De-Socialize

(Social Distancing Analyzer)

A PROJECT REPORT

Submitted by

Aakash Mattoo	19BAI10152
Abhinav Kumar Singh	19BAI10078
Sachin Sharma	19BAI10082
Piyush Saraf	19BAI10041

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

Specialization in

Artificial intelligence and machine learning

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

VIT BHOPAL UNIVERSITY
KOTHRIKALAN, SEHORE
MADHYA PRADESH – 466114

APRIL 2021

VIT BHOPAL UNIVERSITY,KOTHRIKALAN, SEHORE
MADHYA PRADESH – 466114

BONAFIDE CERTIFICATE

Certified that this project report titled **"De-Socialize (Social Distancing Analyzer)"** is the bonafide work of "Aakash Mattoo, Abhinav Kumar Singh, Sachin Sharma and Piyush Saraf" who carried out the project work under my supervision. Certified further that to the best of my knowledge the work reported here does not form part of any other project / research work on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

PROGRAM CHAIR
Dr S Sountharrajan,
Senior Assistant Professor
School of AI &ML division
VIT BHOPAL UNIVERSITY

PROJECT GUIDE
Dr. Ashish Kumar Sahu
Assistant Professor
School of AI &ML division
VIT BHOPAL UNIVERSITY

ACKNOWLEDGEMENT

First and foremost, we would like to thank the Lord Almighty for his presence and immense blessings throughout the project work.

We wish to express our heartfelt gratitude to Dr. Sountharrajan S, Program Chair - B.Tech CSE Spl in Artificial Intelligence and Machine Learning, for much of his valuable support encouragement in carrying out this work.

We would like to thank our Project Coordinator, Dr. Ashish Kumar Sahu for continually guiding and actively participating in my project, giving valuable suggestions to complete the project work.

We would like to thank all the technical and teaching staff of the School of Computing Science and Engineering, who extended directly or indirectly all support.

A heartfelt gratitude to Dr. Nageswara Guptha, Division Head (Artificial Intelligence and Machine Learning), who provided us the possibility to complete this course.

Last, but not the least, we are deeply indebted to our respective parents who have been the greatest support while we worked day and night for the project to make it a success.

***

TABLE OF CONTENTS

***

LIST OF ABBREVIATIONS

1	**SSD**	Single Shot Detector
2	**mAP**	mean Average Precision
3	**Fps**	Frames per second
4	**HOG**	Histograms of Oriented Gradient
5	**SVM**	Support Vector Machine
6	**IoU**	Intersection over Union
7	**SS**	Selective Search
8	**RPN**	Region Proposal Network
9	**CNN**	Convolutional Neural Networks
10	**RoI**	Region of Interest
11	**OID**	Open Image Dataset
12	**TT**	Training Time
13	**NoI**	Number of Iterations
14	**TL**	Total Loss
15	**YOLO**	You Only Look Once
16	**PIR**	Passive Infrared Technology
17	**COCO**	Common Object In Context
18	**TP**	True Positive
19	**TN**	True Negative
20	**FP**	False Positive
21	**FN**	False Negative
22	**MAE**	Mean Absolute Error

***

LIST OF TABLES

***

LIST OF FIGURES

***

ABSTRACT

With its deadly spread to more than 180 nations, the coronavirus disease 2019 (COVID-19) has caused a global crisis. This has been declared a pandemic by the World Health Organization. As a result of this situation, the international community is scrambling to find ways to halt the spread of this contagious virus. In the current situation, social distancing is believed to be the strongest spread stopper, and all affected countries have agreed to adopt Social Distancing. Its aim is to reduce physical contact between potentially infected people and healthy people. In order to obey social distancing, the WHO recommended that people maintain a gap of at least 6 feet between them.

Human detection using visual surveillance systems is a well-established field of research that relies on manual methods for detecting suspicious activities, but it has limitations. In the field of video surveillance, there are numerous studies available. KTH human motion dataset contains six categories of events, while INRIA XMAS multi-view dataset and Weizmann human action dataset contain 11 and 10 categories of actions, respectively, among several publicly available datasets. Recent advances in this direction advocate the need for intelligent systems to identify and record human activities. Due to a number of constraints such as low-resolution video, varying articulated posture, clothing, lighting and background nuances, and minimal machine vision capabilities, human detection is an ambitious goal. However, prior awareness of these obstacles will increase detection efficiency.

The aim is to reduce physical contact between potentially infected people and healthy people. In order to obey social distancing, the WHO recommended that people maintain a gap of at least 6 feet between them. After the outbreak of the novel coronavirus, several countries have relied on technology-based solutions in various capacities to control the outbreak. As an outcome, our team has decided to work on proper implementation of the Social Distancing rules and compare the success of common object detection and tracking schemes in monitoring the social distancing in this current work.

***

Chapter 1
INTRODUCTION

1.1 – Introduction

The word "social distancing" has taken the world by storm and is changing the way we live. Distancing oneself from others has become a mantra that transcends languages and cultures all over the world. COVID-19, the world's fastest-growing pandemic, has forced us to adapt to this new way of life. COVID-19 has infected nearly 4 million people, according to the World Health Organization (WHO).
By definition, people cannot spread germs if they are not close together, so social distancing is arguably the most successful nonpharmaceutical way to prevent disease spread.
About 1000K+ people have died as a result of the outbreak all over the world. So far, the lethal virus has infected people in 213 countries. COVID-19 spreads from person to person by touch or near proximity to an infected person, which is the main source of concern. Given how heavily populated certain places are, this has been a difficult task.

1.2 – Motivation for the Work

We are constantly told in India to avoid physical contact and keep a distance of at least one metre between ourselves and others. Prime Minister Narendra Modi has frequently mentioned it in his nationally broadcasted addresses to the people. He said – "If you want to tame the spread of the coronavirus, the only way to do it is to break the cycle of transmission by practicing social distancing. "

There's also the issue of overcrowding to consider. India has a population density of 464 people per square kilometre, compared to 153 in China, the world's most populated country, and 36 in the United States. In India, the average family has five members, and 40% of all homes - or 100 million - have only one bed.
There are many drowsiness detection systems, that are effective, but they have few drawbacks, and the methodology isn't effective. To detect Driver Drowsiness, we are using *TensorFlow object detection API*, where in we are *using Faster RCNN Algorithm* to detect closed eyes(drowsy) and open eyes (not drowsy). Our Model is very effective

as it can give accuracy of 95%. As this accuracy is very good, this model can help to reduce the percentage of accidents caused by drowsiness of driver. We will get to know about the existing work and their disadvantages further in the next chapter.

1.3 – Problem Statement and Objective of the work

Distancing oneself from others is not a modern concept; it dates back to the fifth century (source) and is even mentioned in religious texts. By definition, people cannot spread germs if they are not close together, so social distancing is arguably the most successful nonpharmaceutical way to prevent disease spread.
Our findings of positive net benefits suggest that this response is preferable to taking no action to combat the outbreak. Although there could be other policy combinations that could be implemented in the event of a pandemic or in the future, we'll leave that to future research. Second, we are solely concerned with calculating the total net benefits of social distancing, which means we are oblivious to the possible distributional effects of such policies. It stands to reason that the most disadvantaged members of society will bear the brunt of the consequences.

*****

Chapter 2
LITERATURE SURVEY

2.1 – Literature Review

With the assumption that social distancing is the most reliable strategy for preventing the spread of infectious disease, it was chosen as an unprecedented step on January 23, 2020, against the backdrop of December 2019, when COVID-19 emerged in Wuhan, China. The outbreak in China reached a peak in the first week of February, with 2,000 to 4,000 new confirmed cases every day, in less than a month. For the first time since the outbreak began, there was a sign of relief when no new confirmed cases were recorded for five days in a row, from March 23 to March 24, 2020. This is evident in the use of social distancing steps in China, which were later implemented globally to regulate COVID-19.

Controlling the spread of infectious diseases is done by social distancing. As the name indicates, social distancing entails people physically separating themselves from one another in order to reduce close contact and hence the spread of an infectious disease (such as coronavirus). It was also proposed that lifting social distancing too quickly could lead to an earlier secondary peak, which could be flattened by gradually easing the interventions. As we all know, social distancing is an essential yet financially costly step to flatten the infection curve.

All of the reviewed literature and related research work paints a good image of how human identification can be effectively applied to a variety of applications to address the current situation, such as checking recommended hygiene requirements, social distancing, work practises, and so on. A Sliding Window method is used to build an Object Detection model. An image is divided into regions of varying sizes, and each region is then grouped into the appropriate groups, as the name implies. This device can be integrated into existing surveillance camera systems in a variety of public locations. When people get too close to each other, an alarm goes off (less than 6 feet). The programme that is used to monitor people's activities in a security framework must have clear ethical and privacy consequences, or else the data associated with this system will be vulnerable to hacking.

2.2 – Existing Algorithms

After the outbreak of the novel coronavirus, several countries have relied on technology-based solutions in various capacities to control the outbreak. Existing Social Distancing Analyzer was a human-intervention device in which different designated individuals would go through all of the CCTV surveillance recordings and then use manual analytics to determine which areas of the state had the most violations of the social distancing standard. And, with the assistance of the appropriate authorities, they will take the requisite steps to put an end to the breach of the standard. Then, with the assistance of the appropriate authorities, they will take the requisite steps to avoid the breach of the norm in the red-zoned areas. When faced with knowns and unknowns, we humans have a propensity to behave differently. We have a tendency to be liberal towards our acquaintances, but we take the norm for granted. A model, on the other hand, will not have this propensity and will thus treat all the same.

2.3 – Shortcomings of Previous Work

There are a few free social distancing analyzers available, but the object detectors used in these are sluggish and unreliable. Human interference would lead to judging bias and, as a result, issues. Humans will concentrate on something new after doing something boring. Machines never get tired of doing the same thing over and over again. The authorities' warnings to keep their distance would have a negative effect on the public's perception. Instead, there would be no negative emotional reaction if an alarm was sounded. Some areas can be left unregulated due to human inefficiency. Such errors will not be made by machines. Because of people's night meetings, low light conditions can become a problem in the spread of disease. The situation can become even more urgent during the summer, when the global temperature is at its highest. Typically, in cities where people live in congested housing and there is no proper air cross-system in place. As a result, they devise strategies to get out of their homes with their families late at night to take advantage of the opportunity. As a result, there is a need to devise strategies for model to work at night. In such a scenario, appropriate steps to track the safety distance requirements are needed to prevent more positive cases and to keep the death toll under control. For all the aforementioned problem, a deep learning-based solution needs to be proposed and worked upon.

*****

Chapter 3
PROPOSED WORK

3.1 – Proposed Methodology

This report presents a method to design and develop an accurate, cost-effective, and robust social distancing analyzer for real monitoring purposes. Main objective is to develop a model based on deep learning architecture.

Firstly, we need to collect the dataset, various images of persons. After Gathering all the required data we have to do the annotation of the images using **Labelimg**. XML files will be created for the annotation of each image.

Then we need to divide the dataset into train and test folders. We have to choose the model for training, There are various state of the art DNN models available for Object detection like **RCNN, Faster RCNN, SSD** and different versions of **YOLO.SO** We have to choose the best once we get to know about YOLO.

3.1.1 – The YOLO technique

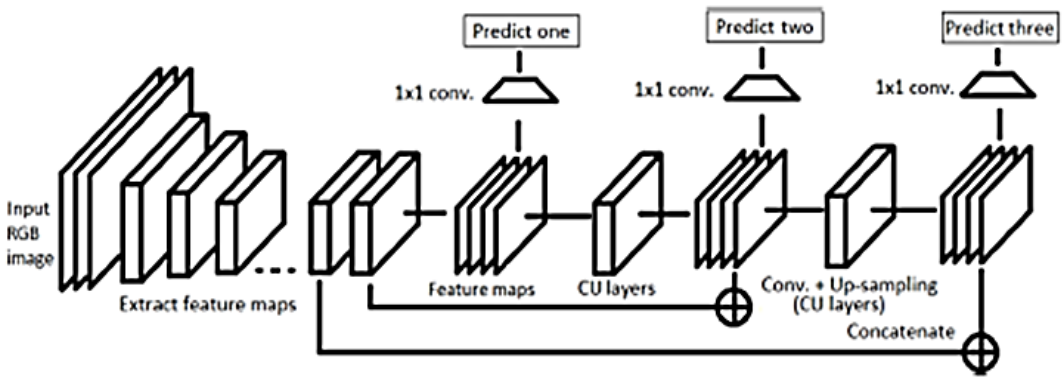


Figure 1 – Schematic representation of YOLO v3 architecture

As the name suggests, YOLO can predict the type and location of an object by looking only once at the image. YOLO considers the object detection problem as a regression task instead of classification to assign class probabilities to the anchor boxes. Instead of selecting the interesting regions from the image, YOLO takes the problem of object detection as the regression problem where the object detection and classification take place in a single neural network. This network is extremely fast, it processes images in real-time at 45 frames per second. A smaller version of the network, Fast YOLO, processes an astounding 155 frames per second.

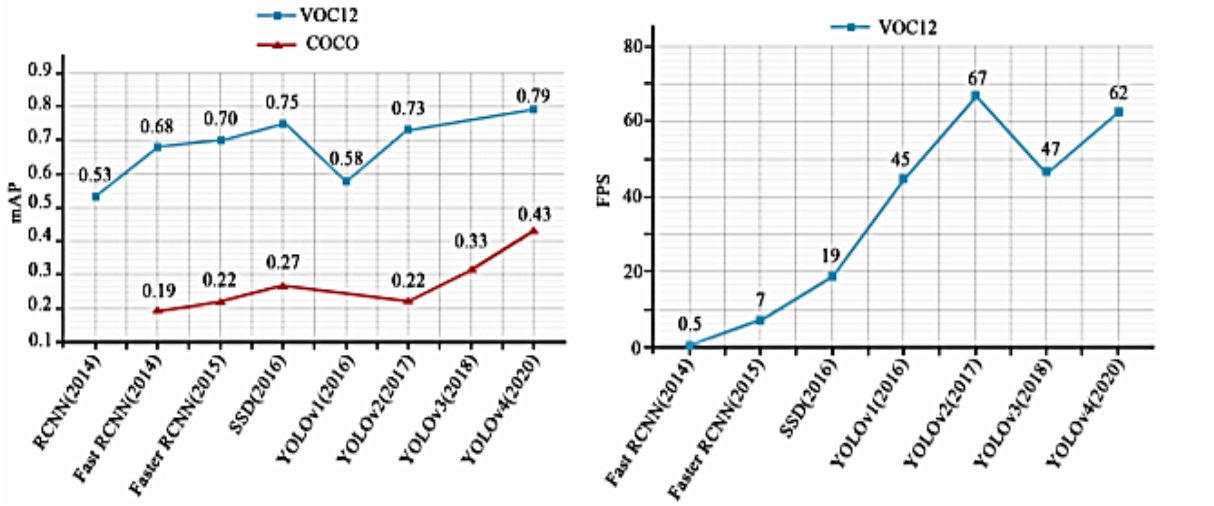


Figure 2 – Mean Average Precision (mAP) and Speed (FPS) overview of eight most popular object detection models on Microsoft Common Objects in Context (MS-COCO) and PASCAL Visual Object Classes (VOC) datasets.

From the figure it is clear that YOLOv4 offers the best trade-off for the speed and the accuracy for a multi-class object detection purpose. However we will use YOLOv3 model in our project as it has perfect balance between processing time and accuracy. YOLO Is pretrained on more than 80 classes in which person is also there so we don't need to train our model.

3.1.2 – Using YOLO

For using **YOLOv3** we have to download the pretrained weight, its **config file** and the **.name** file (contains the details of trained classes). After that we have to feed each frame of our input video stream/web cam to our model. We have to extract only person class from the model. This can be done by using **'personIdx'** function. Then we have to extract the co-ordinates of the boundary boxes drawn over persons in a frame. This can be implemented using **numpy.array().**

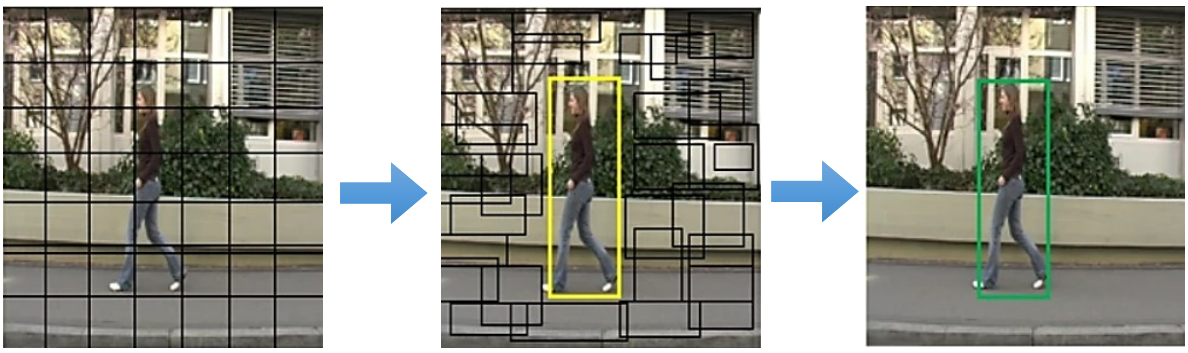
Then calculate the centroids of the boundary boxes of persons. This will be done by **numpy method.** Now we have to calculate the Euclidian Distance between each centroids pairs in a frame. For this we have to use Distance metric method contained in the scipy.spatial module.

$$d(\mathbf{p},\mathbf{q}) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_i - q_i)^2 + \cdots + (p_n - q_n)^2}.$$

After that we have to compare each calculated distance with a threshold pixel and if less then make the boundary box Red otherwise make its color to green. We have to make a counter which will increase whenever calculated distance will be less then the threshold.

Print counter value on screen for each frame. This will represent the number of violations. If input is video stream then Combine all frames in a video stream and write it to the memory using **OpenCV**. If input video stream is from the webcam, then display each of the processed frames in a real time

It is impossible to detect distances between pedestrians from monocular images without any additional details. One method (though not quite accurate) is to ask the user for precise inputs before estimating the distance between the pedestrians. Extrapolation may be used to find the distance between various points on the frame if the user could mark two points on the frame that are 6 feet apart.



S x S on input image Bounding boxes & confident score Final Detection

Figure 3 – The working of YOLO model

This would have been valid if the camera had been equidistant from all of the pedestrians' walking points on the plane. The larger the pedestrians get as they get closer to the camera. The real distance between two points on the frame that are the same number of pixels apart is smaller the closer they are to the sensor.

3.1.3 – System Architecture

A system architecture is the conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviors of the system.

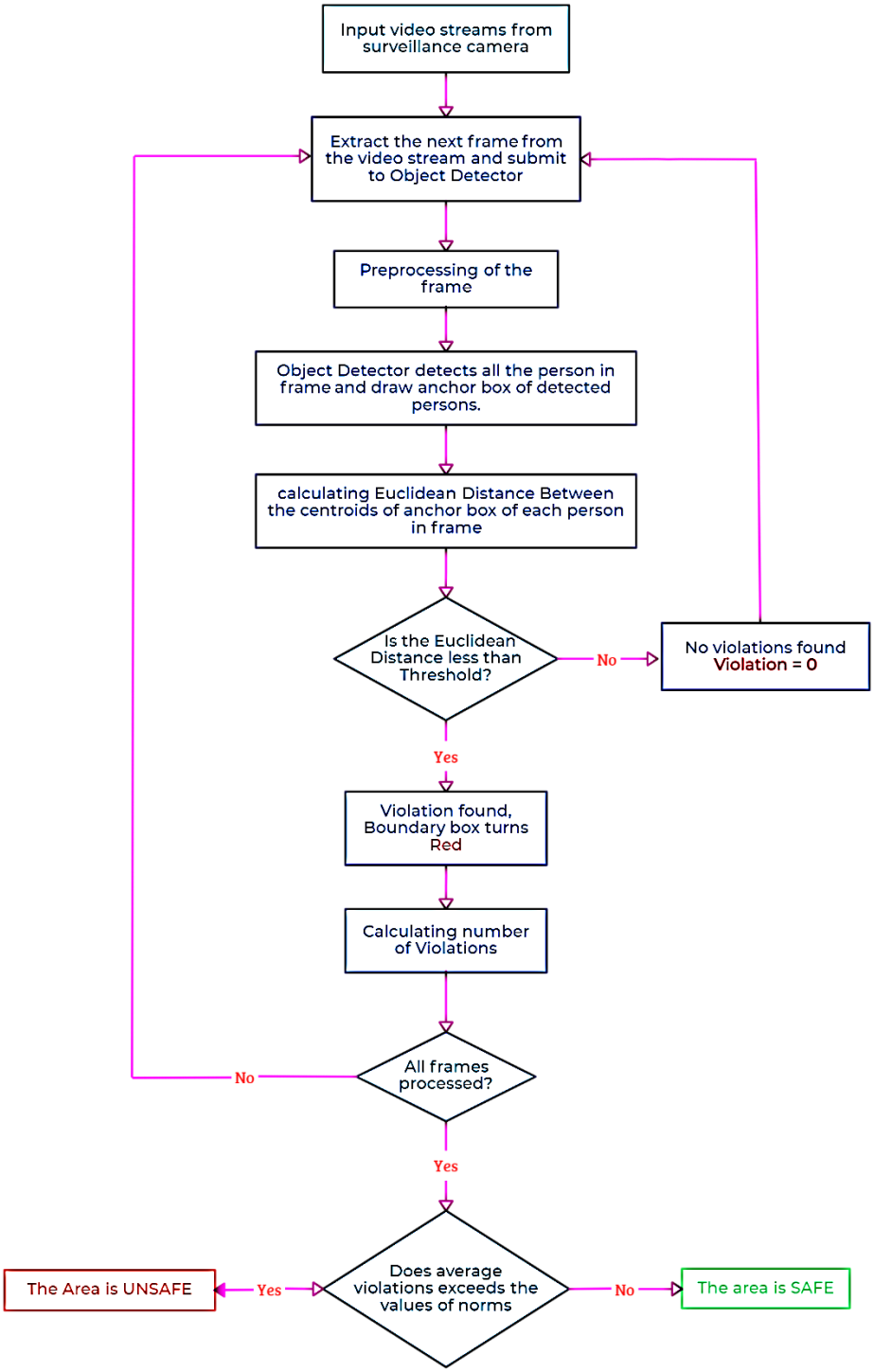


Figure 4 – The System Architecture of the project

3.2 – Requirements

The method for Social Distancing Analyzer mentioned in this paper must comply with the relevant key requirements:

3.4.1 – Hardware Requirements

- ❑ CCTV surveillance cameras spread across important and crowded streets for getting input data
- ❑ Interconnecting wires or network, bringing live streamed data from CCTV to the model.
- ❑ Alarms on each CCTV for raising alert.
- ❑ High power computation unit GPU should be Geforce GTX 1080 (4GB)
- ❑ Minimum RAM – 4GB DDRM CORE: i5 minimum 1.6 GHz or faster processor.

3.4.2 – Software Requirements

- ❑ Software Needed :
  - OpenCV 3
  - PIL
  - Python3
  - Darknet
  - Numpy
- ❑ A model to process the input data and give out necessary signal (Red or Green zone)
- ❑ A database for storing the area location and camera number to identify the coordinates of red zone.

*****

Chapter 4
MODULE DESCRIPTION

4.1 – Introduction to Modules

The purpose of these modules is to introduce the reader to the workflow of the modules in a sequential manifold. We start by explaining how to prepare our workspace and get our systems ready with all the necessary hardware and software requirements. Then, we perform a detailed study of the required dataset and then collect and prepare it with utmost importance. After introducing Workspace and preparing data, we will begin with training our model, once the model is trained using all the required parameters, will test it's working and self-evaluate, like using the live webcam in our case. The following detailed modules will give a better overview of the work performed.

Our project is widely classified into the following Modules :

4.1.1 – Preparing Workspace
4.1.2 – Person Detection (YOLO)
4.1.3 – Data Computation
4.1.4 – Evaluating Model

4.2 – Module Workflow

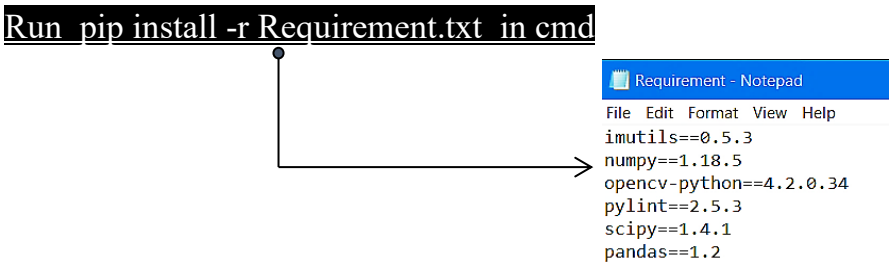
A workflow process will have the above modules being divided into the number of steps and each step may have one or more actions and approvals associated with it.

4.2.1 – Preparing Workspace

- Creating a folder in any location named Social Distancing Analyzer.

- Creating an Environment in Anaconda with Python (v. 3.6.9) in the above directory.

- Installing required libraries by using pip installer



- Download all the necessary files required like yolo.config, coco.name and the pretrained yolo3.weight files.
  - Use the command git clone github.com/AlexeyAB/darknet.

Use https://pjreddie.com/media/files/yolov3.weights for pretained weight !

4.2.2 – Person Detection (YOLO)

In this module we will focus on the detection of person in the frame.\

- Extract and pre-process the frames from the input video stream.

```
vs = cv2.VideoCapture(args["input"]
if args["input"] else 0)
while True: (grabbed, frame) = vs.read() #for extracting each frame
```

- Next we will load the pretrained yolov3 object detector.

```
weightsPath = os.path.sep.join([config.MODEL_PATH, "yolov3.weights"])
configPath = os.path.sep.join([config.MODEL_PATH, "yolov3.cfg"])
net = cv2.dnn.readNetFromDarknet(configPath, weightsPath)
```

- We Only need The person class to be detected.

```
LabelsPath = os.path.sep.join([config.MODEL_PATH, "coco.names"])
LABELS = open(labelsPath).read().strip().split("\n")
LABELS.index("person")
```

- Now Each frame is to feed into the pretrained yolov3 object detector which will output boundary boxes around person in each frame with their confidence score

Figure 5 – Capturing of frames by the model

4.2.3 – Data Computation

- The Output of YOLO will be centroids coordinates(x,y), confidence score of prediction and width, height of the boundary box for each prediction.

- For confidence Score less than 0.3 we will ignore the detection with help of if statement .
  We will calculate the top Left and Bottom Right co-ordinates of boundary box for further Computation.
  x = int(centerX - (width / 2)) y = int(centerY - (height / 2))

- We will Calculate each pair of centroid Distance in each frame.

- Methods For Calculating Distances are –
  - Minkowski
  - Euclidean
  - Hamming

  We will choose Euclidean Distance because of its simplicity.

- After That ***D*** is compared with the threshold value. This value will vary based on the angle view of camera.
  Webcam : Threshold = 200-300
  CCTV: in range 40-50px.
- We will increase a created counter in *D<Threshold*

```
for i in range(0, D.shape[0]):
   for j in range(i+1, D.shape[1]):
       if D[i, j] < config.MIN_DISTANCE:
       violate.add(i)
       violate.add(j)
```

- We will Assign red color boundary to violations and green to non violations.
  if i in violate: color = (0, 0, 255

- We will print no. of violations for each frame
  text = "Social Distancing Violations: {}".format(len(violate))

4.2.4 – Evaluating Model

In order to evaluate the model, we need to convert the ckpt file that was generated after training the model to inference_graph.py file. This file is used for evaluating the model via webcam.

For this we would require two person and we will try to evaluate our model using the laptop webcam. We would continuously move closer and farther to check the real time efficiency and the accuracy of the code.

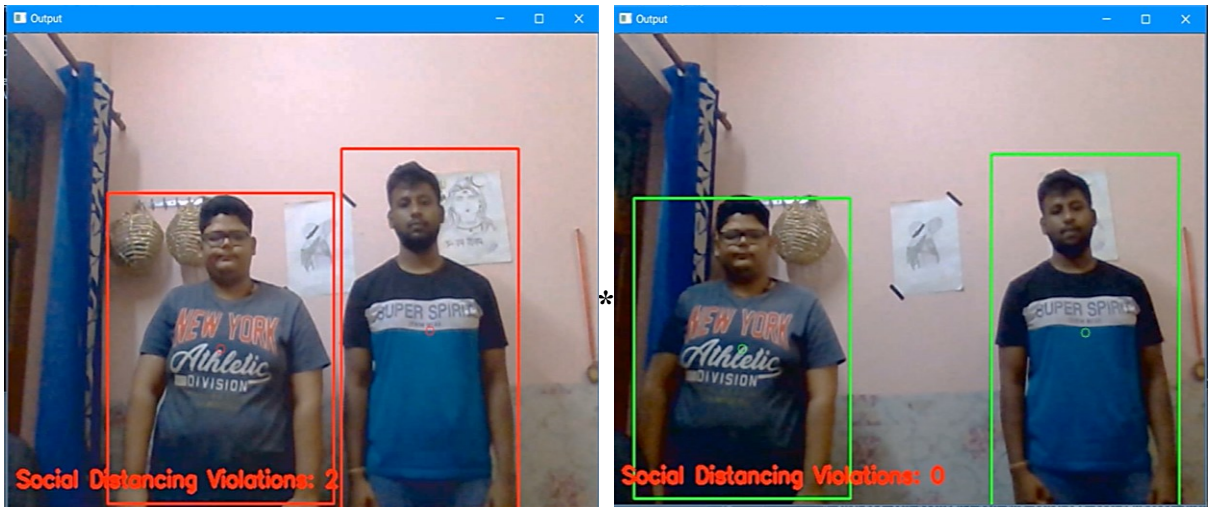


Figure 6 – Evaluation with and without violations

*****

Chapter 5
WORK DONE AND TESTING

5.1 – Coding and Explanation

We have using Python Programming Language for coding. Below are various coding excerpt for the working of the model. The code takes the video file path as input. Running main.py will open a window of the first frame in the video and subsequently it would calculate distances between the frames of person and detect whether social distancing norms are followed.

5.1.1 – coding.py

```
# base path to YOLO directory
MODEL_PATH = "yolo-coco"

# initialize minimum probability to filter weak detections along
with the
# threshold when applying non-maxim suppression
MIN_CONF = 0.3
NMS_THRESH = 0.3

# for using gpu
USE_GPU = True

# define the minimum safe distance (in pixels) that two people can
be from each other
MIN_DISTANCE = 50
```

5.1.2 – detection.py

```
# imports
from .config import NMS_THRESH
from .config import MIN_CONF
import numpy as np
import cv2

# function to detect people
def detect_people(frame, net, ln, personIdx=0):
```

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

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

    # construct a blob from the input frame and then perfrom a
forward pass
    # of the YOLO object detector, giving us the bounding boxes and
    # associated probabilities
    blob = cv2.dnn.blobFromImage(frame, 1 / 255.0, (416, 416),
swapRB=True, crop=False)
    net.setInput(blob)
    layerOutputs = net.forward(ln)

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

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

            # filter detections by (1) ensuring that the object
detected was a person and
            # (2) that the minimum confidence is met
            if classID == personIdx and confidence > MIN_CONF:
                # scale the bounding box coordinates back relative
to the size of
                # the image, keeping in mind that YOLO actually
returns the center (x, y)-coordinates
                # of the bounding box followed by the boxes' width
and height
                box = detection[0:4] * np.array([W, H, W, H])
                (centerX, centerY, width, height) =
box.astype("int")

                # use the center (x,y)-coordinates to derive the
top and left corner of

                # the bounding box
                x = int(centerX - (width / 2))
                y = int(centerY - (height / 2))
```

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

    # apply non-maxima suppression to suppress weak, overlapping
bounding boxes
    idxs = cv2.dnn.NMSBoxes(boxes, confidences, MIN_CONF,
NMS_THRESH)

    # ensure at least one detection exists
    if len(idxs) > 0:
        # loop over the indexes being kept
        for i in idxs.flatten():

# extract the bounding box coordinates
            (x, y) = (boxes[i][0], boxes[i][1])
            (w, h) = (boxes[i][2], boxes[i][3])

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

    # return the list of results
    return results
```

5.1.3 – social_distancing_detector.py

```
# imports
from configs import config
from configs.detection import detect_people
from scipy.spatial import distance as dist
import numpy as np
import argparse
import imutils
import cv2
import os

# construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--input", type=str, default="", help="path
to (optional) input video file")
ap.add_argument("-o", "--output", type=str, default="", help="path
to (optional) output video file")
```

```
ap.add_argument("-d", "--display", type=int, default=1,
help="whether or not output frame should be displayed")
args = vars(ap.parse_args())

# load the COCO class labels the YOLO model was trained on
labelsPath = os.path.sep.join([config.MODEL_PATH, "coco.names"])
LABELS = open(labelsPath).read().strip().split("\n")

# derive the paths to the YOLO weights and model configuration
weightsPath = os.path.sep.join([config.MODEL_PATH,
"yolov3.weights"])
configPath = os.path.sep.join([config.MODEL_PATH, "yolov3.cfg"])
# load the YOLO object detector trained on COCO dataset (80
classes)
print("[INFO] loading YOLO from disk...")
net = cv2.dnn.readNetFromDarknet(configPath, weightsPath)

# check if GPU is to be used or not
if config.USE_GPU:
    # set CUDA s the preferable backend and target
    print("[INFO] setting preferable backend and target to
CUDA...")
    net.setPreferableBackend(cv2.dnn.DNN_BACKEND_CUDA)
    net.setPreferableTarget(cv2.dnn.DNN_TARGET_CUDA)

# determine only the "output" layer names that we need from YOLO
ln = net.getLayerNames()
ln = [ln[i[0] - 1] for i in net.getUnconnectedOutLayers()]
# initialize the video stream and pointer to output video file
print("[INFO] accessing video stream...")
# open input video if available else webcam stream

vs = cv2.VideoCapture(args["input"] if args["input"] else 0)
writer = None

# loop over the frames from the video stream
while True:
    # read the next frame from the input video
    (grabbed, frame) = vs.read()

    # if the frame was not grabbed, then that's the end fo the
stream
    if not grabbed:
        break
```

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

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

        # if the index pair exists within the violation set, then
update the color
        if i in violate:
            color = (0, 0, 255)

        # draw (1) a bounding box around the person and (2) the
centroid coordinates of the person

        cv2.rectangle(frame, (startX, startY), (endX, endY),
color, 2)
        cv2.circle(frame, (cX, cY), 5, color, 1)

    # draw the total number of social distancing violations on the
output frame
    text = "Social Distancing Violations: {}".format(len(violate))
    cv2.putText(frame, text, (10, frame.shape[0] - 25),
cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 0, 255), 3)
# check to see if the output frame should be displayed to the
screen
    if args["display"] > 0:
  # show the output frame
        cv2.imshow("Output", frame)
        key = cv2.waitKey(1) & 0xFF

        # if the 'q' key is pressed, break from the loop
        if key == ord("q"):
```

```
            break
# if an output video file path has been supplied and the video
writer ahs not been
    # initialized, do so now
    if args["output"] != "" and writer is None:
        # initialize the video writer
        fourcc = cv2.VideoWriter_fourcc(*"MJPG")
        writer = cv2.VideoWriter(args["output"], fourcc, 25,
(frame.shape[1], frame.shape[0]), True)

    # if the video writer is not None, write the frame to the
output video file
    if writer is not None:
        print("[INFO] writing stream to output")
```

5.2 – Live Testing

After the implementation of the code, we advanced towards the live testing via the CCTV camera. The first output of the code is the original video with detected pedestrians in it. The pedestrians are localized with green boxes.
The code detected the persons in frame, calculated their distance and detected whether the social distancing is violated or not with great precision.

Figure 7 – Live Testing via a CCTV camera

*****

Chapter 6
CONCLUSION AND RESULTS

6.1 – Experimental Results

Our Project uses deep learning to monitor social distancing in real-time. It monitors the same using object detection and tracking approach with the assistance of bounding boxes. Thereafter the cluster of peoples are identified and checked whether the threshold distance between them is maintained or not aided by the bounding boxes again. Take into consideration that this approach is highly sensitive to the spatial location of the camera, to fine tune adjust the corresponding field of view.

6.1.1 – Evaluation Measure

Precision, Recall, and AP are three important evaluation measures for object detectors. The following describes the significance of these metrics in the sense of individual identification in low-light situations. Precision indicates how well the model predicted the people. The amount of truly detected people over the total of truly detected people and undetected people in an image is referred to as **recall**.
After any true object has been found, AP is the **mean of the precision score**. It considers how well the object detection algorithms work. AP, which is equivalent to mAP in COCO detection metrics, is used as an assessment indicator in this study due to its comprehensive assessment ability.

6.1.1 –Performance Evaluation

COCO detection metrics are used to measure the efficiency of the trained model after a series of experiments. Table 1 displays precision (Prec), recall (Rec), F1-score, false positives (FP), true positives (TP), false negatives (FN), and mAP at two different network sizes (320, 416) with IoU thresholds of 0.5, 0.75, and 0.5:0.95 at two different network sizes (320, 416). The TP, FP, and F1-score are used to measure precision and recall.
The TP, FP are used to measure precision and recall, while the F1-score is determined using the precision and recall resultant values. The model had overall good efficiency,

with network size 416 with IoU threshold 0.5 having the highest mAP value of 97.84 percent, according to the evaluation results based on the mAP.
At two network sizes, the precision-recall curve (PR-curve) of COCO evaluation at the IoU threshold varies from 0.5 to 0.95.

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

To evaluate the performance of our social distance monitoring solution, we perform few tests at three different fixed camera distances 400 cm, 500 cm, and 600 cm. *Table 6.1* summarises the quantitative findings in terms of object distance in pixels and centimetres, real known distance in centimetres, and per test error rate.

Table 6.1 – Experimental Results

Test	C_d	k	PD (pixels)	Du (cm)	Ad (cm)	AE (cm)
D_{T1T2} = 100 cm	400 cm	0.34236	292.098	-	-	-
1	-	-	153.0	52.3	52	0.3
2	-	-	414.1	141.8	140	1.8
D_{T1T2} = 100 cm	500 cm	0.40635	246.099	-	-	-
3	-	-	131.1	53.3	52	1.3
4	-	-	340.3	138.3	140	-1.7
D_{T1T2} = 100 cm	600 cm	0.49022	203.994	-	-	-
5	-	-	107.5	52.7	52	0.7
6	-	-	285.0	139.7	140	-0.3
Mean Absolute Error (MAE) = 1.01 cm						

We can see that our model exhibited overall good performance. People violating the safety distance are highlighted by red bounding boxes; whereas, green bounding boxes show people following safety distance criteria. The code detected the persons in frame, calculated their distance and detected whether the social distancing is violated or not with great precision.

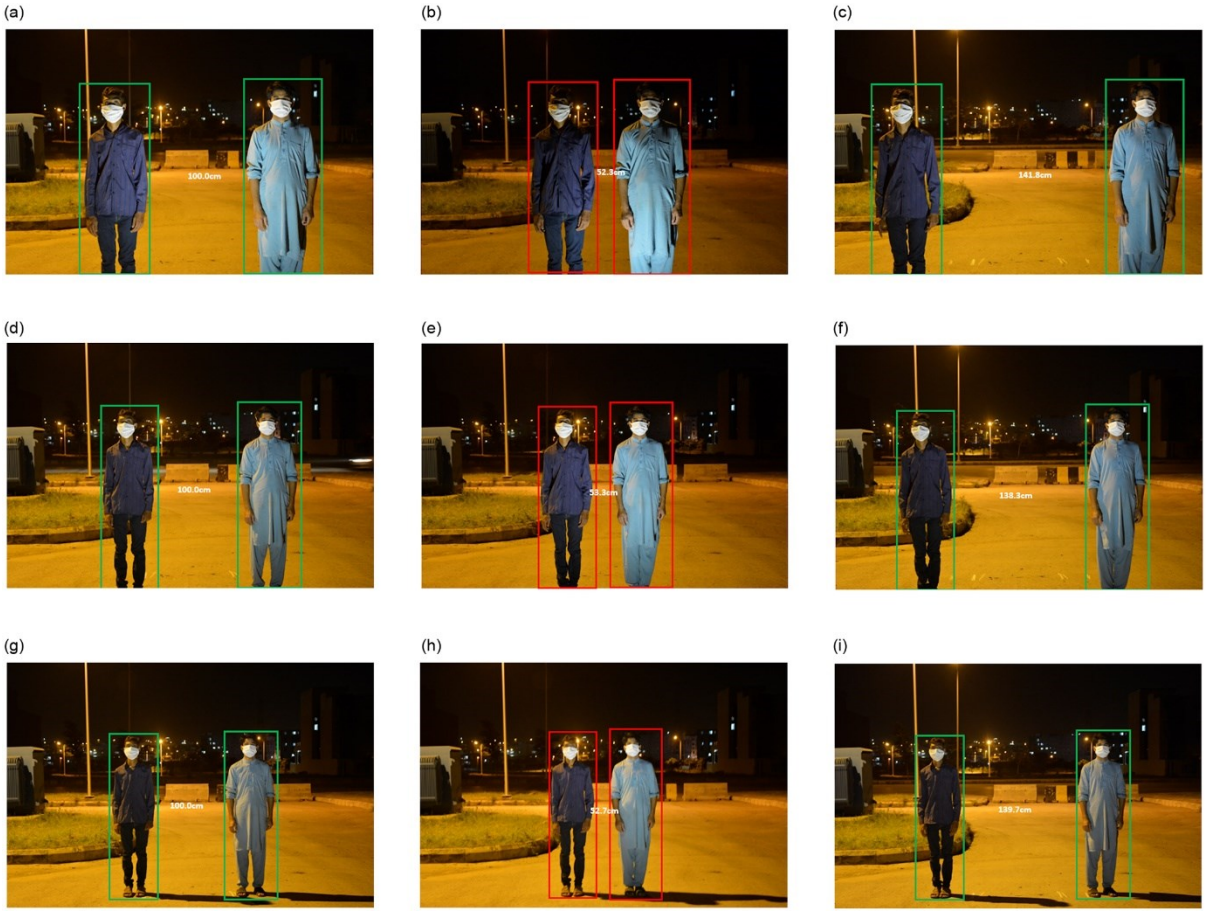


Figure 8 – Test visualizations of our social distance monitoring approach.

With reference to above image

a. **Cd = 400 cm**
b. **Test 1**
c. **Test 2**
d. **Cd = 500 cm**
e. **Test 3**
f. **Test 4**
g. **Cd = 600 cm**
h. **Test 5**
i. **Test 6.**

6.2 – Conclusions

Our custom Social Distancing Analyzer model was extensively tested under a wide range of environments, with differing lighting situations, poses and occlusions, and its mostly a success under real world monitoring conditions. However, we find some circumstances occurred where its success was not up to par like few examples where the qualified model did not work well include cases like when we are mapping the actual

distance in feet into pixel distance via hit and trial method . No Algorithm is Used. Also, our model has depth Error for front view i.e., If a person in frame is behind another person but far from him/her still our model detect it as a violation. We are using yolo which theoretically generates good FPS but still only 10-12 frames are processed per second. For different scenarios (CCTV, Webcam) different threshold is required but we are changing the threshold manually.
Moreover, when bright light was aimed towards the camera lens in the background or very low light settings. This is considered appropriate, though, since the compiled dataset does not contain any images in low light conditions, so we do not expect it to function well under these conditions either.

6.3 – Recommendations for Future work

As previously mentioned in this our report, our social distancing detector lacked proper camera calibration, preventing us from (easily) mapping pixel distances to actual observable units (i.e., meters, feet, etc.). As a result, using proper camera calibration is the first step in developing our social distancing detector. For accurate mapping we can use pixel calibration algorithms in our model. As a result, you'll get better results and be able to calculate real observable units (rather than pixels).

For depth Error we can use depth Estimation camera or we can train our model with deep neural networks. (https://smartlabai.medium.com/image-based-depth-estimation-with-deep-neural-networks-5ae70545ff5f)

We can automatically estimate the threshold for different scenarios with the help of calibration algorithms. We can use YOLO v4 for faster executions for the video stream. Consider using a Single Shot Detector (SSD) on your GPU to speed up the pipeline even further. This will significantly increase frame throughput rate.

APPENDICES AND REFERENCES

i. *W. H. Organization, "WHO corona-viruses (COVID-19)," https://www. who.int/emergencies/diseases/novel-corona-virus-2019, 2020, [Online; accessed May 02, 2020].*

ii. *WHO, "Who director-generals opening remarks at the media briefing on covid-19-11 march 2020." https://www.who.int/dg/speeches/detail/, 2020, [Online; accessed March 12, 2020].*

iii. *L. Hensley, "Social distancing is out, physical distancing is inheres how to do it," Global News–Canada (27 March 2020), 2020.*

iv. *ECDPC, "Considerations relating to social distancing measures in response to COVID-19 second update," https://www.ecdc.europa.eu/ en/publications-data/considerations, 2020, [Online; accessed March 23, 2020].*

v. *M. W. Fong, H. Gao, J. Y. Wong, J. Xiao, E. Y. Shiu, S. Ryu, and B. J. Cowling, "Nonpharmaceutical measures for pandemic influenza in nonhealthcare settingssocial distancing measures," 2020.*

vi. *F. Ahmed, N. Zviedrite, and A. Uzicanin, "Effectiveness of workplace social distancing measures in reducing influenza transmission: a systematic review," BMC public health, vol. 18, no. 1, p. 518, 2018.*

vii. *W. O. Kermack and A. G. McKendrick, "Contributions to the mathematical theory of epidemics–i. 1927." 1991.*

viii. *C. Eksin, K. Paarporn, and J. S. Weitz, "Systematic biases in disease forecasting–the role of behavior change," Epidemics, vol. 27, pp. 96– 105, 2019.*

ix. *M. Zhao and H. Zhao, "Asymptotic behavior of global positive solution to a stochastic sir model incorporating media coverage," Advances in Difference Equations, vol. 2016, no. 1, pp. 1–17, 2016.*

x. *P. Alto, "Landing AI Named an April 2020 Cool Vendor in the Gartner Cool Vendors in AI Core Technologies," https://www.yahoo.com/ lifestyle/landing-ai-named-april-2020-152100532.html, 2020, [Online; accessed April 21, 2020].*

xi. *A. Y. Ng, "Curriculum Vitae," https://ai.stanford.edu/~ang/ curriculum-vitae.pdf.*

xii. *L. AI, "Landing AI Named an April 2020 Cool Vendor in the Gartner Cool Vendors in AI Core Technologies," https://www.prnewswire.com/ news-releases/, 2020, [Online; accessed April 22, 2020].*

xiii. *B. News, "China coronavirus: Lockdown measures rise across Hubei province," https://www.bbc.co.uk/news/world-asia-china51217455, 2020, [Online; accessed January 23, 2020].*

xiv. *N. H. C. of the Peoples Republic of China, "Daily briefing on novel coronavirus cases in China," http://en.nhc.gov.cn/2020-03/20/c 78006. htm, 2020, [Online; accessed March 20, 2020].*

xv. *K. Prem, Y. Liu, T. W. Russell, A. J. Kucharski, R. M. Eggo, N. Davies, S. Flasche, S. Clifford, C. A. Pearson, J. D. Munday et al., "The effect of control strategies to reduce social mixing on outcomes of the covid19 epidemic in wuhan, china: a modelling study," The Lancet Public Health, 2020*

xvi. *S. Brutzer, B. Hoferlin, and G. Heidemann, "Evaluation of background ¨ subtraction techniques for video surveillance," in CVPR 2011. IEEE, 2011, pp. 1937–1944.*

xvii. *S. Aslani and H. Mahdavi-Nasab, "Optical flow based moving object detection and tracking for traffic surveillance," International Journal of Electrical, Computer, Energetic, Electronic and Communication Engineering, vol. 7, no. 9, pp. 1252–1256, 2013.*

xviii. *P. Dollar, V. Rabaud, G. Cottrell, and S. Belongie, "Behavior recognition ´ via sparse spatio-temporal features," in 2005 IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance. IEEE, 2005, pp. 65–72.*

*****