Social Distancing Detection Software

Motivation:

The rampant coronavirus disease 2019 (COVID-19) has brought global crisis with its deadly spread to **more than 180 countries**, and over 150 million **confirmed cases** along with **more than 3 million deaths globally** as on April 30, 2021. The **absence of any active therapeutic agents** and **the lack of immunity against COVID19** increases the vulnerability of the population. Since the availability of vaccines are scarce at the moment, social distancing is a feasible and effective approach to fight against this pandemic and contain it.

Objective:

Motivated by this notion, we aim to **estimate Crowd Sizes and Fight the Virus,** i.e. develop a solution that can **estimate crowd sizes.** By determining the crowd size in a particular place, we can understand if people are social distancing properly or not.

The goal of the project is to:

1. Count the number of people in stores/buildings/streets etc.
2. Identify whether the people are maintaining social distancing limits
3. Optimize the algorithm to perform the above-mentioned functions at real-time.

Existing Solutions:

Broadly speaking, there are currently four methods we can use for counting the number of people in a crowd:

i. Density Estimation-Based Methods:

We first create a **density map** for the objects. Then, the algorithm **learns a linear map** between the extracted features and their object density maps. We can also use **random forest regression** for the model to learn **non-linear mapping**.

ii. Regression-Based Methods:

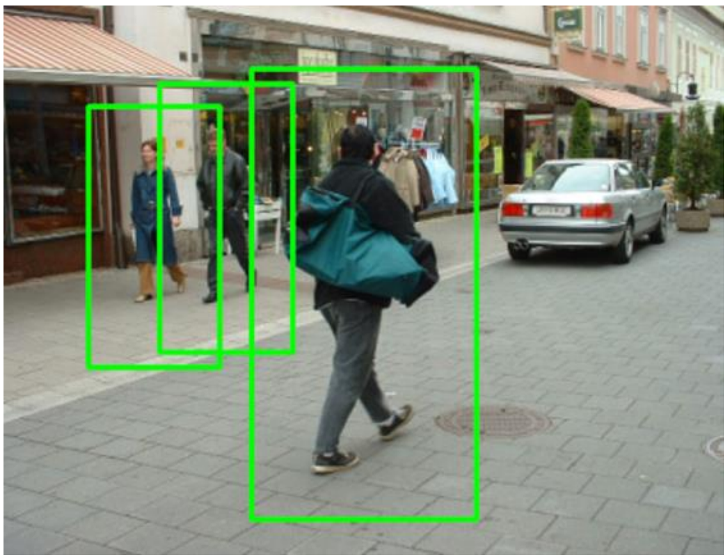
Regression-based methods trumps detection-based methods in extracting low-level features. **We first crop patches from the image and then, for each patch, extract the low-level features.**

iii. <u>CNN (Convolutional Neural Network)-Based methods:</u>

Instead of looking at the patches of an image, we build an **end-to-end regression method** using CNNs. This **takes the entire image as input** and directly generates the crowd count. CNNs work really well with regression or classification tasks, and they have also proved their worth in **generating density maps**.

iv. <u>Detection-based Methods:</u>

Here, we use a **moving window-like detector** to identify people in an image and obtain a count of the people. The methods used for detection **require well trained classifiers that can extract low-level features**. Although these methods work well for detecting faces, **they do not perform well on crowded images as most of the target objects are not clearly visible.**

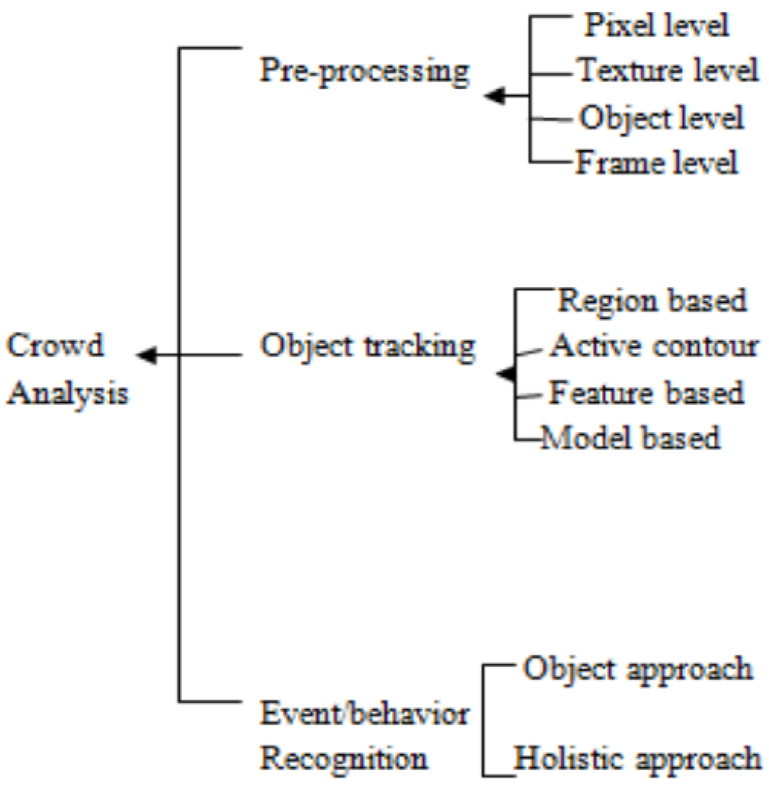


Taxonomy of Crowd Analysis:

a) Pre-Processing:

Pixel-Level Analysis: Pixel-based analysis depends on very local features to estimate the number of people in a crowd scene. Because this method utilizes low-level features, most of the pixel-based methods are focused on crowd density estimation rather than identifying individuals.

Texture Level Analysis: This analysis explores high level features when compared to pixel- based approaches, it is also mostly used to estimate the number of people in scene rather than identifying individuals.



Object Level Analysis: Methods that rely on object level analysis try to identify individuals' objects in scene. They tend to produce more accurate information when compared to pixel level analysis or texture level analysis, but identifying individuals in a single image or a video sequence is mostly feasible in lower density crowds.

Frame Level Analysis: Frame level analysis model behaviors of full scene within the field of view of a camera.

b) Object Tracking:

Attempts to minimize the constraints such as occlusion, color intensity, illumination condition, appearance, etc.

Region Based Approach: It is a robust computer vision approach in unconstrained crowd scenes where information such as density, direction and velocity are extracted using optical flow technique.

Active Contour Based Approach: Active contour based approach is used to model the target partial occlusions and to extent some noise. Weakness of this approach includes no change in the color histogram when similar objects are encountered in a crowd (e.g. Head).

Feature Based Approach: Feature based approach is presented in feature image by describing the blob level feature. Examples include size, shape, elongation, luminance histogram and displacement histogram.

Model Based Approach: Model based approach can solve blob merge and split constraints. This approach is used to segment and track multiple people occlusions. Bottom up image analysis is used to improve efficiency in computer vision.

c) Event / Behavior Recognition:

Another important process in a crowd analysis is event / behavior recognition. It can be characterized by regular motion patterns such as direction, speed, etc.

Object Based Approach: A crowd is analyzed by treating a collection of individuals to estimate the velocities, direction and abnormal motion. The complexity occurs when occlusions exist that may affect the process of analysis such as object detection, tracking trajectories and recognizing the activities in a dense crowd.

Based on this approach, there are two methods, feature correlation and binary function. Feature correlation is used for positioning at the center of the head while binary head is defined to represent the distance between the agents.

Holistic Based Approach: A crowd is analyzed by treating a single entity to estimate the velocities, direction and abnormal motion. The analysis covers medium to high density scenarios in global entities.

Proposed Solution:

The solution proposed by this project is that instead of training the algorithm using the conventional method we use the already available datasets and integrate them with latest object detection software such as YOLOv3 to ensure that the detection happens in real-time.

The YOLO architecture boasts of residual skip connections, and up sampling. **The most salient feature of v3 is that it makes detections at three different scales.** YOLO is a fully convolutional network and its eventual output is generated by applying a 1 × 1 kernel on a feature map. In YOLO v3, **the detection is done by applying 1 × 1 detection kernels on feature maps of three different sizes at three different places in the network.**

Advantages of YOLOv3 are as follows:

- Better at detecting smaller objects
- More Bounding boxes per image
- Capable of multilabel processing on images rather than SoftMax.

Therefore, in this project we will perform:

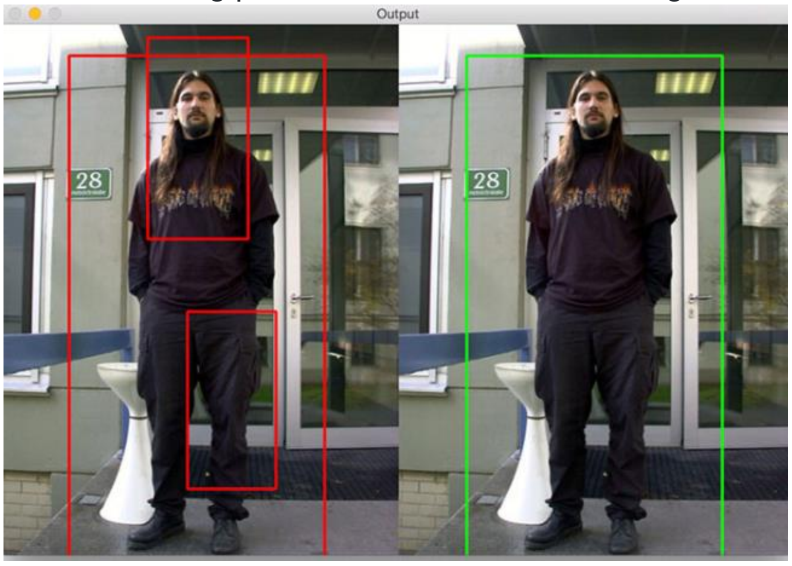
Person Detection:

1. We will be using YOLOv3, trained on COCO dataset for object detection.
2. YOLO treats object detection as a regression problem, taking a given input image and simultaneously learning bounding box coordinates and corresponding class label probabilities.
3. It is used to return the person prediction probability, bounding box coordinates for the detection, and the centroid of the person.

Distance Measurement:

1. Euclidean distance is then computed between all pairs of the returned centroids of boxes which have detected the person.
2. Based on these pairwise distances, we check to see if any two people are less than/close to 'N' pixels apart

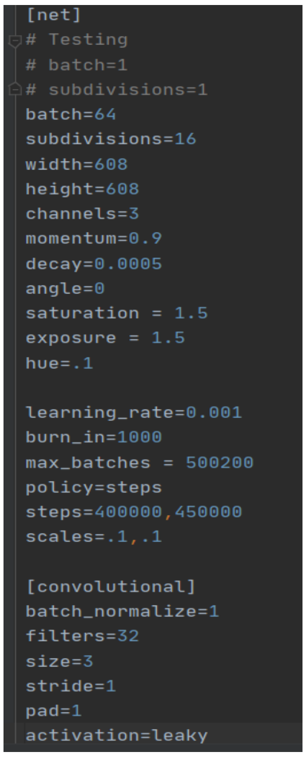
3. We also understand that sometimes people walking by might accidentally come in close proximity to each other hence we try to implement 2 levels of alerts depending on the distance between 2 individuals.
4. We will also be using **Non-maxima Suppression (NMS),** to reduce the overlapping of 2 boxes and combining the boxes to one to represent the true detection. The following picture shows how the NMS algorithm works



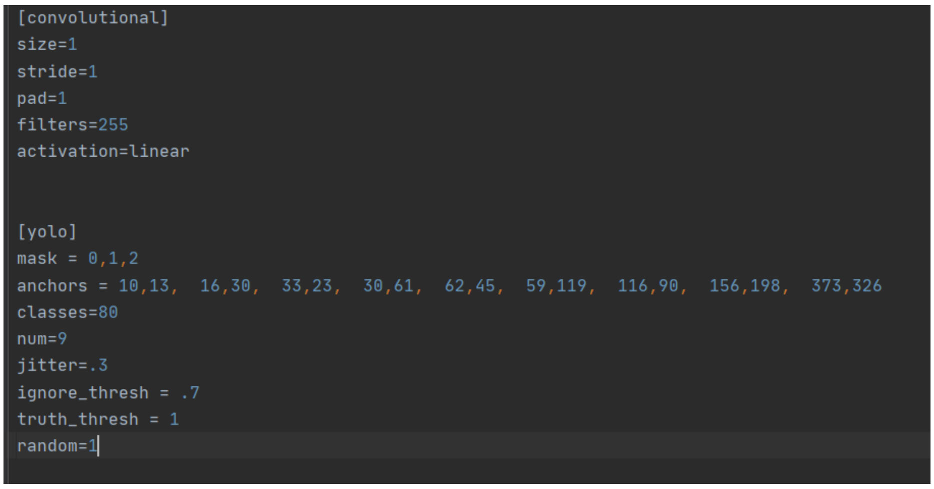
Left: Without NMS, Right: With NMS

Implementation:

1. First in order to implement the YOLOv3 we need to create a configuration file which determines the number of layers and the structure of the network, thereby creating the base of the detection software.



```
[net]
# Testing
# batch=1
# subdivisions=1
batch=64
subdivisions=16
width=608
height=608
channels=3
momentum=0.9
decay=0.0005
angle=0
saturation = 1.5
exposure = 1.5
hue=.1

learning_rate=0.001
burn_in=1000
max_batches = 500200
policy=steps
steps=400000,450000
scales=.1,.1

[convolutional]
batch_normalize=1
filters=32
size=3
stride=1
pad=1
activation=leaky
```



```
[convolutional]
size=1
stride=1
pad=1
filters=255
activation=linear


[yolo]
mask = 0,1,2
anchors = 10,13,  16,30,  33,23,  30,61,  62,45,  59,119,  116,90,  156,198,  373,326
classes=80
num=9
jitter=.3
ignore_thresh = .7
truth_thresh = 1
random=1
```

This is not the entire code as the code contains several more similar convolution layers that allow the detection to be accurate.

[net] section:

- **batch=64** - number of samples (images, letters, ...) which will be processed in one batch
- **subdivisions=16** - number of mini_batches in one batch, size mini_batch = batch/subdivisions, so GPU processes mini_batch samples at once, and the weights will be updated for batch samples (1 iteration processes batch images)
- **width=608 -** network size (width), so every image will be resized to the network size during Training and Detection
- **height=608 -** network size (height), so every image will be resized to the network size during Training and Detection
- **channels=3 -** network size (channels), so every image will be converted to this number of channels during Training and Detection
- **inputs=256 -** network size (inputs) is used for non-image data: letters, prices, any custom data
- **Angle/Saturation/momentum/decay** are part of data augmentation in order to get better results when applied into the real world.

[yolo] section:

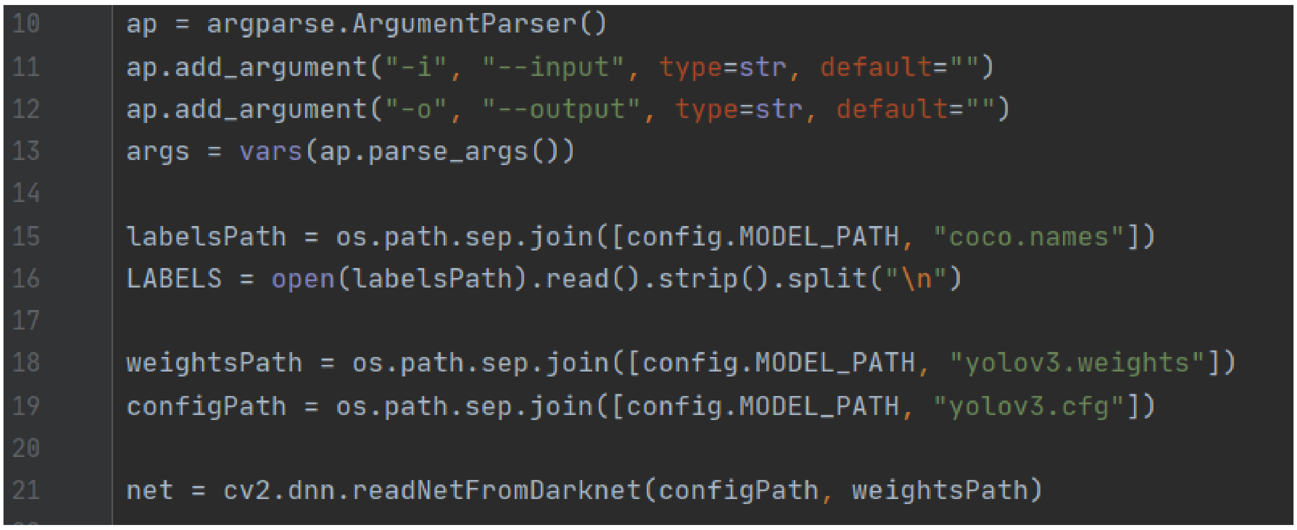
- **mask= 0,1,2** - layer of prediction 0,1,2 anchor boxes
- **anchors= 10,13, 16,30, 33,23….** - the relevant pre-selection boxes
- **classes=80** - Number of objects that the network needs to identify
- **num = 9 -** Each grid cell predicts several boxes, which is consistent with the number of anchors. When you want to use more anchors, you need to increase the num, and if you adjust the num.
- **jitter = 0.3 -** Jitter is the argument of crop, and jitter. Used to generate more data.

After training the model, we store the weight of the convolution network in a separate file as ***yolo.weights*** and store it so that it can be used for further applications. Hence, we can modify the same algorithm in the future to track cars/animals etc., with just minor tweaks as we already possess the classification parameters for every class.

2. Now we create a config file which allows us to choose the input for the video footage, the mail id to which we send the alert notifications. But however, as we were not able to procure a raspberry pi to capture the footage and send to the system in these troubling times, we will be using pre recorded videos in system path.

3. Now in python we implement the final code that combines all the individual tasks such as taking the yolo. weights, importing the video and providing a live analysis of the distancing norms followed by individuals.

We will now segment the code and analyze each of them



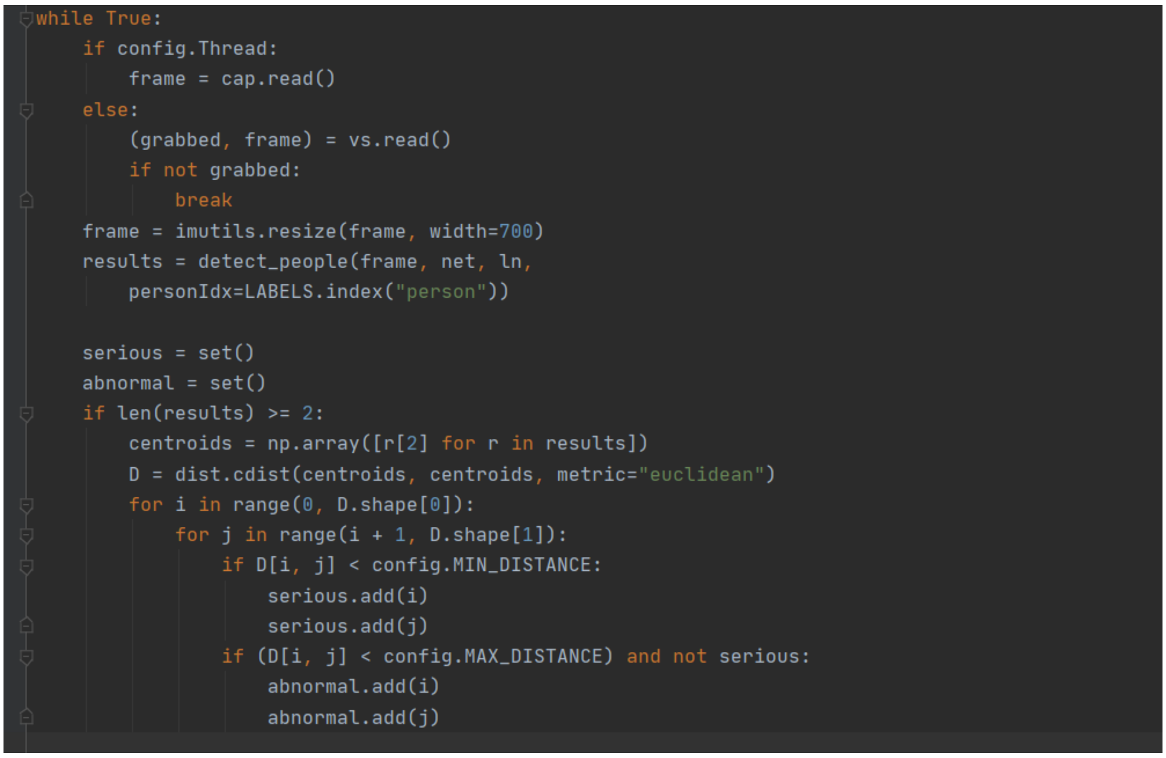
```python
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--input", type=str, default="")
ap.add_argument("-o", "--output", type=str, default="")
args = vars(ap.parse_args())

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

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)
```

- Here we take in the path of the location of the input video and also the location for the output video to be stored.
- We derive the paths for YOLO weights and model configuration.
- The next lines load the COCO labels to which our YOLO models were trained

Small piece of the Coco dataset

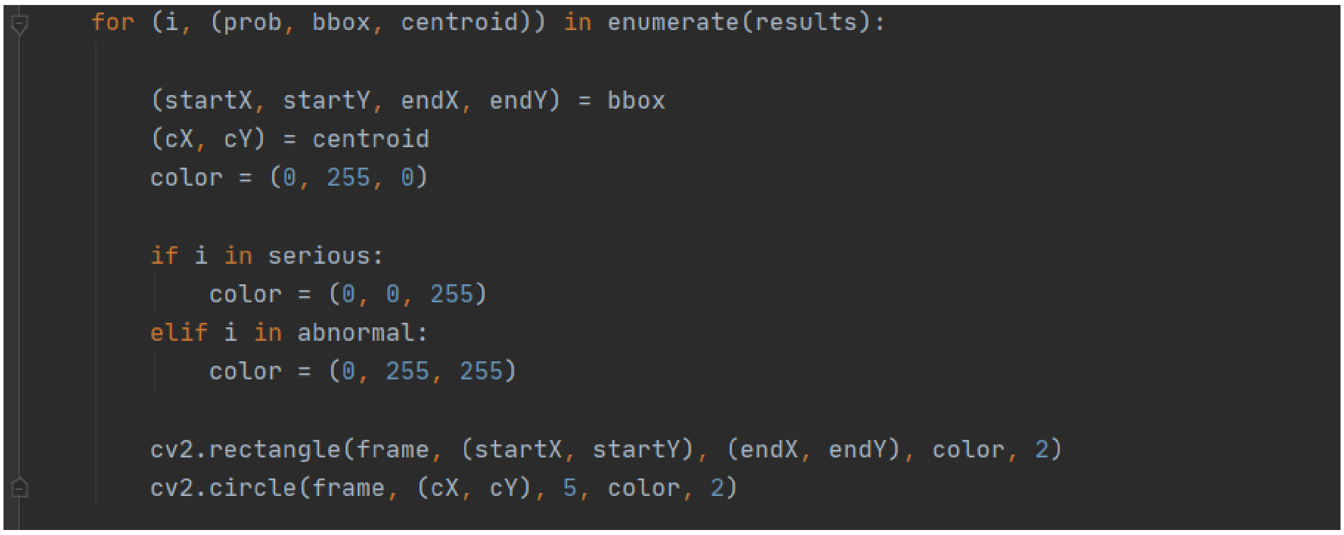
The next step is to go through the video frame by frame and run the detection, to identify if any social distancing violations are made.

Referring to the code snippet in the following page:



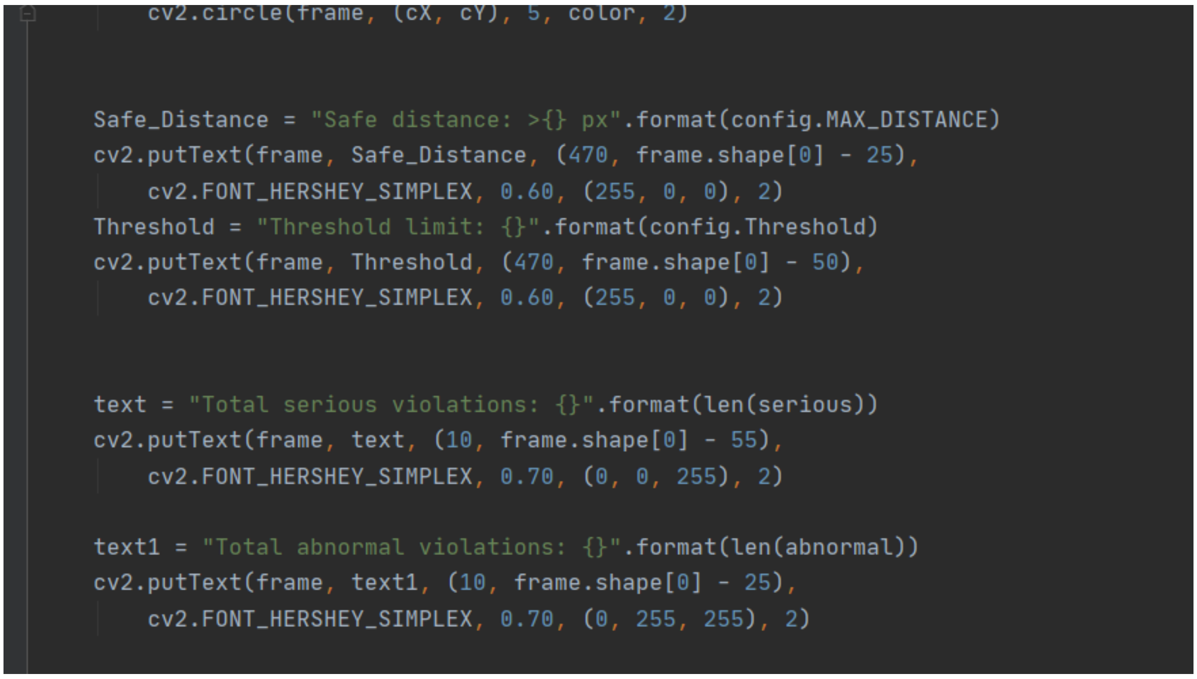
```python
while True:
    if config.Thread:
        frame = cap.read()
    else:
        (grabbed, frame) = vs.read()
        if not grabbed:
            break
    frame = imutils.resize(frame, width=700)
    results = detect_people(frame, net, ln,
        personIdx=LABELS.index("person"))

    serious = set()
    abnormal = set()
    if len(results) >= 2:
        centroids = np.array([r[2] for r in results])
        D = dist.cdist(centroids, centroids, metric="euclidean")
        for i in range(0, D.shape[0]):
            for j in range(i + 1, D.shape[1]):
                if D[i, j] < config.MIN_DISTANCE:
                    serious.add(i)
                    serious.add(j)
                if (D[i, j] < config.MAX_DISTANCE) and not serious:
                    abnormal.add(i)
                    abnormal.add(j)
```

- ➢ Here we loop over the frames of the video, resize the video in order to make sure the scale is right in all cases as we are dealing with in pixels.
- ➢ Then we extract the distances between the centroid of two boxes and calculate the Euclidian distances between the 2 centroids.
- ➢ So, we add the abnormal set and the serious distance indices to their respective sets.



```python
for (i, (prob, bbox, centroid)) in enumerate(results):

    (startX, startY, endX, endY) = bbox
    (cX, cY) = centroid
    color = (0, 255, 0)

    if i in serious:
        color = (0, 0, 255)
    elif i in abnormal:
        color = (0, 255, 255)

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

- This piece of code loops over the frames again and identifies the bounding boxes that need to be highlighted. And overlays them with color appropriate to the violation level.
- We can also add additional information into the overlay of the video that shows useful data such as the total number of people in the footage, number who are committing violations and how serious along with highlighted boxes.

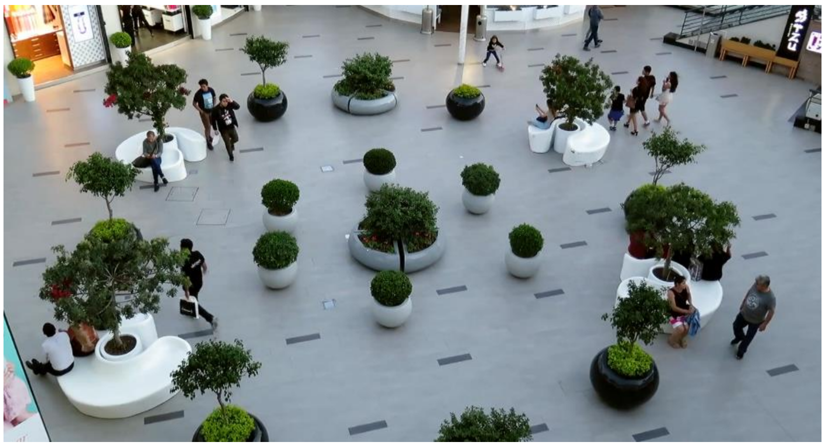


```
        cv2.circle(frame, (cX, cY), 5, color, 2)


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


    text = "Total serious violations: {}".format(len(serious))
    cv2.putText(frame, text, (10, frame.shape[0] - 55),
        cv2.FONT_HERSHEY_SIMPLEX, 0.70, (0, 0, 255), 2)

    text1 = "Total abnormal violations: {}".format(len(abnormal))
    cv2.putText(frame, text1, (10, frame.shape[0] - 25),
        cv2.FONT_HERSHEY_SIMPLEX, 0.70, (0, 255, 255), 2)
```

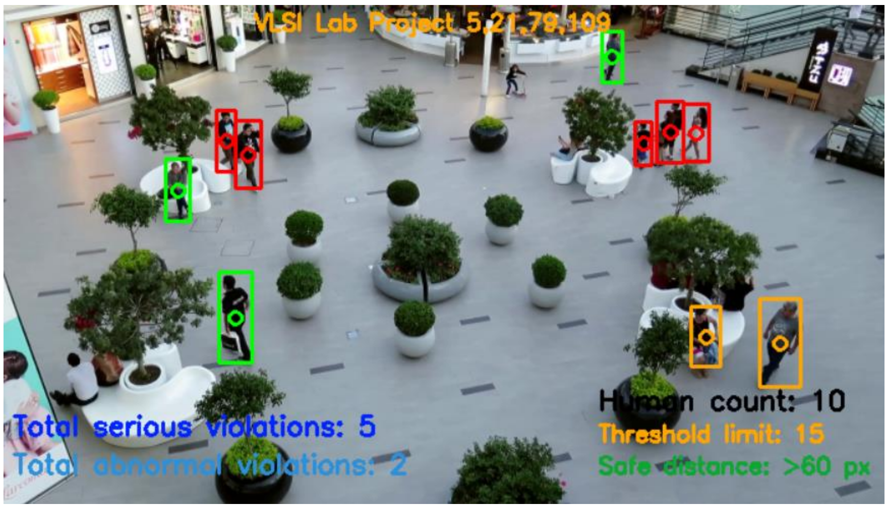
Result and Discussion:

Now we test our model on some CCTV footages to make sure the detection system works appropriately and produces the desired results of the project.

Here are some snapshots of some sample footage, the images show before and after the

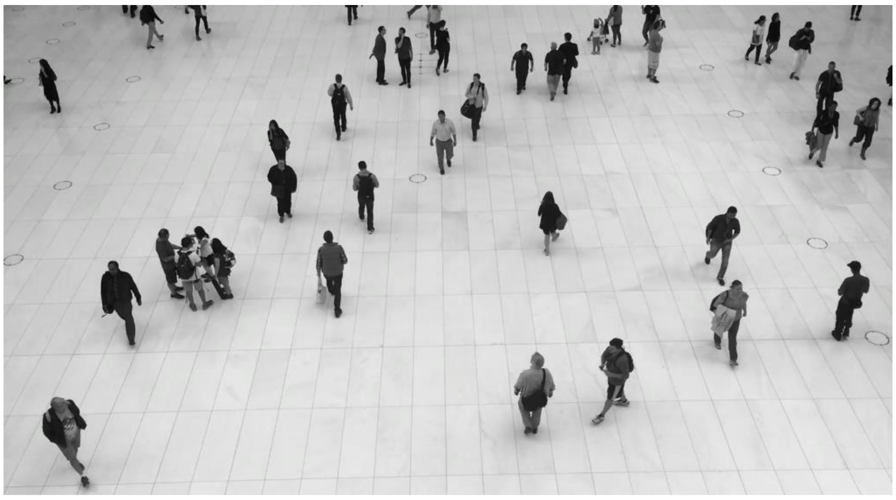


Original Footage

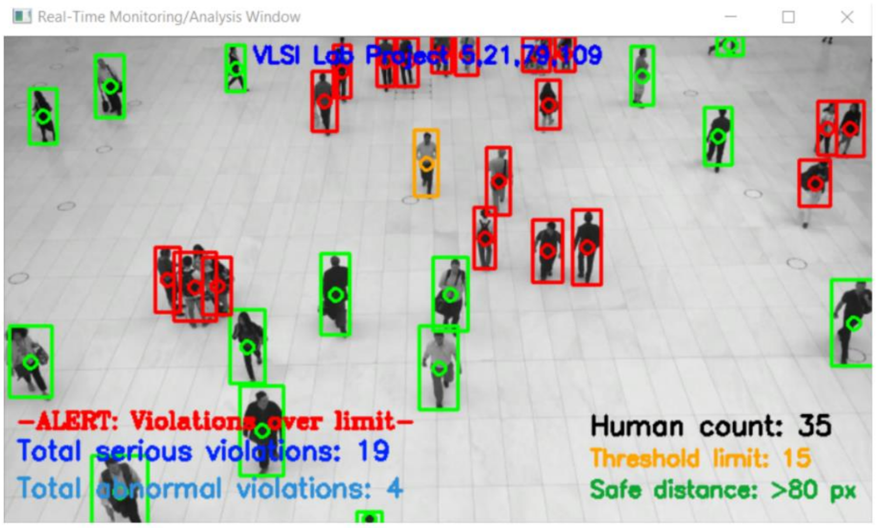

VLSI Lab Project 5,21,79,109
Human count: 10
Threshold limit: 15
Safe distance: >60 px
Total serious violations: 5
Total abnormal violations: 2

Footage after the model was run

Video 2:



Original Footage


Real-Time Monitoring/Analysis Window
VLSI Lab Project 5,21,79,109
-ALERT: Violations over limit-
Total serious violations: 19
Total abnormal violations: 4
Human count: 35
Threshold limit: 15
Safe distance: >80 px

Footage after the model was run

We have also had to change the safe distance threshold value for each video as the angle of footage taken for each is different and hence the number of pixels between 2 centroids of boxes that have detected people would be different.

Hence we can infer that the model is successful and is able to run and analyze a video at real time. We were also able to identify and differentiate different levels of social distancing violations at real-time.

Conclusion

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.

Social distancing is a non-pharmaceutical infection prevention and control intervention implemented to avoid/decrease contact between those who are infected with a disease causing pathogen and those who are not, so as to stop or slow down the rate and extent of disease transmission in a community. This eventually leads to decrease in spread, morbidity and mortality due to the disease.

In a country as densely populated as India, automating the detection of crowding can help in the enforcement of social distancing. The existing CCTV infrastructure can be leveraged to analyze real-time video and automatic alerts can be sent to the police to prevent illegal congregations.

References

https://www.pyimagesearch.com/2020/02/03/how-to-use-opencvs-dnn-module-with-nvidia-gpus-cuda-and-cudnn/

https://www.pyimagesearch.com/2018/11/12/yolo-object-detection-with-opencv/

https://www.pyimagesearch.com/2015/11/09/pedestrian-detection-opencv/

https://www.pyimagesearch.com/2020/06/01/opencv-social-distancing-detector/

https://jamesbowley.co.uk/accelerate-opencv-4-2-0-build-with-cuda-and-python-bindings/

https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b

https://www.mdpi.com/1424-8220/20/1/43/htm

https://www.activestate.com/blog/how-to-monitor-social-distancing-using-python-and-object-detection/

https://www.analyticsvidhya.com/blog/2019/02/building-crowd-counting-model-python/

https://www.hindawi.com/journals/mpe/2020/6692257/

https://www.lume.ufrgs.br/bitstream/handle/10183/27633/000764413.pdf?sequence=1&origin=publication_detail

https://link.springer.com/article/10.1007/s11135-021-01117-7

https://www.technoarete.org/common_abstract/pdf/IJERCSE/v5/i4/Ext_42371.pdf

https://arxiv.org/pdf/2005.01385.pdf

https://ieeexplore.ieee.org/document/5555536

http://vision.cse.psu.edu/publications/pdfs/GeCollinsRubackPAMI2011.pdf

https://www.irjet.net/archives/V7/i8/IRJET-V7I8698.pdf