SOCIAL DISTANCING MONITOR

Ananya Bhadauria

Introduction

- Since Covid-19 is an airborne disease, the need for following social distancing has never been more crucial than it is right now since it is a behaviour that has an impact on everyone, whether people observe it or not, thanks to a cascading effects brought on by one single person.
- We would be able to emphasize in a feed how many people are not keeping a safe distance, thanks to this computer vision-based monitoring technology.
- It emphasizes that authorities take action in addition to reporting the general situation, such as the proportion of people who are abiding by the rules or breaking them.

Motivation

- To keep in check the number of new cases originating on a day-to-day basis due to neglection of guidelines.
- The detection tool was created to identify and warn individuals to keep a keeping a safe distance from one another while assessing a video feed from CCTV cameras
- To alert the government about potential hotspots/ places where guidelines are not followed by the majority of civilians to take necessary actions.

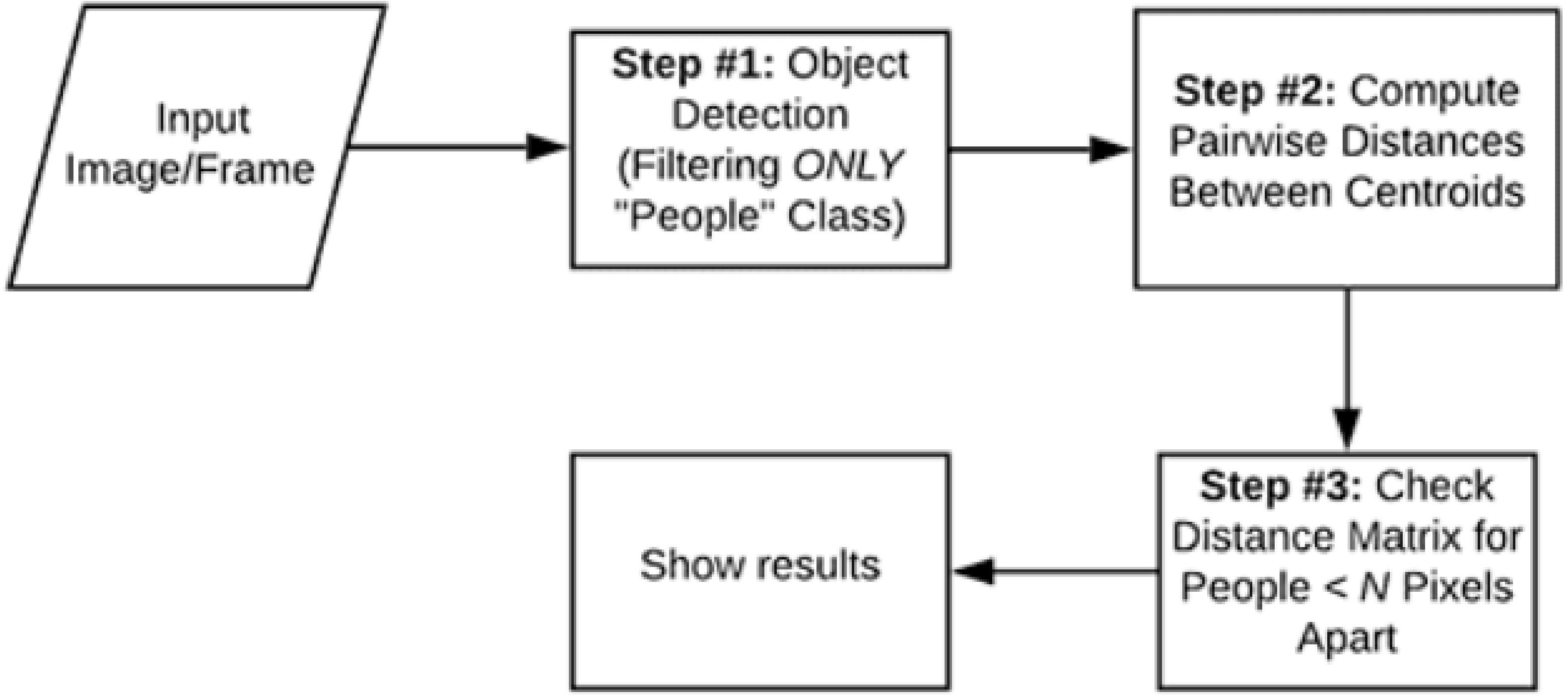
Objective

- Manual operators are currently deployed to oversee guideline violations. But this has limitations like operator fatigue, human errors and will require a huge workforce for large scale monitoring as is the case with social distancing monitoring systems.
- To lay out a version for an unsupervised type of monitoring system to identify social distancing guideline breakers effectively using machine learning based models working across the CCTVs footage across the world.
- This system must be fast and work accurately in real-time scenarios.

Workflow

- Step 1: Data collection for Machine learning model (COCO Dataset).
- Step 2: Applying ML algorithm (Yolov3) and storing data.
- Step 3: Computing pairwise distance between centroids of bounding boxes and their relative depths.
- Step 4: Checking if euclidean distance b/w two centroids is greater than the recommended distance(120px in our case).
- Step 5: Testing the model on test data
- Step 6: Test and document the final system.

Flow Diagram


Input Image/Frame
Step #1: Object Detection (Filtering *ONLY* "People" Class)
Step #2: Compute Pairwise Distances Between Centroids
Step #3: Check Distance Matrix for People < *N* Pixels Apart
Show results

Why YOLO not CNN

- Yolo algorithm is the most modern computer vision algorithm . It is based on Regression , instead of selecting the interesting part of the image it produces classes and boundary boxes for whole image in one run of the algorithm.
- It is 100x faster than the CNN model and is appropriate for real time monitoring.
- CNN produces more false alerts than YOLO, thus has low accuracy at higher FPS.
- Another key difference is that YOLO sees the complete image at once as opposed to looking at only a generated region in CNN. This contextual information helps in avoiding false positives

Algorithm Used

- Define the minimum safe distance (in pixels) that two people can be from each other.
- Derive the paths to the YOLO weights and model configuration.
- load the YOLO object detector trained on COCO dataset (80 classes)
- Determine only the layer names that we need from YOLO.
- Grab the dimensions of the frame and initialize the list of results.
- Construct a blob from the input frame and then perform a forward.
- Pass the YOLO trained weights transfer learning object detector, giving us our bounding boxes and associated probabilities.

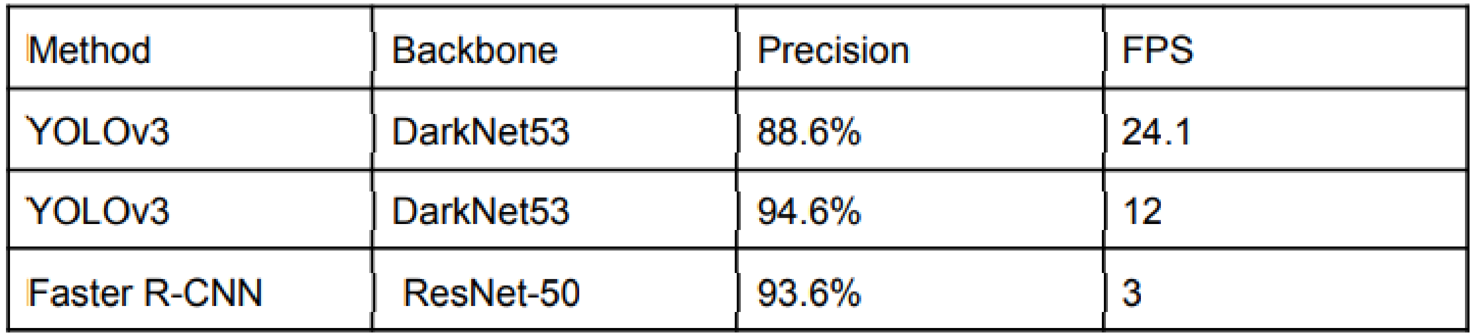
- loop over each of the detections and extract the class ID and confidence (i.e., probability) of the current object detection.
- initialize our lists of detected bounding boxes, centroids, and confidences, respectively.
- Use the center (x, y)-coordinates to derive the top and left corner of the bounding box.
- Apply non-maxima suppression to suppress weak, overlapping bounding boxes
- loop over the indexes we are keeping and extract the bounding box coordinates.

- Update our results list to consist of the person prediction probability, bounding box coordinates, and the centroid.
- Read the next frame from the file and if the frame was not grabbed, then we have reached the end of the stream.
- Determine the depth of bounding boxes using information on heights in pixel and focal lengths.
- Resize(448*448) the frame and then detect people (and only people) in and initialize the set of indexes that violate the minimum social distance using Euclidean distance and mark them with red bounding boxes.

- loop over the results and extract the bounding box and centroid coordinates, then initialize the colour of the annotation
- If the index pair exists within the violation set, then update the colour draw a bounding box around the person and the centroid coordinates of the person.
- Draw the total number of social distancing violations on the output frame.
- Show the output frame.

Conclusion

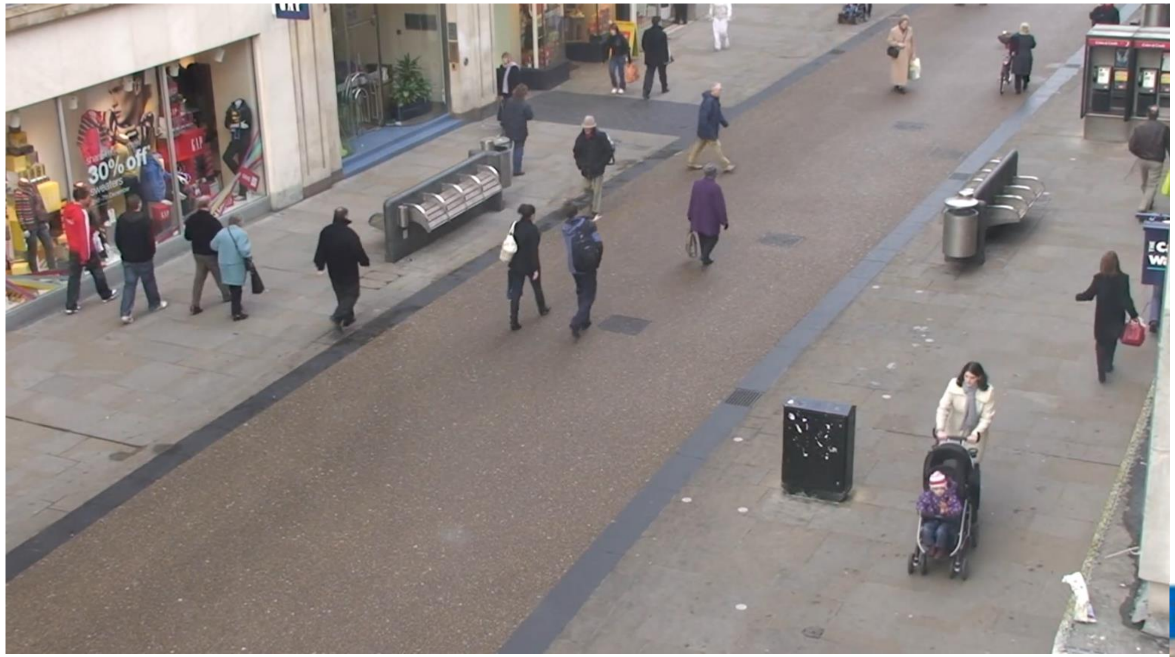
- Our model successfully detected the social distancing violations in a live camera feed accurately.
- Our model reported an accuracy of 94.6% at 12FPS and 88.6% at 24.1FPS which are good enough to perform real time monitoring.

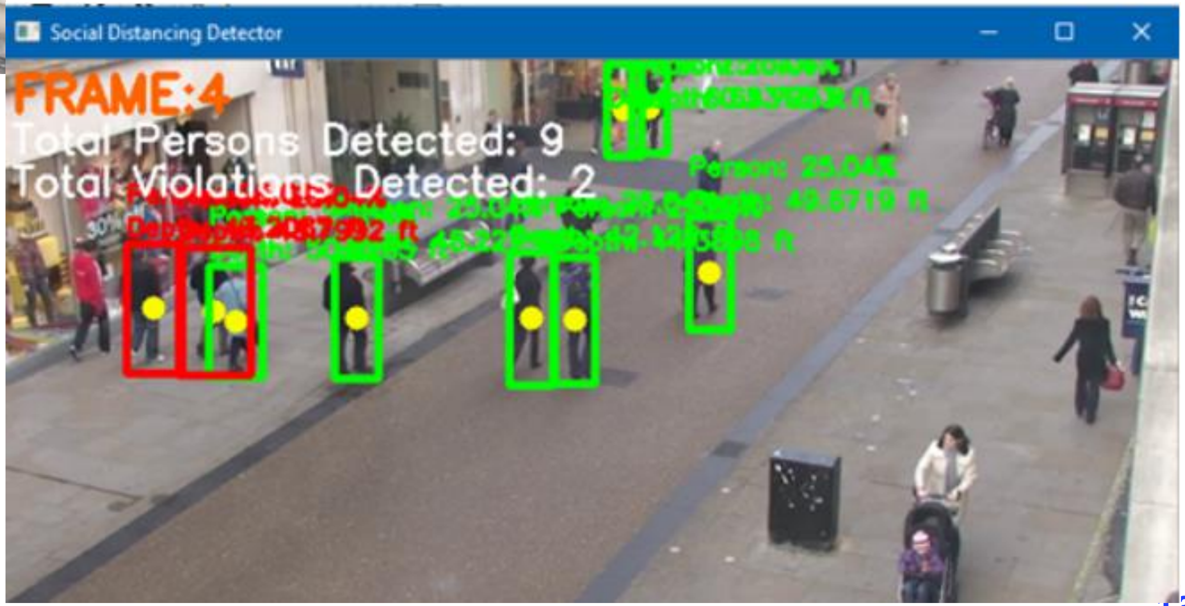


Method	Backbone	Precision	FPS
YOLOv3	DarkNet53	88.6%	24.1
YOLOv3	DarkNet53	94.6%	12
Faster R-CNN	ResNet-50	93.6%	3

Figure 6: YOLO vs Faster R-CNN vs FPS

Output Analysis





FRAME:4
Total Persons Detected: 9
Total Violations Detected: 2

Depth Analysis

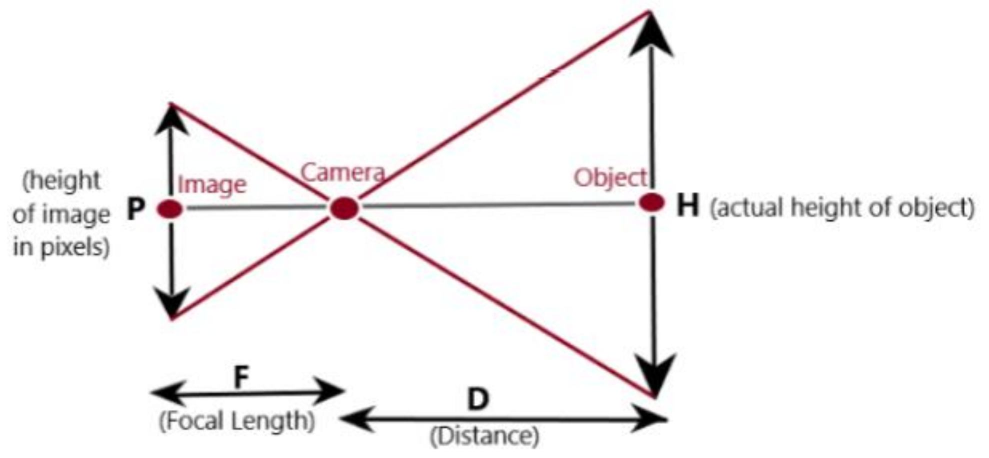


Figure 2: Depth using camera

Based on the similar triangle approach

$$\frac{F}{D} = \frac{P}{H}$$

For finding the focal length (F)

$$F = \frac{P.D}{H}$$

For finding the depth of object from camera (D)

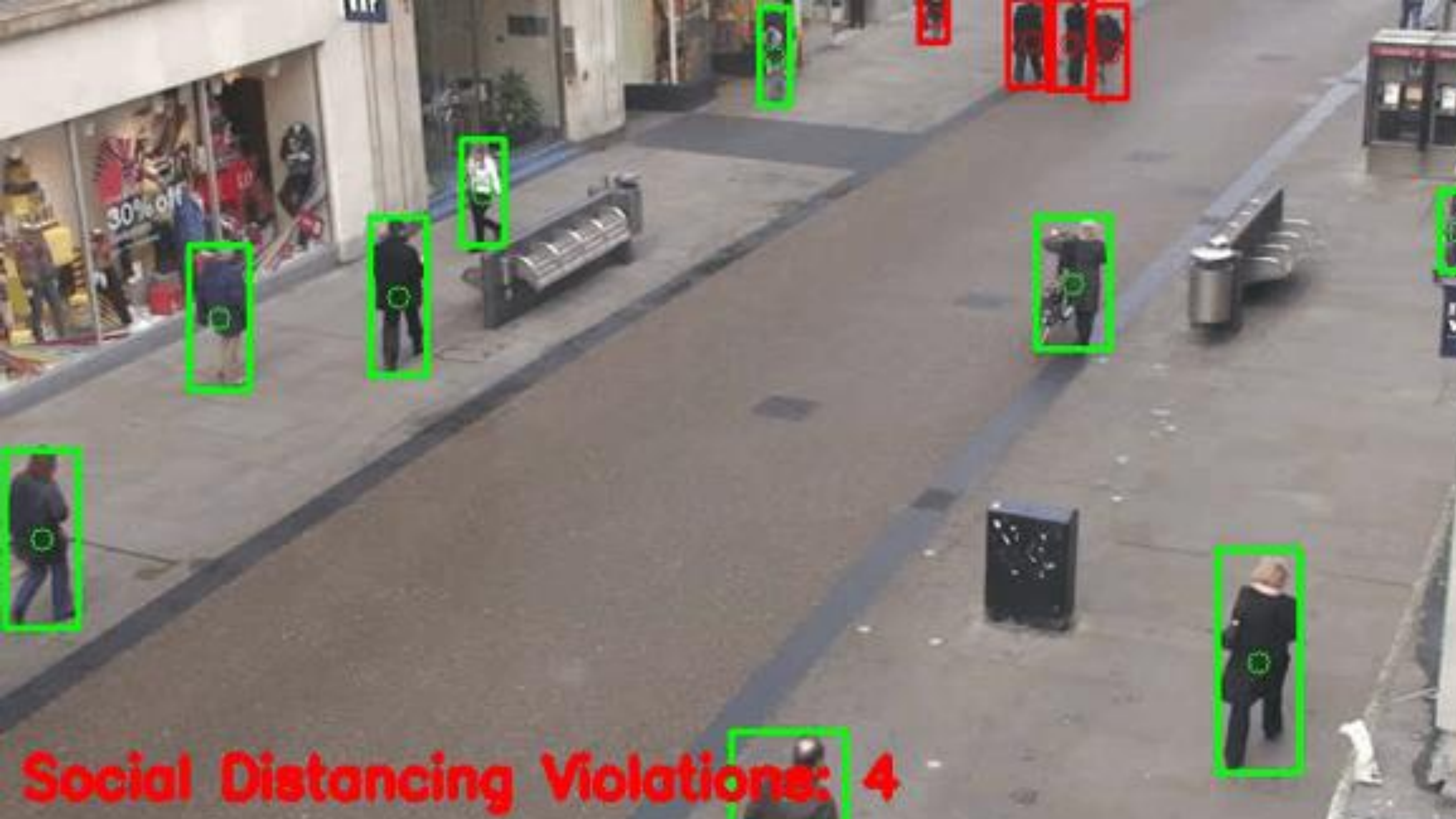
$$D = \frac{F.H}{P}$$

For finding the corresponding actual height (H)

$$H = \frac{P.D}{F}$$

Future Scope

- The ideal approach was to include depth sensors like LiDARs and radars to be fused with the vision element in order to obtain the precise depth from camera to person. The simplest and most accurate method of determining camera depths is with these sensors.
- As opposed to recognising a different individual in every frame, the Deep SORT algorithm for detection and tracking might offer the discovered person a unique ID that would stay with that specific person in all the frames, making it easier to spot on video. Particularly for tracking applications, this will be quite helpful. We can combine this system with face recognition to identify the persons violating the guidelines and impose a fine on them.
- Since, we are using the COCO dataset, which contains 80 classes this can be used for detecting pedestrians and vehicles in self-driving cars. The car will be alerted if another car/pedestrian is found in proximity of it and then be instructed to change the path from path of Collison.


30% off
Social Distancing Violations: 4

THANK YOU

Ananya Bhadauria