distance analyser -1	35
7.2	Social distance analyser -2	35
7.3	Social distance analyser -3	36

LIST OF ABBREVIATIONS

CNN -	Convolutional Neural Network
YOLO V3 -	You Only Look Once, Version 3
ROI -	Region of interest
Open CV -	Open Source Computer Vision

CHAPTER 1

INTRODUCTION

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 possible to minimize 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.

These methods can also be applied to detect the social distance between people. The following points summarizes the main components of this approach:

1. Deep learning has gained more attention in object detection was used for human detection purposes.
2. Develop a social distancing detection tool that can detect the distance between people to keep safe.
3. Evaluation of the classification results by analysing real-time video streams from the camera.

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 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 present 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 as shown in Figure 1.1.

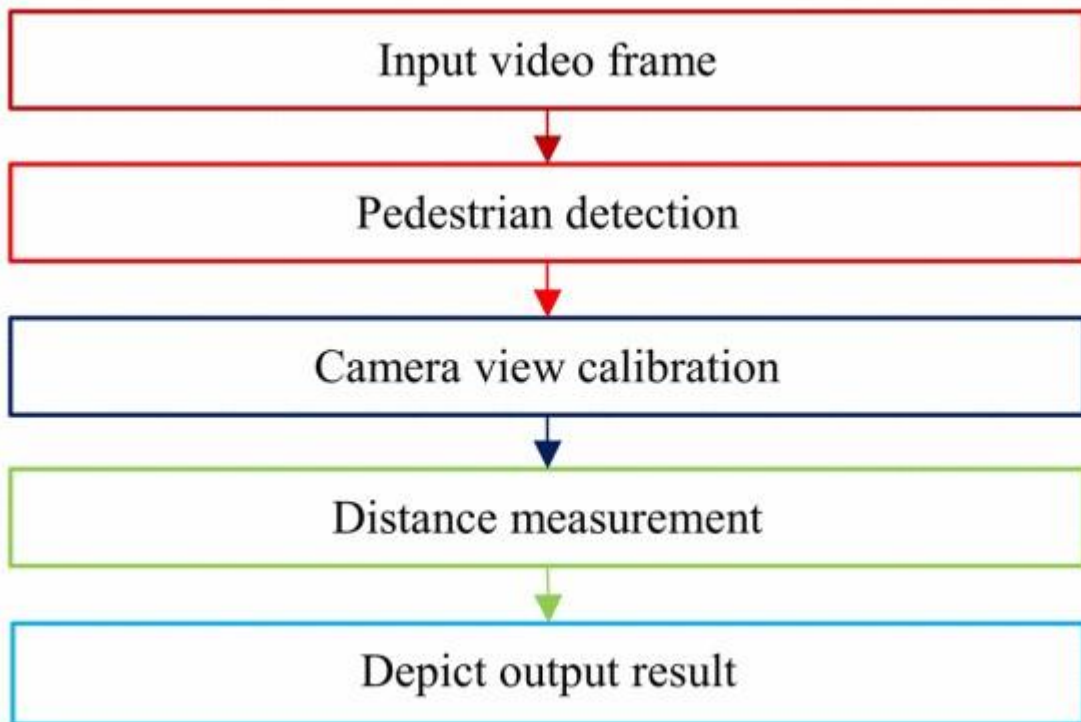


Figure 1.1 General Architecture

1.1 PEDESTRIAN 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. 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 is shown in Figure 1.2. The YOLO algorithm was considered as an object detection taking a given input image and simultaneously learning bounding box coordinates (t_x , t_y , t_w , t_h), object confidence and corresponding class label probabilities (P_1 , P_2 , ..., P_c). The YOLO trained on the COCO dataset which consists of 80 labels including human or pedestrian classes. In this work, the only box coordinates, object confidence and Pedestrian object class from detection result in the YOLO model were used for pedestrian detection.

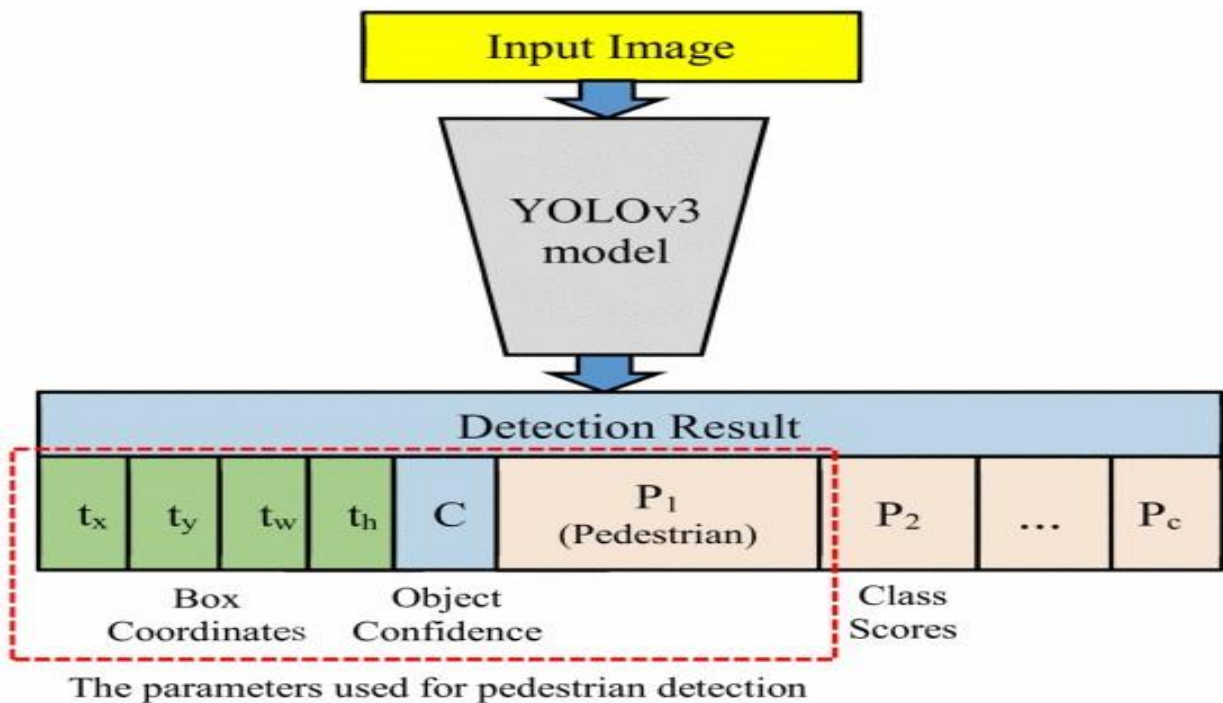


Figure 1.2 Pedestrian Detection

1.2 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. 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.

1.3 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 centre 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.

CHAPTER 2

LITERATURE SURVEY

2.1 MEASURING SOCIAL DISTANCING PRACTICES WITH GEO-LOCATED TWEETS

INFERENCES

Social distancing is an important tool in the fight against COVID-19; however, its implementation by the general public can vary widely. Although a state governor may issue an order for the practice, individuals in different states may respond to this order in different ways. Social distancing policies in the United States reduced the daily growth rate of COVID-19 cases. However, if we only consider the social distancing policy duration and daily confirmed cases, it is difficult to rule out potential confounders, including additional policies for wearing masks and improving hygiene as well as other social norms.

Therefore, understanding actual reductions in travel and social contacts is critical to measuring the effectiveness of such policies. Using mobile phone data, mobility patterns were strongly correlated with decreased rates of COVID-19 case growth for the 25 most affected counties in the United States. These social distancing policies may remain in effect for an extended period of time. Thus, the public may begin to relax their practices, making additional policies necessary.

Researchers showed the effectiveness of strict social distancing followed by testing and contact tracing by modelling mobility data from in the Boston metropolitan area. Additionally, epidemiologists have already modelled the impact of social distancing policies on the course of disease outbreaks. These models may be more effective when incorporating actual measures of social distancing rather than assuming that official policies are implemented in practice.

2.2 A VIDEO OBSERVATIONAL ANALYSIS

INFERENCES

Social distancing has been a critical non-pharmaceutical measure to slow the spread of the COVID-19 coronavirus. The World Health Organization recommends social distancing as a most effective anti-pandemic measure. Historical pandemics are a case in point, including the Spanish 1918 flu pandemic, in which cities implementing social distancing faced lower death rates. Similarly, in the ongoing COVID-19 pandemic, geographical differences in transmission rates and death rates appear to be linked to how firmly social distancing is implemented.

While the effectiveness of social distancing hinges on the cooperation of the public, it may be challenged by the human compulsion for engaging in proximate face-to-face interactions. Therefore, policy-makers need evidence of social distancing compliance to inform policy decisions on whether to implement rules on a voluntary or compulsory basis. So far, however, little is known about the extent to which people keep physical distance from others in everyday practice, not only in the ongoing crisis but during pandemics in general.

Particularly, the study focuses on the extent to which people keep 1.5 meter physical distance from strangers in public space. Of the various social distancing practices (e.g., staying at home, preventive quarantine, avoiding crowds), keeping a physical distance from others is the most directly related to disease transmission. After all, exposure to the virus occurs if people are in physical proximity to a COVID-19 positive individual. Other social distancing measures, such as avoiding non-essential trips outdoors, will only indirectly affect the risk of transmission, because they prevent or reduce the likely hood of physical proximity between people.

2.3 A PROXIMITY-BASED INDOOR NAVIGATION SYSTEM TACKLING THE COVID-19 SOCIAL DISTANCING MEASURES

INFERENCES

In this paper we present an efficient and cost-effective web-based indoor navigation system provides the hints on how to arrive at destination to mobile users equipped with a smartphone. For example, let consider a patient at the hospital that has to arrive at a specific ward. He/she can use a web-app through his/her smartphone to ask for indications towards the destination. In our vision, the web-app should have the following features:

- 1) it should be able to detect the position of the user inside the buildings and track it during the time,
- 2) it should have knowledge of the environment,
- 3) it should be able to calculate the best path between the user and destination optimizing the metric of interest (e.g., the shortest path, the largest way and/or passage, the lowest people density level, and so on) and
- 4) it should drive the user till the destination visualizing a map and providing vocal or iconographic instructions.

From a technical point of view, our solution implements a multi-layer communication infrastructure integrating IoT, Edge and Cloud solutions. At different layers, specific data are collected and processed to match user requirements (e.g., the destination that he/she has to reach), environment requirements (people density, size of passages ,...) and distancing measure regulations for COVID-19. In this paper, since we have to limit the treatment of the proposed solution, we present the reference architecture and provide design and development details on user tracking and the indoor navigation service; algorithms running at the cloud to optimize paths will be investigated in our future works.

We will discuss some experimental results on the communication system according to two types of criteria: objective criteria (i.e., Received Signal Strength

Indicator (RSSI) values) to provide quantitative evaluations of the proposed solution, and subjective criteria (i.e., Mean Opinion Score (MOS)) to analyse the experience in the adoption of the application. All these criteria suitably combined validates and verify the applicability of the presented Indoor Navigation Application.

2.4 PERSON DETECTION FOR SOCIAL DISTANCING AND SAFETY VIOLATION ALERT BASED ON SEGMENTED ROI

INFERENCES

This study focuses on detecting people in areas of interest using the Mobile Net Single Shot Multi box Detector (SSD) object tracking model and OpenCV library for image processing. The distance will be computed between the persons detected in the captured footage and then compared to a fixed pixels' values. The distance is measured between the central points and the overlapping boundary between persons in the segmented tracking area.

With the detection of unsafe distances between people, alerts or warnings can be issued to keep the distance safe. In addition to social distance measure, another key feature of the system is detecting the presence of people in restricted areas, which can also be used to trigger warnings. Some analysis has been performed to test the effectiveness of the program for both purposes.

From the results obtained, the distance tracking system achieved between 56.5% to 68% accuracy for testing performed on outdoor and challenging input videos, while 100% accuracy was achieved for the controlled environment on indoor testing. Whereas for the safety violation alert feature based on segmented ROI, it was found to have achieved better accuracy, i.e. between 95.8% to 100% for all tested input videos.

In addition to social distancing measure, this study also includes detecting people in restricted or dangerous areas that will trigger a warning in the event of safety violation. Heavy transportation pathway, aircraft pathway, personal property, construction area and gas plant can be considered as important or hazardous regions commonly require visual surveillance. Therefore, these areas need to be monitored to reduce the possibilities of people entry that will lead to unwanted incidents.

2.5 MONITORING SOCIAL DISTANCING UNDER VARIOUS LOW LIGHT CONDITIONS WITH DEEP LEARNING AND A SINGLE MOTIONLESS TIME OF FLIGHT CAMERA

INFERENCES

The purpose of this work is to provide an effective social distance monitoring solution in low light environments in a pandemic situation. The raging coronavirus disease 2019 (COVID-19) caused by the SARS-CoV-2 virus has brought a global crisis with its deadly spread all over the world. In the absence of an effective treatment and vaccine the efforts to control this pandemic strictly rely on personal preventive actions, e.g., hand washing, face mask usage, environmental cleaning, and most importantly on social distancing which is the only expedient approach to cope with this situation. Low light environments can become a problem in the spread of disease because of people's night gatherings.

Especially, in summers when the global temperature is at its peak, the situation can become more critical. Mostly, in cities where people have congested homes and no proper air cross-system is available. So, they find ways to get out of their homes with their families during the night to take fresh air. In such a situation, it is necessary to take effective measures to monitor the safety distance criteria to avoid more positive cases and to control the death toll. In this paper, a deep learning-based solution is proposed for the above-stated problem.

The proposed framework utilizes the you only look once v4 (YOLO v4) model for real-time object detection and the social distance measuring approach is introduced with a single motionless time of flight (ToF) camera. The risk factor is indicated based on the calculated distance and safety distance violations are highlighted. Experimental results show that the proposed model exhibits good performance with 97.84% mean average precision (mAP) score and the observed mean absolute error (MAE) between actual and measured social distance values is 1.01 cm.

2.6 A COMPREHENSIVE SURVEY OF ENABLING AND EMERGING TECHNOLOGIES FOR SOCIAL DISTANCING

INFERENCES

This paper aims to provide a comprehensive survey on how emerging technologies, e.g., wireless and networking, artificial intelligence (AI) can enable, encourage, and even enforce social distancing practice. In Part I, an extensive background of social distancing is provided, and enabling wireless technologies are thoroughly surveyed. In this Part, emerging technologies such as machine learning, computer vision, thermal, ultrasound, etc., are introduced.

These technologies open many new solutions and directions to deal with problems in social distancing, e.g., symptom prediction, detection and monitoring quarantined people, and contact tracing. Finally, we discuss open issues and challenges (e.g., privacy preserving, scheduling, and incentive mechanisms) in implementing social distancing in practice. As an example, instead of reacting with ad-hoc responses to COVID-19-like pandemics in the future, smart infrastructures (e.g., next-generation wireless systems like 6G, smart home/building, smart city, intelligent transportation systems) should incorporate a pandemic mode in their standard architectures/designs.

By reducing the frequency and closeness of human physical contacts, social distancing can lower the probability of the disease transmission from an infected person to a healthy one, thereby significantly limiting the disease's spread and severity. During the ongoing COVID-19 pandemic, many governments have implemented various social distancing measures such as travel restrictions, border control, public places closures, and quarantines. Nevertheless, the implementation of such aggressive and large-scale measures is facing significant challenges such as negative economic impacts, personal rights violation, difficulties in changing people's behaviour, and the difficulties arisen when there are many people staying at home. In such context, emerging technologies such as Artificial Intelligence (AI) can play a key role in addressing those challenges.

CHAPTER 3

EXISTING SYSTEM

3.1 IMAGE SEGMENTATION

Computer vision is a field of computer science that enables computers to identify and process objects in videos and images just the way we humans do. Although computer vision might seem like not a very old concept but it dates back to the late 1960s when the first digital image scanner which transformed images into grids of numbers was invented.

You would have probably heard about object detection and image localization. When there is a single object present in an image, we use image localization technique to draw a bounding box around that object. In the case of object detection, it provides labels along with the bounding boxes; hence we can predict the location as well as the class to which each object belongs.

Image segmentation results in more granular information about the shape of an image and thus an extension of the concept of Object Detection.

We divide the images into regions of different colours which helps in distinguishing an object from the other at a finer level

3.2 SEMANTIC SEGMENTATION

Semantic Segmentation is the process of segmenting the image pixels into their respective classes. For example, in the figure above, the cat is associated with yellow colour; hence all the pixels related to the cat are coloured yellow. Multiple objects of the same class are considered as a single entity and hence represented with the same colour.

3.3 INSTANCE SEGMENTATION

Instance segmentation is being more thorough and usually comes into picture when dealing with multiple objects. The difference here is, the detected object is masked with a colour hence all the pixels associated with the image are given the same colour. Multiple objects of the same class are treated as distinct entities and hence represented with different colours.

CHAPTER 4

PROBLEM DESCRIPTION

4.1 PROBLEM STATEMENT

We are detecting a social distance from a random CCTV footage. In this work, YOLOv3 is used for human detection as it improves predictive accuracy, particularly for small-scale objects. The main advantage is that it has adjusted network structure for multiscale object detection.

4.2 OBJECTIVES

A tool to prevent spreading of corona virus (covid-19) by using computer vision on video surveillance. A social distancing analyser tool to regulate social distancing protocol using video surveillance of CCTV Cameras and Drones.

CHAPTER 5

PROPOSED SYSTEM

A tool to prevent spreading of corona virus (COVID-19) by using computer vision on video surveillance. A social distancing analyser tool to regulate social distancing protocol using video surveillance of CCTV cameras and Drones.

Deploying it on current surveillance systems and drones used by police to monitor large areas can help to prevent corona virus by allowing automated and better tracking of activities happening in the area.

METHODOLOGY STEPS:

- Inputs and Outputs of the code
- Working procedure
- Future works

5.1 INPUTS AND OUTPUTS OF THE CODE

5.1.1 Mark necessary points:

Running a program opens a window of the first frame in the video. At this point the code expects the user to mark 6 points by clicking appropriate positions on the frame. These points are necessary to provide a user perspective of the video.

5.1.2 Input: First 4 points:

The first 4 among the 6 required points are used to mark the Region of Interest (ROI) where you want to monitor. Moreover, the lines marked by these points should be

parallel in the real world as seen from above. For example, these lines could be the curbs of the road.

These 4 points need to be provided in a pre-defined order which is the following.

Point1 (bl): Bottom left

Point2 (br): Bottom right

Point3 (tl): Top left

Point4 (tr): Top right

5.1.3 Input: Last 2 points:

The last two points are used to mark two points 6 feet apart in the region of interest.

For example, this could be a person's height (easier to mark on the frame).

5.1.4 Output: Detected pedestrians in original video

The first output of the code is the original video with detected pedestrians in it. The pedestrians are localized with purple boxes.

5.1.5 Output: Social Distancing Parameters

1. **6ft violations:** Number of times the pedestrians violated the 6 feet safe distance threshold
2. **Social-distancing Index:** Quantifies the social distancing being maintained. 50% means half of the interactions violate the safe 6 feet distance criteria.

5.2 WORKING PROCEDURE

Detecting distances between pedestrians from monocular images without any extra information is not possible. One way (not very accurate though) is to ask the user for specific inputs leading to a distance estimation between the pedestrians. If the user could mark two points on the frame that are 6 feet apart, using extrapolation, one could find the distance between different points on the frame. This would have been true if the camera was equidistant to all the points on the plane where the pedestrians were walking. The closer the pedestrians are to the camera the bigger they are. The closer the two points (which are the same number of pixels apart) on the frame to the camera, the smaller is the actual distance between them.

To cope with this issue, the code receives 4 points input from the user to mark two lines that are parallel seen from the above. The region marked by these 4 points are considered ROI (can be seen in yellow in the gif above). This polygon-shaped ROI is then warped into a rectangle which becomes the bird's eye view. This bird's eye view then has the property of points (which are the same number of pixels apart) being equidistant no matter where they are. All it needs is a multiplier that maps the distance between two points in pixels to distance in real life units (such as feet or meters). This is where the last two user input points come into play.

Deep Learning is used to detect and localize the pedestrians which are then mapped to a bird's eye view projection of the camera as explained above. Once we have the coordinates of the pedestrians in the bird's eye view the social distancing parameters become straightforward.

5.3 FUTURE WORKS

The code could use a lot of improvements in several areas. A few are suggested below

- A more accurate approach of mapping the camera frame to a bird's eye view.
- The code uses an existing multi-class classifier that was trained on 1000 classes from the COCO dataset. The code could use a re-trained classifier specifically trained for binary class (pedestrian, no-pedestrian).
- A more accurate way of calculating social distance parameters can be introduced.

CHAPTER 6

RESULT 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. The sequences are depicted from top to bottom. The points represent each pedestrian for social distancing detection. The flow chart is shown in Figure 6.1.

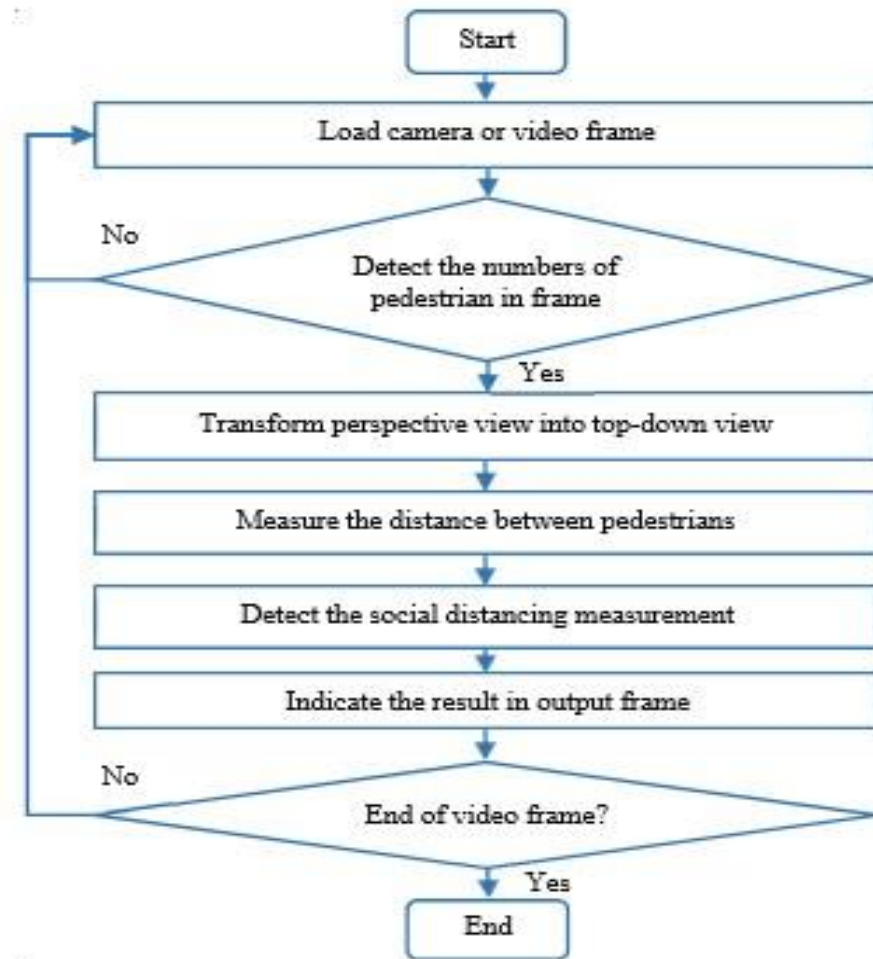


Figure 6.1 Processing Diagram

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. 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. Hence, only the pedestrians walking within the specified space will be counted for people density measurement.

CHAPTER 7

CONCLUSION AND FUTURE WORKS

A methodology of social distancing detection tool using a deep learning 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.

Since this application is intended to be used in any working environment; accuracy and precision are highly desired to serve the purpose. Higher number of false positive may raise discomfort and panic situation among people being observed. There may also be genuinely raised concerns about privacy and individual rights which can be addressed with some additional measures such as prior consents for such working environments, hiding a person's identity in general, and maintaining transparency about its fair uses within limited stakeholders.

7.1 APPENDIX-I

7.1.1 HARDWARE REQUIREMENTS

- OS : Windows 8.1 or above (32 bit or 64 bit)
- RAM : 4GB
- Graphics card : Integrated video card
- Monitor : XGA (1024*768)

7.1.2 SOFTWARE REQUIREMENTS

- Python 3.8
- Pip
- Numpy
- Tensorflow
- Keras
- Open cv2
- Anaconda

7.2 APPENDIX - II

7.2.1 CODE :

```
# import the necessary packages

import time

import math

import cv2

import numpy as np


confid = 0.5

thresh = 0.5

# vname=""

vname=input("Video name in videos folder: ")

if(vname==""):

    vname="Town"

vid_path = "./videos/"+vname+".mp4"

angle_factor = 0.8

H_zoom_factor = 1.2

# Calibration needed for each video


def dist(c1, c2):

    return ((c1[0] - c2[0]) ** 2 + (c1[1] - c2[1]) ** 2) ** 0.5
```

```

def T2S(T):

    S = abs(T/((1+T**2)**0.5))

    return S


def T2C(T):

    C = abs(1/((1+T**2)**0.5))

    return C


def isclose(p1,p2):

    c_d = dist(p1[2], p2[2])

    if(p1[1]<p2[1]):

        a_w = p1[0]

        a_h = p1[1]

    else:

        a_w = p2[0]

        a_h = p2[1]

    T = 0

    try:

        T=(p2[2][1]-p1[2][1])/(p2[2][0]-p1[2][0])

    except ZeroDivisionError:

        T = 1.633123935319537e+16

```

```

S = T2S(T)
C = T2C(T)
d_hor = C*c_d
d_ver = S*c_d
vc_calib_hor = a_w*1.3
vc_calib_ver = a_h*0.4*angle_factor
c_calib_hor = a_w *1.7
c_calib_ver = a_h*0.2*angle_factor
# print(p1[2], p2[2],(vc_calib_hor,d_hor),(vc_calib_ver,d_ver))
if (0<d_hor<vc_calib_hor and 0<d_ver<vc_calib_ver):
    return 1
elif 0<d_hor<c_calib_hor and 0<d_ver<c_calib_ver:
    return 2
else:
    return 0

# load the COCO class labels our YOLO model was trained on
labelsPath = "./coco.names"
LABELS = open(labelsPath).read().strip().split("\n")

np.random.seed(42)

# derive the paths to the YOLO weights and model configuration

```

```

weightsPath = "./yolov3.weights"

configPath = "./yolov3.cfg"


# load our YOLO object detector trained on COCO dataset (80 classes)

print("[INFO] loading YOLO from disk...")

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


# determine only the *output* layer names that we need from YOLO

ln = net.getLayerNames()

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

FR=0


# initialize the video stream and pointer to output video file

vs = cv2.VideoCapture(vid_path)

# vs = cv2.VideoCapture(0) ## USe this if you want to use webcam feed

writer = None

(W, H) = (None, None)


fl = 0

q = 0


# loop over the frames from the video stream

```

```

while True:

    # read the next frame from the file

    (grabbed, frame) = vs.read()

    # if the frame was not grabbed, then we have reached the end
    # of the stream

    if not grabbed:

        break

    if W is None or H is None:

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

        FW=W

        if(W<1075):

            FW = 1075

        FR = np.zeros((H+210,FW,3), np.uint8)

        col = (255,255,255)

        FH = H + 210

        FR[:] = col

```

```

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

net.setInput(blob)

start = time.time()

layerOutputs = net.forward(ln)

end = time.time()

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

boxes = []

confidences = []

classIDs = []

# 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 LABELS[classID] == "person":

    if confidence > confid:

# 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 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)])
    confidences.append(float(confidence))

```

```

        classIDs.append(classID)

# apply non-maxima suppression to suppress weak, overlapping
    # bounding boxes

    idxs = cv2.dnn.NMSBoxes(boxes, confidences, confid, thresh)

# ensure at least one detection exists
    if len(idxs) > 0:

        status = []
        idf = idxs.flatten()
        close_pair = []
        s_close_pair = []
        center = []
        co_info = []

# loop over the indexes we are keeping
        for i in idf:

            # 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

            cen = [int(x + w / 2), int(y + h / 2)]

```

```

center.append(cen)

cv2.circle(frame, tuple(cen),1,(0,0,0),1)

co_info.append([w, h, cen])


status.append(0)

for i in range(len(center)):

    for j in range(len(center)):

        g = isclose(co_info[i],co_info[j])


        if g == 1:

            close_pair.append([center[i], center[j]])

            status[i] = 1

            status[j] = 1

        elif g == 2:

            s_close_pair.append([center[i], center[j]])

            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 = status.count(0)

kk = 0

for i in idf:

    cv2.line(FR,(0,H+1),(FW,H+1),(0,0,0),2)

    cv2.putText(FR, "Social Distancing Analyser wrt. COVID-19", (210, H+60),
                cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 0), 2)

    cv2.rectangle(FR, (20, H+80), (510, H+180), (100, 100, 100), 2)

    cv2.putText(FR, "Connecting lines shows closeness among people. ", (30,
H+100),

                cv2.FONT_HERSHEY_SIMPLEX, 0.6, (100, 100, 0), 2)

    cv2.putText(FR, "-- YELLOW: CLOSE", (50, H+90+40),

                cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 170, 170), 2)

    cv2.putText(FR, "-- RED: VERY CLOSE", (50, H+40+110),

                cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 2)

    #cv2.putText(frame, "-- PINK: Pathway for Calibration", (50, 150),

                #cv2.FONT_HERSHEY_SIMPLEX, 0.5, (180,105,255), 1)

    cv2.rectangle(FR, (535, H+80), (1060, H+140+40), (100, 100, 100), 2)

    cv2.putText(FR, "Bounding box shows the level of risk to the person.", (545,
H+100),

                cv2.FONT_HERSHEY_SIMPLEX, 0.6, (100, 100, 0), 2)

```

```

cv2.putText(FR, "-- DARK RED: HIGH RISK", (565, H+90+40),
            cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 150), 2)
cv2.putText(FR, "-- ORANGE: LOW RISK", (565, H+150),
            cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 120, 255), 2)

cv2.putText(FR, "-- GREEN: SAFE", (565, H+170),
            cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 150, 0), 2)

tot_str = "TOTAL COUNT: " + str(total_p)
high_str = "HIGH RISK COUNT: " + str(high_risk_p)
low_str = "LOW RISK COUNT: " + str(low_risk_p)
safe_str = "SAFE COUNT: " + str(safe_p)

cv2.putText(FR, tot_str, (10, H +25),
            cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 0, 0), 2)
cv2.putText(FR, safe_str, (200, H +25),
            cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 170, 0), 2)
cv2.putText(FR, low_str, (380, H +25),
            cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 120, 255), 2)
cv2.putText(FR, high_str, (630, H +25),
            cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 0, 150), 2)

```

```

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

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

if status[kk] == 1:

    cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 0, 150), 2)


elif status[kk] == 0:

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


else:

    cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 120, 255), 2)


    kk += 1

for h in close_pair:

    cv2.line(frame, tuple(h[0]), tuple(h[1]), (0, 0, 255), 2)

for b in s_close_pair:

    cv2.line(frame, tuple(b[0]), tuple(b[1]), (0, 255, 255), 2)

FR[0:H, 0:W] = frame

frame = FR

cv2.imshow('Social distancing analyser', frame)

cv2.waitKey(1)


if writer is None:

    fourcc = cv2.VideoWriter_fourcc(*"MJPG")

```

```
writer = cv2.VideoWriter("op_"+vname+".avi", fourcc, 30,  
                        (frame.shape[1], frame.shape[0]), True)
```

```
#Writing to video
```

```
writer.write(frame)
```

```
print("Processing finished: open"+"op_"+vname+".avi")
```

```
writer.release()
```

```
#Closes video file or capturing device
```

```
vs.release()
```

7.3 APPENDIX - III

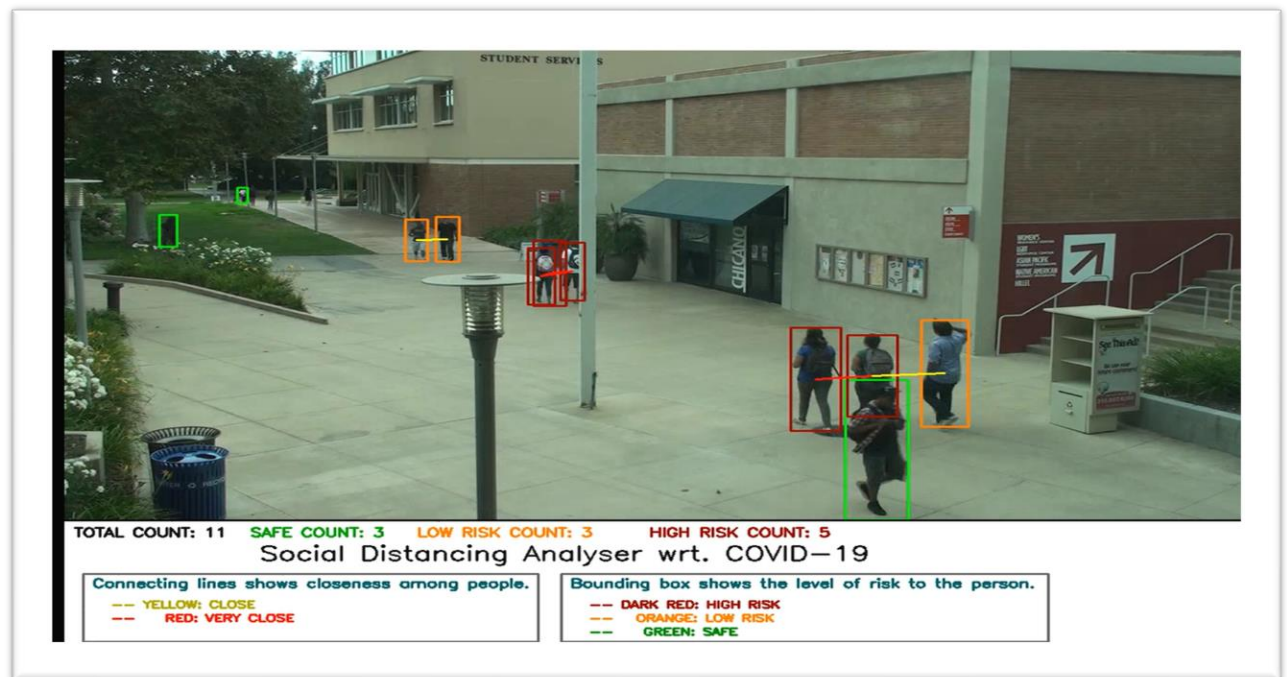


Fig 7.1 Social distance analyser -1

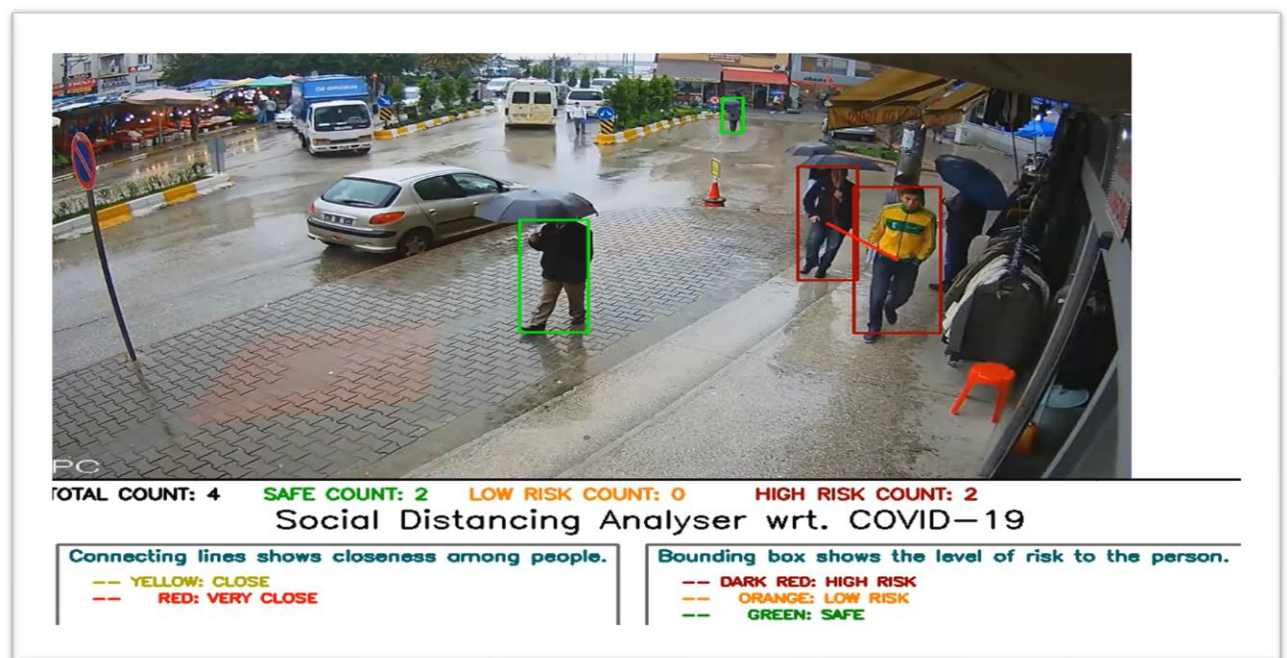


Fig 7.2 Social distance analyser -2



Fig 7.3 Social distance analyser -3

REFERENCES

1. Centers for Disease Control (CDC). Implementation of Mitigation Strategies for Communities with Local COVID-19 [Online]. Available at <https://www.who.int/emergencies/diseases/novel-coronavirus-2019> (Accessed 8 May 2020).
2. D.T. Nguyen, W. Li, P.O. Ogunbona, "Human detection from images and videos: A survey", *Pattern Recognition*, 51:148-75, 2016.
3. J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database", In *Computer Vision and Pattern Recognition*, 2009.
4. J. Redmon, S. Divvala, R. Girshick, A. Farhadi, "You only look once: Unified, realtime object detection", In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779-788. 2016.
5. Krizhevsky, I. Sutskever, G.E. Hinton, "Imagenet classification with deep convolutional neural networks", In *Advances in neural information processing systems*, pp. 1097-1105, 2012.
6. 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.
7. K. Simonyan, A. Zisserman, "Very deep convolutional networks for large-scale image recognition", *arXiv preprint arXiv:1409.1556*, 2014.
8. Landing AI Creates an AI Tool to Help Customers Monitor Social Distancing in the Workplace [Online]. 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).

9. Ministry of Health Malaysia (MOHM) Official Portal. COVID-19(Guidelines) [Online]. Available at <https://www.moh.gov.my/index.php/pages/view/2019-ncovwuhanguidelines>(Accessed 8 May 2020).
10. R. Girshick, J. Donahue, T. Darrell, J. Malik. "Rich feature hierarchies for accurate object detection and semantic segmentation." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 580-587. 2014.
11. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, "Rethinking the inception architecture for computer vision", In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2818- 2826, 2016.