CSC 420 Final Project Report
Team EyeDK
Social Distancing Detection

Team Member: Evan Pan, Hanyan Jiang, Victor Zhang

Abstract

In this report, we explored how we can build a social distance detection application and compared the various ways on how this application can be improved. We wanted to determine which state of the art models are more robust and will allow us to perform fast real-time object detection in the context of social distancing analysis. Additionally, we attempted to improve the accuracy of distance measurement by determining factors such as: if our camera can be calibrated to remove distortion, if our frames can be warped into a bird eye perspective, and if we can improve our model by fine tuning parameters. This will enable us to measure using real world coordinates, and hence giving much more accurate distance calculations.

Among all the computer vision methods for object detection, we screen over all the traditional models and deep learning models, then pin down to two methods: YOLOv5 and Detectron2 due to their distinguished performance on the state-of-art object detection tasks. Experiments are done to select the best model between them in the form of a benchmark test, and we focus on their behaviours in both accuracy and inference speed on three different datasets: JAAD, EPFL and our custom data. The YOLOv5 shows significantly better performance in GPU training than Detectron2, which manages to maintain almost the same accuracy in prediction. Therefore, we select YOLOv5 as our baseline model for this program and develop based on its pretrained architecture.

Although our detector was not perfect, we were still able to get qualitatively accurate results in our visualizer and listed out potential optimizations that can further improve our results. Given that social distancing has become an important part of staying healthy in our daily lives, our application can serve as a starting point in building technologies that will help us be more aware of our surroundings.

Table of contents

Introduction

Covid-19 [1] has changed various aspects of our modern lives. In particular, physical interaction between people has changed significantly. Currently the most effective way to limit Covid-19 cases has shown to be avoiding interaction with other people through staying home [2]. However, for people who need to go out (work, food, groceries), social distancing becomes the most effective way of disease prevention.

Social distancing is described as a method by enforcing physical distance between one another and hence reducing the spread of contagious disease. Naturally, to track how social distancing is enforced, detection technologies are developed. Some common methods include the uses of wearable devices[3], voluntarily downloaded apps [4], wi-fi usage analysis [5], and image processing technologies. For our project, we would specifically look for image-based solutions.

There are various advantages of using an image-based solution. First of all, it requires no additional hardware beside a camera, which are wildly available as cell phone cameras and security cameras are ubiquitous in the modern world. Secondly, additional information beside location tracking is available in images. Examples of this are personal features, which can be used to identify a person, and facial garment which can be used to identify whether they are wearing a mask or not.

However, there are also challenges in using an image-based approach. First of all, the objects of interest (people) would need to be identified and localized from the image, this is not a trivial task as feature engineering often fail to generalized this kind of tasks, therefore a learning-based approach would have to be used for such a task, such as R-CNN, Yolo , and detectron2 [6]. A second challenge is to measure distance between individuals detected from the image, this is difficult as the image could be projective, distorted, and there isn't any reference to link the pixel count into a distance metric. In this report, we discuss the challenges of the various approaches and explain the techniques we felt were most appropriate to address the issue of social distance detection.

Problem Statement

The specific task of social distancing monitoring can be further broken down into two sub-problems. The first sub-problem being object detection, and the second sub-problem is to measure the distance between objects that are detected (people).

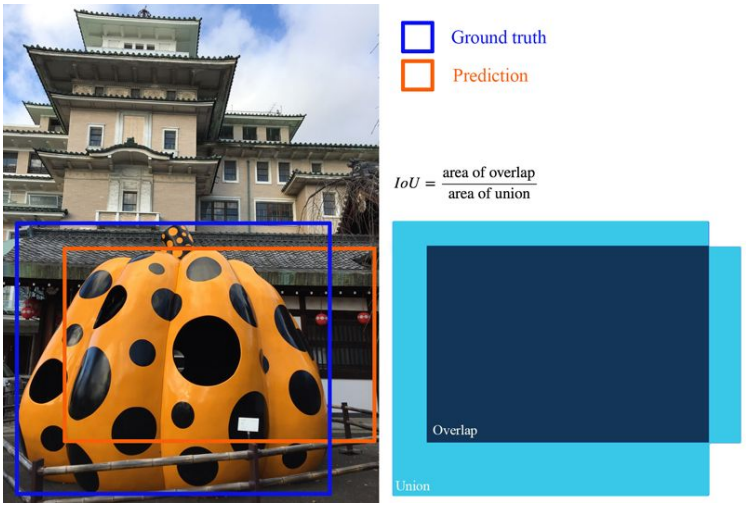
Metrics of Interest - Object detection:

In the case of our application, we not only need to detect if an object is in the image, but we also need to learn the geometrical location of the objects in the image, as we are interested in finding distance between objects. This can be classified as a bounding box problem, which can be more generally described as a segmentation problem. The major challenge for this subproblem would be finding an approach that deals with occlusion, as we need to be able to identify people especially if they are partially blocked by other people. And another potential challenge is being able to detect the object from multiple scales.

The key objectives that are important for this task are processing speed and detection performance. Processing speed is important for our task as a real time algorithm would allow real time intervention, therefore allowing the user to enforce social distancing. The processing speed would be measured in frames per seconds. The performance refers to how accurate the objects are being detected and segmented. The performance would be measured using the mAP(mean average precision), which is calculated with the ratio of the number of true positive and total number of predictions. [13]

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

In object detection, a prediction is considered a true positive when the IOU (Intersection over Union) surpasses a certain threshold.



Since most state of the art object detection algorithms are dependent on very large neural network backbone (i.e. Resnet) that typically require multiple GPUs to train, it is infeasible

for our team to train such models. Therefore, for this subproblem, we will focus our effort on selecting the best model through literature review, and implementing our algorithm using a pretrained model.

Metrics of Interest -Distance Measuring:
To recover 3-D distance from a 2-D image, various approaches can be used, ranging from simply counting pixel distance, computing camera models from external parameters, homography, and other methods. For our project, we wish to create a generalizable method that does not depend on any external parameters.

Since there are no existing benchmarks for this application, we will compare the results of various approaches qualitatively on the same video. Similar to the subproblem of object detection, We will also consider processing time as a key metric for our solution.

Literature Review

For this section, we will first examine existing approaches for object detection, compare the state of the art models, then pick the model that best suits our objectives. Then we will examine some current implementations of social distancing measuring approaches to identify their strengths and weaknesses. Lastly, the distance measuring metrics for these approaches will be compared.

Object detection

Though nowadays the idea of computer vision is closely tied with Deep learning and neural nets, object detection is a topic of interest that predates Alexnet [8] by a long time. Traditionally, object detection algorithms like the Viola–Jones framework [9] or sift [10] generally involves two steps, the first of which is to obtain feature vectors from an image, and the second of which is to match the feature vectors from the image with a feature vector from a known library of objects [11]. The traditional methods work well and efficiently, and are highly explainable. However, compared to modern deep learning methods trained on huge datasets, the traditional methods are less accurate and versatile as shown by a comparative study

Table 2. Average AUC and mAP accuracy for face and head detection on the FDDB dataset and Casablanca dataset.

Detector	FDDB data		Casablanca data	
	AUC	mAP (%)	AUC	mAP (%)
VJ [18]\VJ-CRF [17]	0.6654	67.16	0.42	37.68
HeadHunter [20]\DPM [8]	0.7780	77.78	0.58	51.57
SSD [6]	0.7740	85.00	0.68	57.57
Faster R-CNN [5]	0.9253	91.73	0.55	56.37
R-FCN 50 [3]	0.9339	92.92	0.63	60.53
R-FCN 101 [3]	0.9287	92.53	0.63	60.67
PVANET [13]	0.9168	91.87	0.55	60.11
Local-RCNN [19]	N/A	N/A	0.78	71.76

[12]

As shown by the table, the state of the art Viola Jones algorithm performed poorly as compared to the trained model. However, as the table below from the same study has shown, this method has significantly faster processing speed, which makes traditional models like this more useful in certain applications.

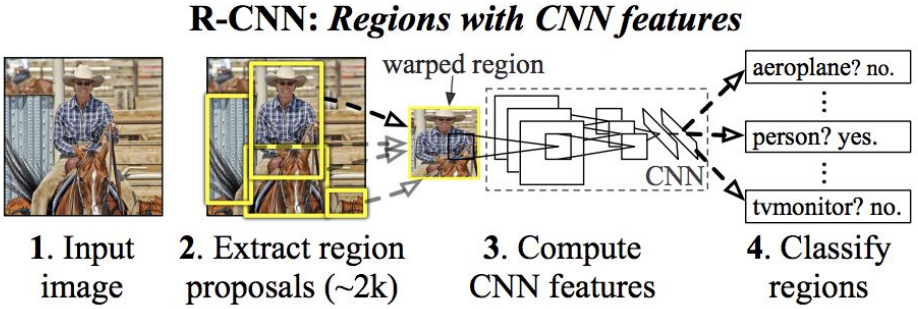
Table 3. Average time and memory complexity for face detection on FDDB.

Detector	Time		Memory consumption (GB)
	GFLOPS	FPS	
Viola-Jones [18]	0.6	60.0	0.1
HeadHunter DPM [20]	5.0	1	2.0
SSD[6]	45.8	13.3	0.7
Faster R-CNN [5]	223.9	5.8	2.1
R-FCN 50 [3]	132.1	6.0	2.4
R-FCN 101 [3]	186.6	4.7	3.1
PVANET [13]	40.1	9.0	2.6
Local RCNN [19]	1206.8	0.5	2.1
Yolo 9000 [16]	34.90	19.2	2.1

[12]

The current state of the art object detection algorithms can be divided into two distinct classes, which are two-step detectrons and one-step detectron. They differ by the number passes the image would go through a neural network. Two-step detectrons are represented by approaches such as the R-CNN (regional based convolutional neural network) class of algorithms, and one-step detectrons are represented by YOLO (you only look once) and SSD (single shot detector) [14].

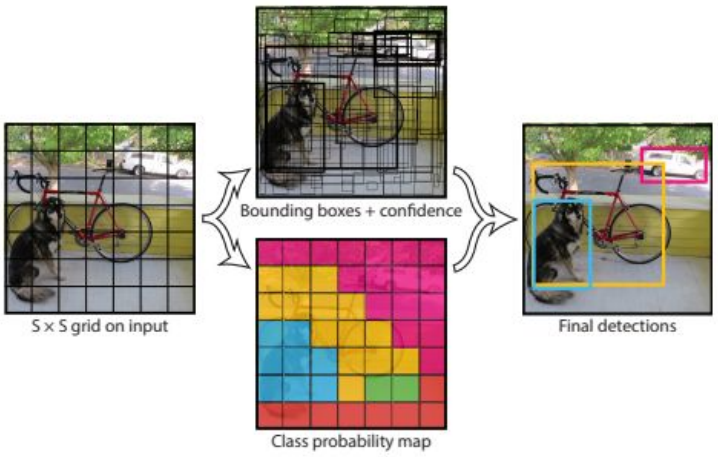
Two-step methods typically involve generating unlabeled boundboxes first (called regional proposals), then using a learned model such as a support vector machine or convolutional neural network to classify each region[15]. Due to the multi-step approach, R-CNNs are typically slower. In the original model the boundboxes (called regional proposals) are generated with a traditional computer vision algorithm called selective search [selective_search], then the regional proposals are classified with a mixture of convolutional neural networks and support vector machines [16].



[15]

The initial approach is improved over time. In the most recent iteration, Faster R-CNN, the regional proposals steps are replaced with a Resnet-Based Convolutional neural network [17], which significantly improves the speed for generating the proposals, and brings the speed of the overall program to close to real time [18].

The one step models on the other hand uses a different approach, which only one pass through a neural network is needed For YOLO, the image is first divided into a S by S grid, then a convolutional neural network is used to generate boundboxes and class prediction for each cell in the grid. The result is then integrated by the network to form the final prediction. [19]



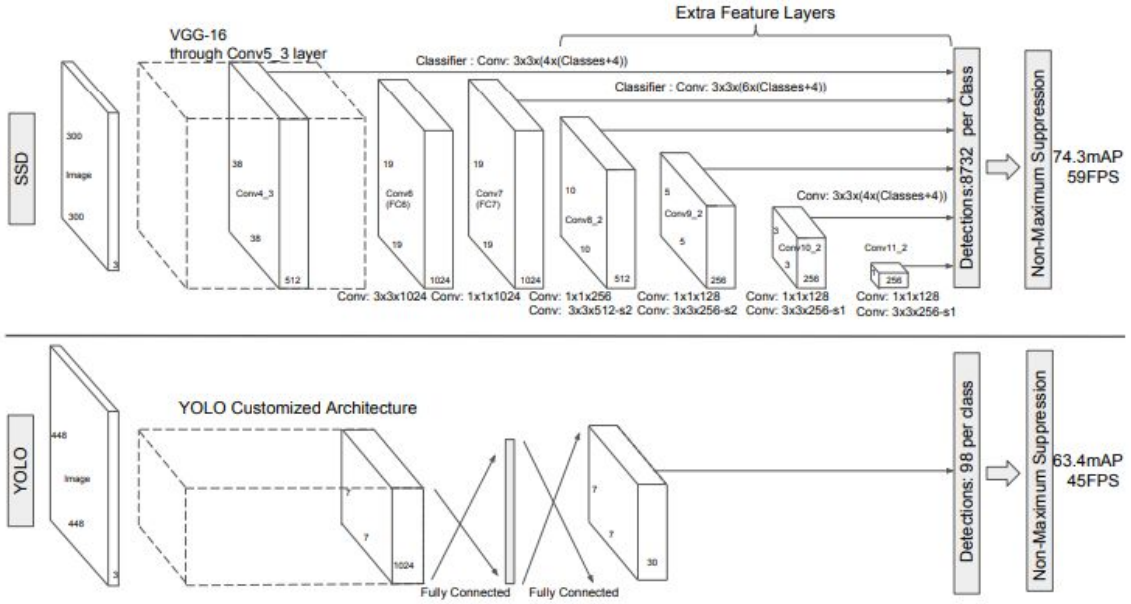
[19]

Unlike the R-CNN approach, since there is no need to run a neural network model with various proposals, the YOLO model is significantly faster. The current iteration of it is the YOLO v5 model released on July 24th [20], though it isn't developed and trained by the author of the original paper, it has a similar architecture as the original models and it has slightly improved precision and run time compared to the previous generations, as shown by a comparison by the author with the coco dataset [7].

Model	AP^{val}	AP^{test}	AP_{50}	$Speed_{GPU}$	FPS_{GPU}		params	FLOPS
YOLOv5s	36.1	36.1	55.3	**2.1ms**	**476**		7.5M	13.2B
YOLOv5m	43.5	43.5	62.5	3.0ms	333		21.8M	39.4B
YOLOv5l	47.0	47.1	65.6	3.9ms	256		47.8M	88.1B
YOLOv5x	**49.0**	**49.0**	**67.4**	6.1ms	164		89.0M	166.4B
YOLOv3-SPP	45.6	45.5	65.2	4.5ms	222		63.0M	118.0B

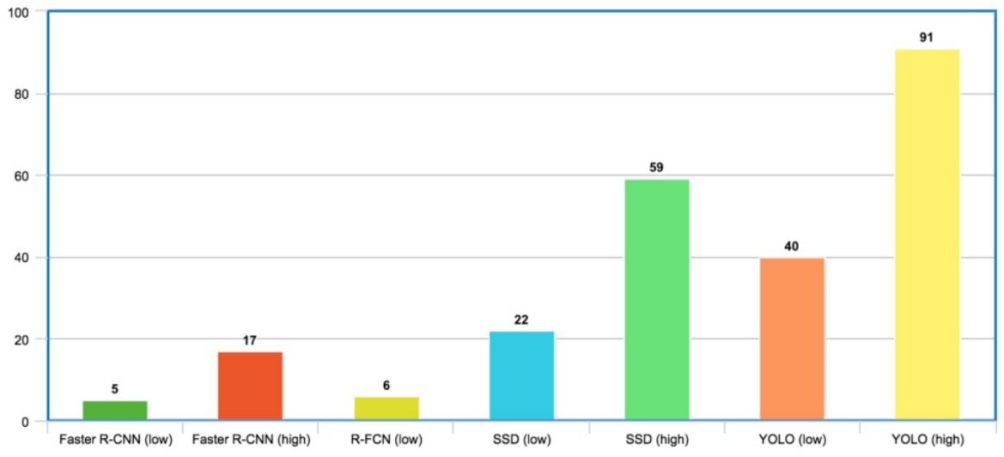
[20]

Another example of the one-step detectron is the SSD (single shot detector) model. This class of models is similar to the YOLO model, as the models also only involve the use of only one pass of a convolutional neural network. However, the model differs from YOLO from a couple of perspectives. First of all, after obtaining the feature map from the backbone CNN model, several convolutional layers are used to obtain features at multiple scales, similar to the use of a gaussian pyramid, and features from all levels of the convolutional network are fed into the next layer. This allows richer information to be fed to the subsequent layers, therefore making the network more fine-grained. Secondly, for each scale, the SSD only generates boundboxes for fixed aspect ratios as opposed to generating boundboxes of any shapes. This significantly reduces the number of potential outputs, simplifying the search [21].



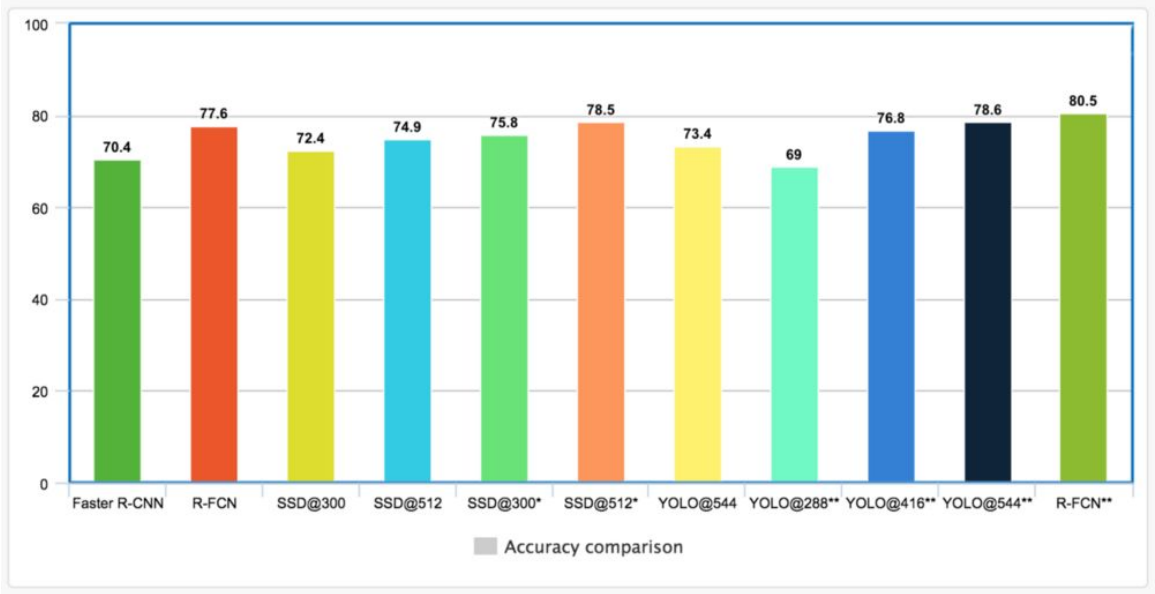
[21]

To choose the best object detection model that will be used by our project, we first need to pick the model that has a reasonable speed. From a comparative study [22] for the performances of these models in different papers. It can be seen that the SSD and YOLO models have a significantly higher FPS, while R-CNN tend to be very slow. This is expected, since faster R-CNN requires two passes in two different neural networks in series, while SSD and YOLO models only require one pass. Since our goal for the project is for developing a real-time algorithm, and hopefully without the need of a GPU, an R-CNN typed model would be infeasible to implement.

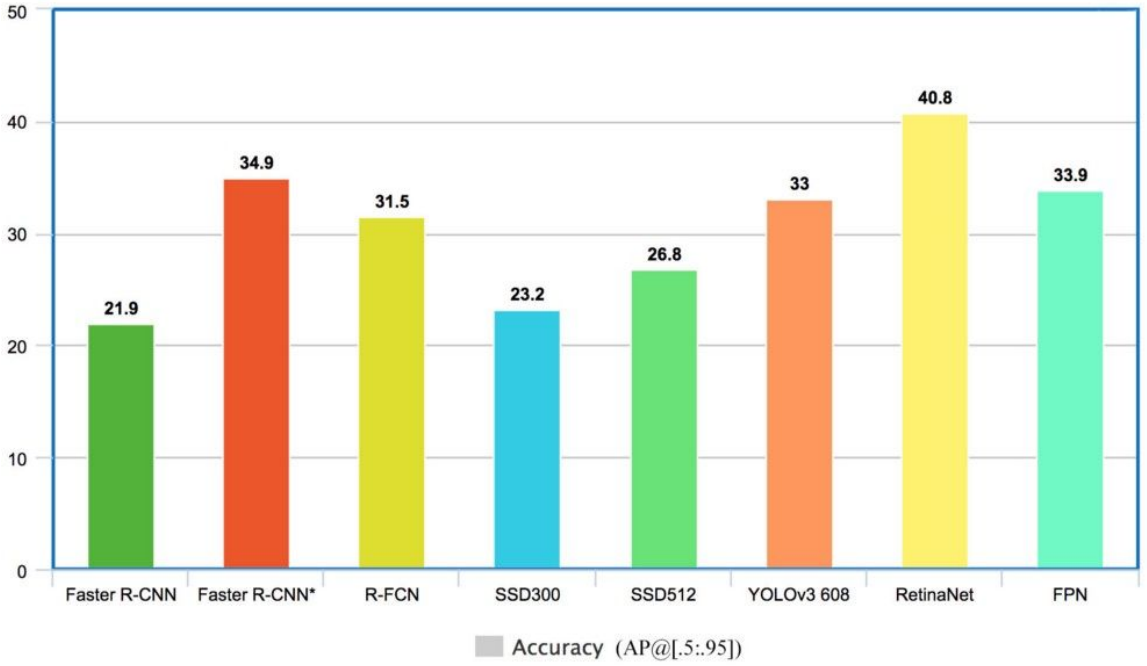


[22]

In terms of mean average precision, SSD and YOLO have similar performance on the older pascal VOC dataset[23] however YOLO performs significantly better on the coco dataset, which has more classes and contains more images. For this reason, as well as the fact that YOLO code are better documented on Github, we have chosen the most recent YOLOv5 as our code for object detection.



[22]



[22]

Camera Calibration - Undistorted Image

Before applying bird eye view transformation to our frames, automatic calibration is a well known computer vision problem that can significantly increase the accuracy of distance measurement by removing any distortion, and will also allow us to measure in real world metrics instead of pixels.

For the best results, the camera should be calibrated using intrinsic and extrinsic parameters. We can use a camera calibrate function in order to get the distortion matrix, and then correct for distortion using the data.

Two major types of distortion currently exist in many modern cameras, namely, tangential and radial distortion [25]. Radial distortion will make straight lines in the 3-D world appear more curved and budged out in the image. While tangential distortion makes some objects appear closer than they are, which usually occurs when the image taking lense is not parallel to the image plane.

(Radial distortion example where chess board lines are not aligned and budged out compared to red lines)

Radial and tangential distortion are solved respectively:

$$x_{corrected} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$
$$y_{corrected} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

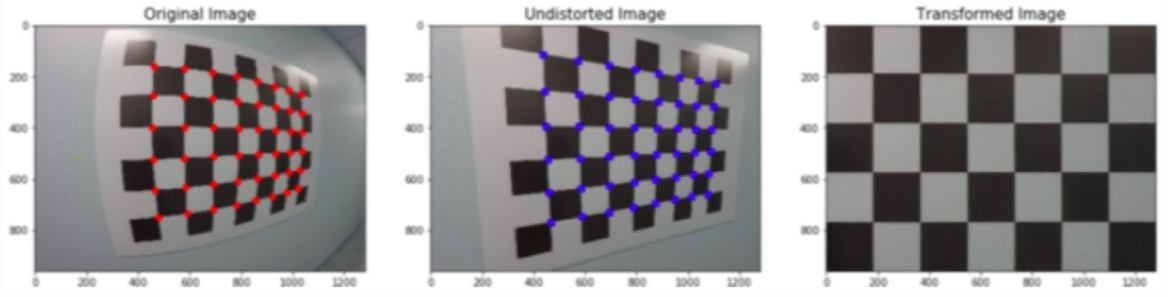
$$x_{corrected} = x + [2p_1 xy + p_2(r^2 + 2x^2)]$$
$$y_{corrected} = y + [p_1(r^2 + 2y^2) + 2p_2 xy]$$

Along with determining the cameras intrinsic and extrinsic parameters which are specific to the camera, determining the distortion coefficients is the process of the calibration.

$$distortion\_coefficients = (k_1 \quad k_2 \quad p_1 \quad p_2 \quad k_3)$$

OpenCV currently supports 3 types of objects for calibration: chessboard, symmetrical circle pattern, and asymmetrical circle pattern [25].

Using many snapshots of the corresponding pattern, and since we know the corners of a 3-D chess board, we can calibrate the camera using the differences in the distance between expected and actual results, and then use this data to undistort images.
In other words, to estimate camera parameters, we need to have 3-D world points of a calibration parameter (chess boards are often used), and then use corresponding 2-D points to solve with OpenCV functions such as cv2.findChessboardCorners.
Example below demonstrating chessboard and after calibration, and then taking a bird eye view of the image [24]:



In our project, we will be assuming a static camera angle with relatively low distortion to the frames, so the impact of projection and distortion will be minimum when estimating distance between bounding boxes. However, ideally we would still want to get rid of any distortion and then compute distances of objects in a bird's eye perspective.

Unfortunately since we don't know anything about the camera that was used for the video input, we cannot estimate the camera intrinsic parameters which are specific to each camera, and also we are unable get a picture of a chessboard taken by the camera in order to calibrate the camera with openCV as well, since OpenCV currently supports chessboard object for calibration.

Ideally, in the future our social distancing detector would want to be able to leverage a proper camera calibration to allow us to map distances in pixels to actual measurable units (e.g metres). Camera calibration is an important step to improving social distance detecting, however we still were able to get decent results due to the nature of the camera angle, and since it is just an approximation of distance it did not impact our results significantly.

Triangle Similarity (Potential alternative to camera calibration)

Triangle similarity essentially attempts to estimate the camera's distance from a known object which is used in order to derive a perceived camera focal length. [triangle_similarity]. To summarize triangle similarity, when we have some marker with a known width and distance from the camera, we can take the 2-D image of the object using our camera and then measure the perceived width. The formula for deriving perceived focal length is F = (P x D) / W, where W is the known width in real world metrics, D is the distance, the P is the number of pixels (or the perceived width). Essentially triangle similarity can be accomplished if we

know the two parameters W and D. Then modern computer algorithms will be able to compute the perceived width of the object and hence derive our focal length. Alternatives to calibrating our camera in OpenCV with chess boards were explored, but unfortunately we were still unable to apply this method as we cannot determine the parameters W and D [34].

Methodology, Experiments and Reviews

Object Detection Model Performance Benchmark:
For the deep learning object detection model, we narrow down our scope to choose between Detectron2, which maintains top class accuracy in object detection, and YOLOv5, which has the fastest speed for in-time object detection. Detectron2 ensembles a variety of object detection models such as Mask R-CNN and Faster R-CNN FPN, which scores top accuracy in open datasets such as COCO and COCO minival[28][29]. YOLOv5 is famous for its amazing speed of training and testing, also small size of model. Since both speed and accuracy are the primary focus for this project, we test both models on public dataset and compare their performance quantitatively to get the best result.

Dataset:
The dataset we are using are Multi-camera pedestrians video from EPFL[30], Joint Attention in Autonomous Driving (JAAD) Dataset and some uncalibrated camera videos donated as 'custom dataset'. Those dataset are selected purposely since our social distance detection program will mainly be used for public area pedestrian walks, and analyzing real time camera footages. The dataset from EPFL contains simulation for multiple person random walking, which can be used to test on the model's computation capability. Dataset JAAD contains footage shot in cars, and videos contain various crosswalks and pedestrians are selected to test on the scalability of the models. Finally, the custom dataset is also selected to increase the variability among datasets and test on models' robustness.

Testing Environment:
This test is set up in the Colab notebook environment, facilitated by default GPU (Tesla K80 GPU) setting. For Detectron2, since there are a lot of model settings, we are using Fast R-CNN R50-FPN backbone on RPN & Fast R-CNN baseline model for top-class prediction accuracy. For YOLOv5, we accept all predefined parameters such as the number of full connections and CNN layers. We use pre trained YOLOv5l weight to maintain a fast GPU speed, while achieving high AP(Average Precision).

Inference FPS\ Dataset	JAAD video 0067	EPFL 6p-c1	EPFL 4p-c0	Custom video
YOLOv5	0.013±0.002s	0.011±0.002s	0.011±0.001s	0.031±0.01s28
Detectron2	0.512±0.201s	0.332±0.521s	0.385±0.028s	0.529±0.511s

Table 1: Object Detection Speed per 300 frames

Inference error\ Dataset	JAAD video 0067	EPFL 6p-c1	EPFL 4p-c0	Custom video
YOLOv5	1/300 frm	3/300 frm	0/300 frm	0/300 frm
Detectron2	0/300 frm	2/300 frm	0/300 frm	0/300 frm

Table 2: Object Detection Accuracy per 300 frames

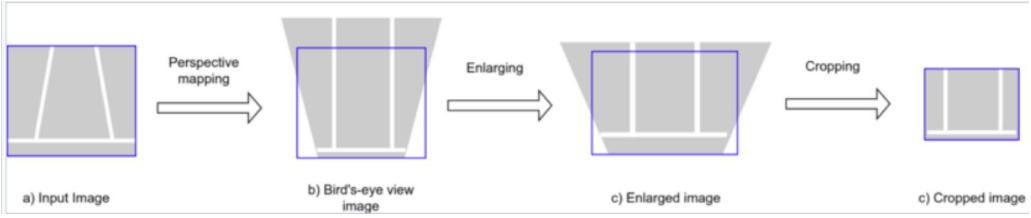
Analysis:

From Table 1, we can see that YOLOv5 is 30-50 times faster than Detectron2. The inference time for each frame is about 0.01-0.03s for YOLOv5, compared to Detectron2 that is about 0.3-0.5s per frame. For custom video, the inference speed tend to increase, this may be caused by the fact that custom videos are not properly calibrated and standardized.

From Table 2, we can see the errors compared between both models are very similar. Those are the conditions when only a part of a person's body enters the frame. Detectron2 has about 0.6 confidence of detecting a ‘person’ class even though there is only one arm in the frame, while VOLOv5 has smaller confidence in those edge conditions, and therefore ruled out by threshold. Since what we care about is the interaction BETWEEN people in pandemic time, the edge case errors are considered as eligible.

Bird Eye View - Warped Image

Bird eye view is a when you warp the image to a top down perspective of a scene and it will allow us to significantly increase the accuracy of distance measurement, especially since euclidean distance sometimes false, for example when 2 people may seem geometrically close by, they can actually be geographically far. Some important assumptions are made with our input video when working with Bird Eye perspective. Firstly, we will assume our input video is a fixed camera angle with respect to the road in the frames. We also need to assume that our surface is planar and free of any interfering obstacles for best results. The steps for processing our image to create a top-down view of a captured scene by applying a homography are: resizing out image to appropriate size to manually select parallelogram surface, apply camera calibration to remove distortion from our frame, and then transforming our image into a bird eye view, and lastly we would enlarge and crop our region of interest [27].



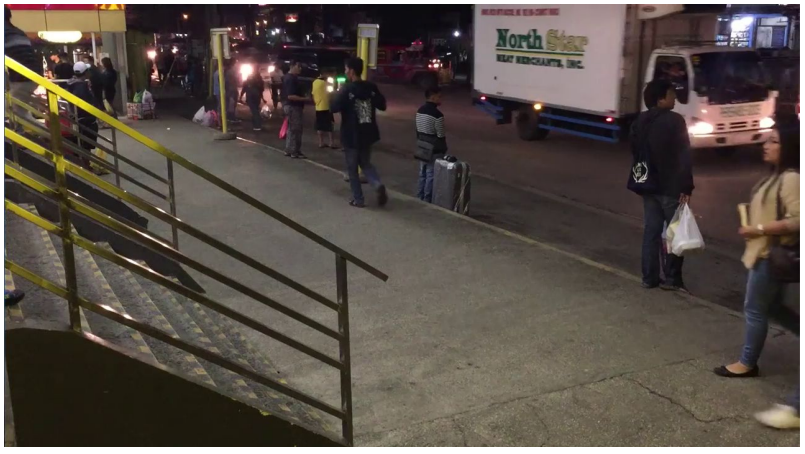
Distance Measurement between bounding boxes

First step involves manually selecting the 4 corners of the scene we want to warp into bird's eye view perspective to get our chosen plane. By using cv2 getPerspectiveTransform, which takes the selected corner points as input, we can determine the transformation matrix. This will calculate the 3x3 perspective transform matrix from four of the corresponding points, which can be used to map the relationship between the original image coordinates to the bird eye view coordinates [31]

We can apply this transformation matrix to each of the bounding boxes we detected, using a pre trained model we discussed earlier, of people in each frame we detected in the first stage. By applying the transformation matrix, we can determine real world coordinates of each bounding box, which is significantly more accurate than measuring distance using original image points as we are now considering coordinates close to the real world [26].

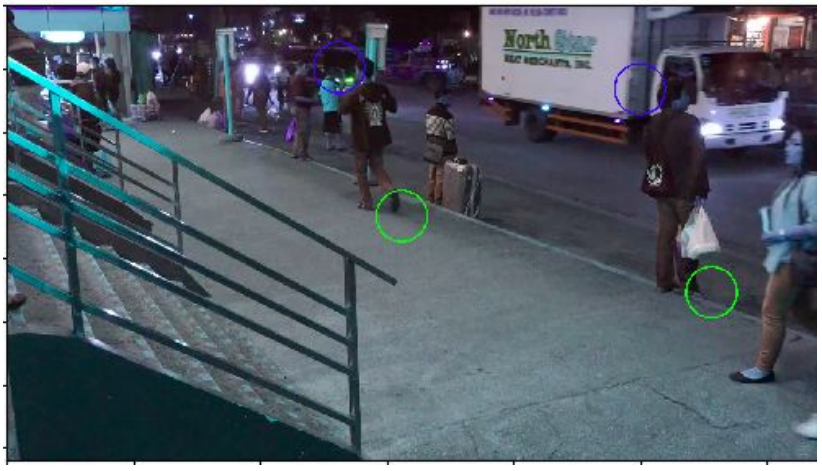
For each person detected, the top left and bottom right bounding box corner points are returned. From these points, we computed the centroid of the box by getting the middle point between them. Using this result, we calculated the coordinates of the point located at the bottom center of the box, which we determined was best to represent the coordinates of a person.

Bird Eye View Example:
Original Frame:



Then by the selecting sidewalk + road as our bird eye view scene we get the following warp:

Bounding box corners circled before warp:



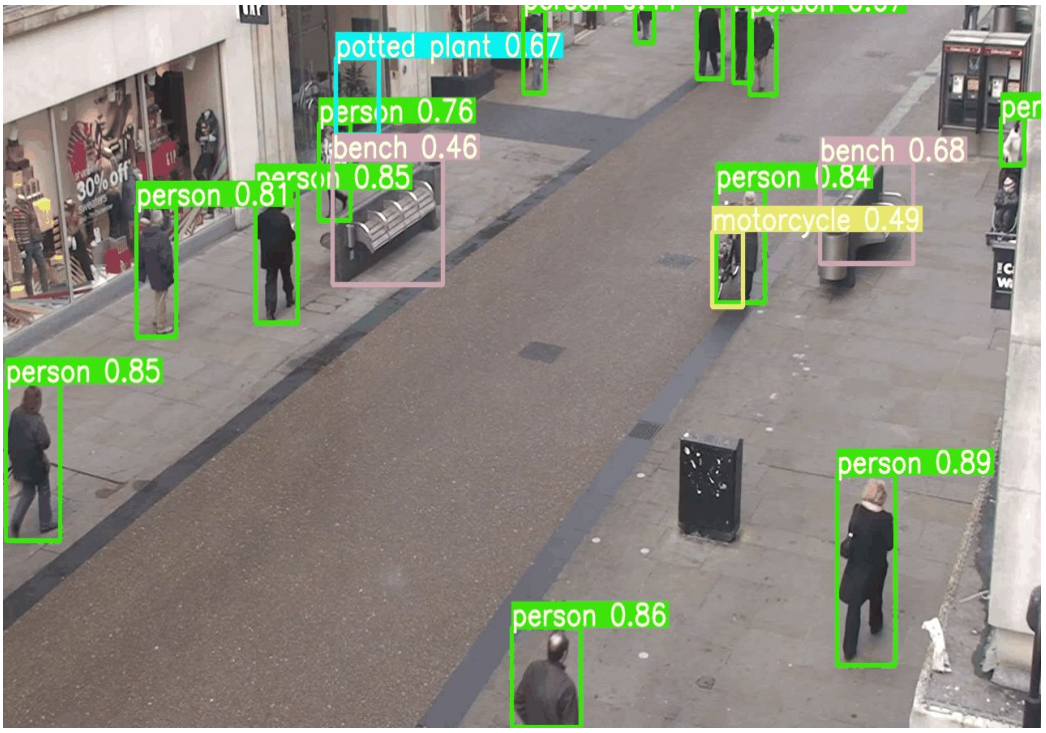
Bounding box bottom center after warp (we use bottom centre as the reference point for each bounding box):

Distance Measurement:

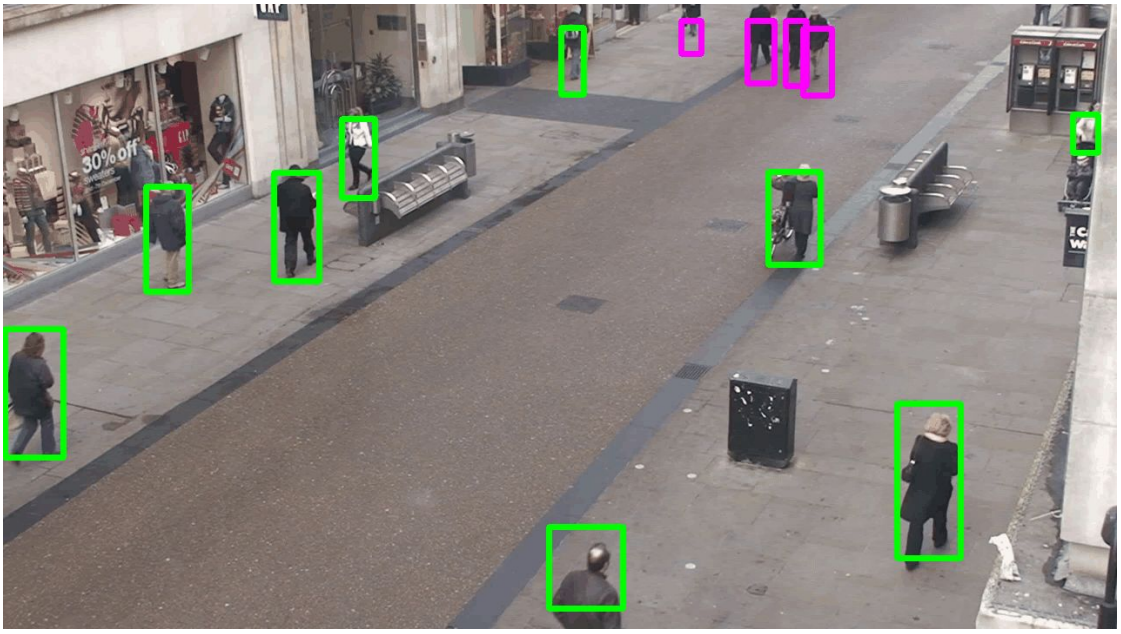
In terms of actually calculating the distance between any two people, we found the euclidean distance for every pair of bottom centroids and stored them. Next, we determined which pair of people violated the social distance threshold and inputted those boxes into our visualizer which highlights their boxes as red.

Social Distancing Demos

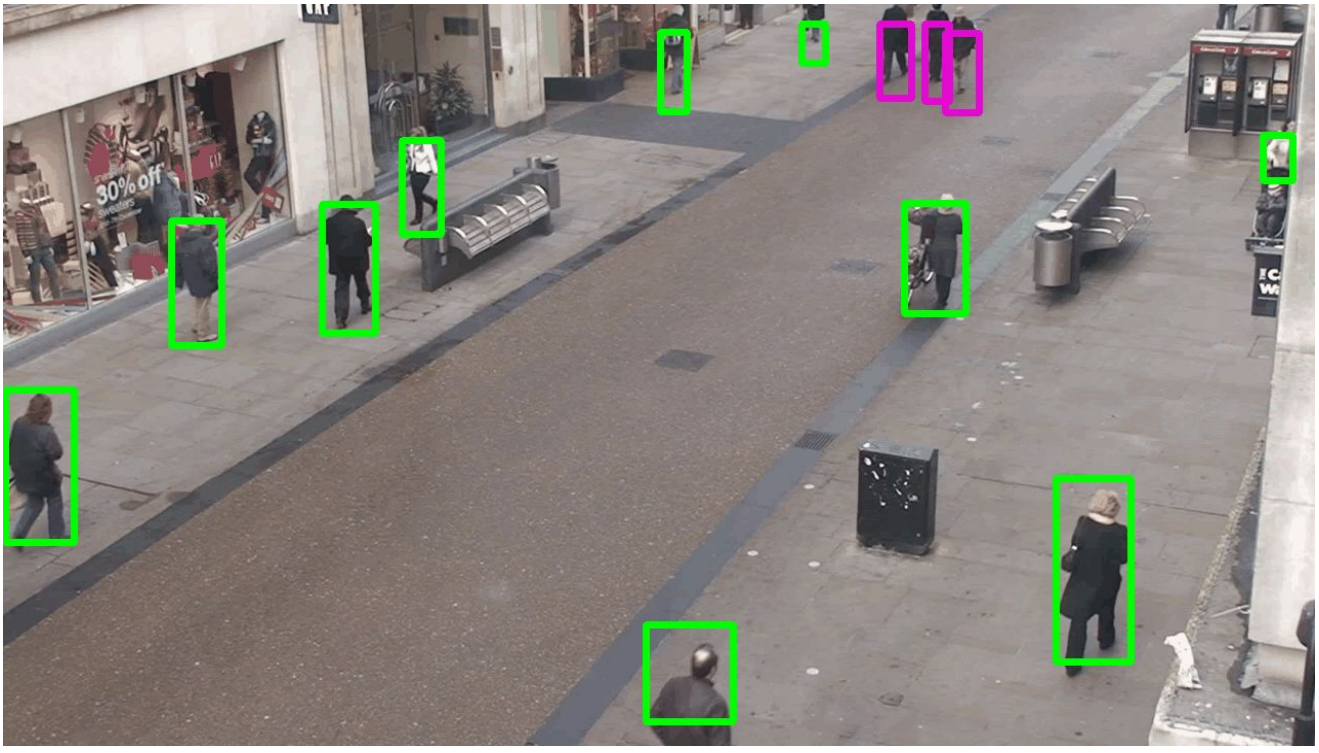
1) **Distance measured with Euclidean Distance:**



(Including social distance violation detection below)



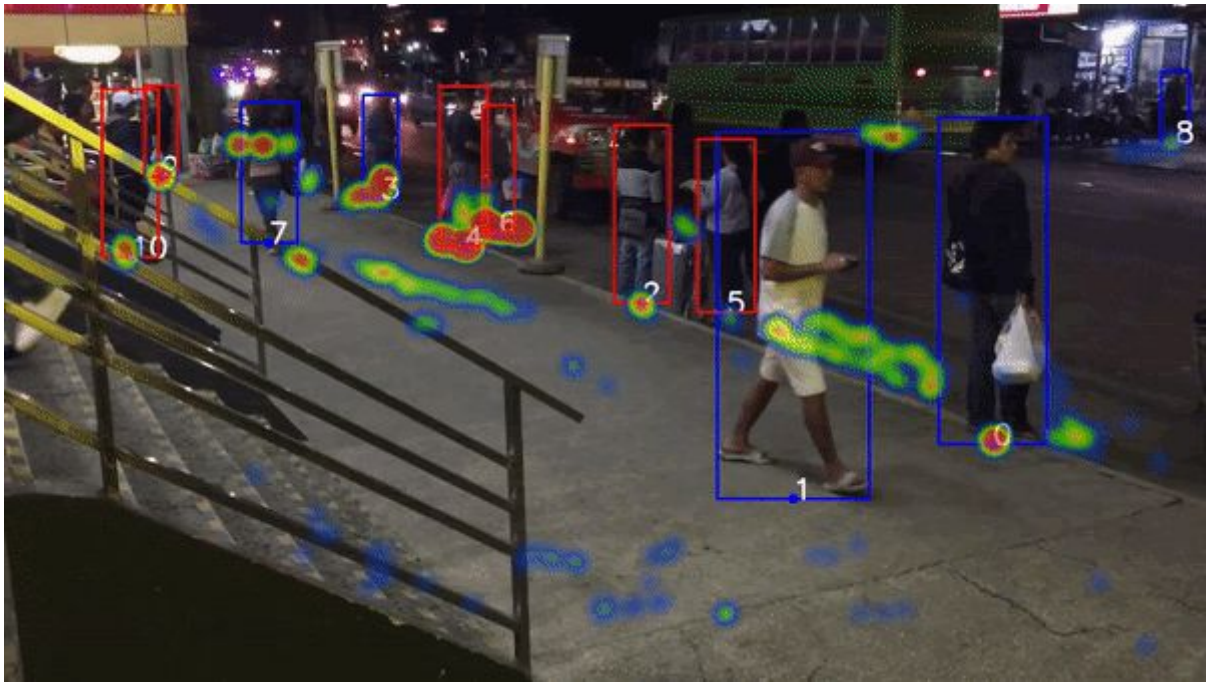
2) Distance measured with Bird Eye coordinates



Overall, by using a static camera angle with a wide view of the scene and relatively low distortion and birds eye warping, along with state of the art object detection models, we were

able to generate a decent social distance detector. Compared between the performance of multiple models across standard video datasets such as JAAD and EPFL, YOLOv5 stands out from our benchmark tests, and is used as our object detection model to explore various transformations that can be applied to improve the accuracy of distance measuring between two people.

3) Distance detection with Heatmap



We also implement heatmaps on the location where people violate the social distance rule powered by python library ‘heatmappy’. The location where violation happens are accumulated from blue to green to red color, signifying the increasing potential danger with elapsed time. This heatmap functionality can not only highlight the dangerous area in pandemic, but also helps to detect public facilities that are poorly designed to make suggestions for public pandemic disease control [35].

Conclusions

With the spread of Covid-19, social distancing has become very important in preventing the spread of the virus and staying healthy. This project can be used as a starting point to encourage people to maintain social distance and be aware when walking into crowded areas. In future iterations of social distance detection, we can improve our accuracy and results with many approaches. For example, YOLOv5 is currently performing prediction in a default mode. But if we tune it to less object class detection and confine the CNN layer based on the video quality, we may save more inference time and obtain higher in-time processing speed. Also for the bird eye view conversion, since we are using homography, it relies on predefined four corners. In the future, we can use a neural net to detect ground in images, and transform to bird-eye view automatically. What's more, as mentioned earlier, we were not able to leverage camera calibration to remove any distortion in our frames. In the future, it would be nice to improve our detector by optimizing our projection and distortion in frames by using a proper calibration for the camera. Doing so will lead to better results and measurable units between distances. This project is intriguing to practice on in the domain of computer vision, while also practical in uses when it comes to serious time such as pandemic Covid-19.

Authors' Contributions

Victor:

- Worked on literature review exploring various ways to estimate and calculate camera calibration, including alternatives such as using triangle similarity.
- Implemented and debugged various ways our model could leverage birds eye projection and how we can implement that into our code
- Explored distance measuring between pairs of people

Haoyan:

- Implementation of YOLOv5 and Detectron2 model baseline, initialize streamline of object detection program in project early stage
- Benchmark test between YOLOv5 and Detectron2, literature resource collection for object detection models and bird eye view problem
- Implementation of heatmap feature in videos

Evan:

- Implemented full pipeline of detection -> homography->distance metrics->display on local machine, integrated both Victor and Haoyan's code
- Sub-problem definition, and Investigated metrics of performance for object detection models in common benchmarks.
- Performed literature review on traditional and Deep-Learning-based object detection models and selecting the best model.
- Ported Haoyan's implementation of the models from Google Collab to Local machine
- Edited Presentation video

References

[1] P. Canada, "Coronavirus disease (COVID-19) outbreak updates, symptoms, prevention, travel, preparation - Canada.ca", Canada.ca, 2020. [Online]. Available: https://www.canada.ca/en/public-health/services/diseases/coronavirus-disease-covid-19.html. [Accessed: 07- Aug- 2020].

[2]P. Canada, "Coronavirus disease (COVID-19): Prevention and risks - Canada.ca", Canada.ca, 2020. [Online]. Available: https://www.canada.ca/en/public-health/services/diseases/2019-novel-coronavirus-infection/prevention-risks.html. [Accessed: 07- Aug- 2020].

[3]"UWB social distancing", Uwb-social-distancing.com, 2020. [Online]. Available: https://www.uwb-social-distancing.com/?gclid=CjwKCAjw9vn4BRBaEiwAh0muDDfoyUuvEQOgL_TSgWDK6RUes-uu1F4hWxUBkpSMDj2jmwDEC2Id3BoC8-AQAvD_BwE. [Accessed: 07- Aug- 2020].

[4]"Social distancing app uses space to save lives", Esa.int, 2020. [Online]. Available: https://www.esa.int/Applications/Telecommunications_Integrated_Applications/Social_distancing_app_uses_space_to_save_lives. [Accessed: 07- Aug- 2020].

[5]2020. [Online]. Available: https://globalreachtech.com/, https://www.itworldcanada.com/blog/need-technology-help-to-maintain-social-distancing/431981. [Accessed: 07- Aug- 2020].

[6]"5 Significant Object Detection Challenges and Solutions", Medium, 2020. [Online]. Available: https://towardsdatascience.com/5-significant-object-detection-challenges-and-solutions-924cb09de9dd. [Accessed: 07- Aug- 2020].

[7]"COCO - Common Objects in Context", Cocodataset.org, 2020. [Online]. Available: https://cocodataset.org/#home. [Accessed: 07- Aug- 2020].

[8]"Open Images V6 - Description", Storage.googleapis.com, 2020. [Online]. Available: https://storage.googleapis.com/openimages/web/factsfigures.html. [Accessed: 07- Aug- 2020].

[9]A. Krizhevsky, I. Sutskever and G. Hinton, "ImageNet classification with deep convolutional neural networks", Communications of the ACM, vol. 60, no. 6, pp. 84-90, 2017. Available: 10.1145/3065386.

[10] P. Piccinini, A. Prati, and R. Cucchiara, “Real-time object detection and localization with SIFT-based clustering,” 03-Jul-2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0262885612000923. [Accessed: 07-Aug-2020].

[11] N. O’Mahony, S. Campbell, A. Carvalho, S. Harapanahalli, G. V. Hernandez, L. Krpalkova, D. Riordan, and J. Walsh, “Deep Learning vs. Traditional Computer Vision,” *Advances in Intelligent Systems and Computing Advances in Computer Vision*, pp. 128–144, 2019.

[12] L. T. Nguyen-Meidine, E. Granger, M. Kiran, and L.-A. Blais-Morin, “A comparison of CNN-based face and head detectors for real-time video surveillance applications,” *2017 Seventh International Conference on Image Processing Theory, Tools and Applications (IPTA)*, 2017.

[13] J. Hui, “mAP (mean Average Precision) for Object Detection,” *Medium*, 03-Apr-2019. [Online]. Available: https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173. [Accessed: 07-Aug-2020].

[14] Intellica.AI, “A Comparative Study of Custom Object Detection Algorithms,” *Medium*, 18-Dec-2019. [Online]. Available: https://medium.com/@Intellica.AI/a-comparative-study-of-custom-object-detection-algorithms-9e7ddf6e765e. [Accessed: 07-Aug-2020].]

[15] R. Gandhi, “R-CNN, Fast R-CNN, Faster R-CNN, YOLO - Object Detection Algorithms,” *Medium*, 09-Jul-2018. [Online]. Available: https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e. [Accessed: 07-Aug-2020].

[16] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation,” *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[17] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[18] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017.

[19] Intellica.AI, “A Comparative Study of Custom Object Detection Algorithms,” *Medium*, 18-Dec-2019. [Online]. Available: https://medium.com/@Intellica.AI/a-comparative-study-of-custom-object-detection-algorithms-9e7ddf6e765e. [Accessed: 07-Aug-2020].

[20] Ultralytics, “ultralytics/yolov5,” *GitHub*, Jul-2020. [Online]. Available: https://github.com/ultralytics/yolov5. [Accessed: 07-Aug-2020].

[21] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “SSD: Single Shot MultiBox Detector,” *Computer Vision – ECCV 2016 Lecture Notes in Computer Science*, pp. 21–37, 2016.

[22] J. Hui, “Object detection: speed and accuracy comparison (Faster R-CNN, R-FCN, SSD, FPN, RetinaNet and...,” *Medium*, 26-Mar-2019. [Online]. Available: https://medium.com/@jonathan_hui/object-detection-speed-and-accuracy-comparison-faster-r-cnn-r-fcn-ssd-and-yolo-5425656ae359. [Accessed: 07-Aug-2020].

[23]“Visual Object Classes Challenge 2012 (VOC2012),” *The PASCAL Visual Object Classes Challenge 2012 (VOC2012)*. [Online]. Available: http://host.robots.ox.ac.uk/pascal/VOC/voc2012/. [Accessed: 07-Aug-2020].

[24] “Camera Calibration,” *OpenCV*. [Online]. Available: https://docs.opencv.org/3.4/dc/dbb/tutorial_py_calibration.html. [Accessed: 07-Aug-2020].

[25] “Camera Calibration¶,” *OpenCV*. [Online]. Available: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_calib3d/py_calibration/py_calibration.html. [Accessed: 07-Aug-2020].

[26] basileroth75, “basileroth75/covid-social-distancing-detection,” *GitHub*. [Online]. Available: https://github.com/basileroth75/covid-social-distancing-detection. [Accessed: 07-Aug-2020].

[27] “RidgeRun's Birds Eye View project research,” *developer.ridgerun.com*. [Online]. Available: https://developer.ridgerun.com/wiki/index.php?title=Birds_Eye_View%2FIntroduction%2FResearch. [Accessed: 07-Aug-2020].

[28] “Papers with Code - COCO test-dev Benchmark (Object Detection),” *The latest in machine learning*. [Online]. Available: https://paperswithcode.com/sota/object-detection-on-coco. [Accessed: 07-Aug-2020].

[29]“Papers with Code - COCO minival Benchmark (Object Detection),” *The latest in machine learning*. [Online]. Available: https://paperswithcode.com/sota/object-detection-on-coco-minival. [Accessed: 07-Aug-2020].

[30] F. Fleuret; J. Berclaz; R. Lengagne; P. Fua, F. Fleuret, J. Berclaz, R. Lengagne, and P. Fua, “Multi-camera pedestrians video,” *CVLAB*, 01-Jan-1970. [Online]. Available: https://www.epfl.ch/labs/cvlab/data/data-pom-index-php/. [Accessed: 07-Aug-2020].

[31] ayushmankumar, “Perspective Transformation - Python OpenCV,” *GeeksforGeeks*, 09-Jul-2020. [Online]. Available: https://www.geeksforgeeks.org/perspective-transformation-python-opencv/. [Accessed: 07-Aug-2020].

[32] A. Rosebrock, “4 Point OpenCV getPerspective Transform Example,” *PyImageSearch*, 18-Apr-2020. [Online]. Available: https://www.pyimagesearch.com/2014/08/25/4-point-opencv-getperspective-transform-example/. [Accessed: 08-Aug-2020].

[33] “Camera Calibrator,” *What Is Camera Calibration? - MATLAB & Simulink*. [Online]. Available: https://www.mathworks.com/help/vision/ug/camera-calibration.html. [Accessed: 08-Aug-2020].

[34] “Find distance from camera to object using Python and OpenCV,” *PyImageSearch*, 18-Apr-2020. [Online]. Available: https://www.pyimagesearch.com/2015/01/19/find-distance-camera-objectmarker-using-python-opencv/. [Accessed: 08-Aug-2020].

[35] A. Pai, “Build your Social Distancing Detection Tool using Deep Learning,” Analytics Vidhya, 28-Jun-2020. [Online]. Available: https://www.analyticsvidhya.com/blog/2020/05/social-distancing-detection-tool-deep-learnin. [Accessed: 08-Aug-2020].