CSC420

Team GDM

Gabriel Deza (1004389342)
Danial Hasan (1003132228)
Mengyu Yang (1003922767)

December 18 2020

1 Abstract

One of the most effective methods for limiting the spread of COVID-19 is by following social distancing. Our work consists of creating a pipeline which detects social distancing violations from video footage of a public space. This can be used to enforce social distancing violations or perform contact tracing if we discover that an individual has contracted the virus. Although there exists some preliminary work in social distancing detection, they do not take into account social bubbles, such as couples or families. Hence, family members walking together are considered to be violating social distancing, although this should not be the case in reality. Analysis shows that previous works have a false positive detection rate of over 63% due to not accounting for groups. Our work accounts for such groups and significantly reduces this false positive rate where in certain scenarios, we are able to achieve a 0% false positive rate.

2 Introduction

With the rise of the COVID-19 pandemic, social distancing has become enforced in public places in most countries. It has been shown to be a highly effective yet simply method to prevent the spread of the virus. Although social distancing has shown great results in certain countries, it is unfeasible to assume that every individual is constantly assessing their distance to others throughout the whole day. Hence, a social distancing detector has the following three benefits. First, a detector can measure the relative amount of violations within an area. This data can be used to determine if an area needs to be redesigned to abide by COVID-19 regulations. Secondly, this tool can be used for social distancing enforcement, where some audiovisual cue can be triggered when too many violations occur. Lastly, a social distancing detector can allow for contact tracing by keeping track of the people who have been in close proximity with a COVID-19-positive person.

However, current existing methods for social distancing detection does not take into account this last point. Each human is not tracked by a unique identifier, meaning the system is unable to differentiate between specific people. Additionally, current systems do not take into consideration people who belong to the same social bubble (families, spouses, etc.) and classify them as violating social distancing as well. This leads to a high rate of false positives. Our work addresses these two problems by introducing both person tracking and automatic group detection.

3 Related Works

3.1 Social Distancing Detection

A social distancing detection system by Yang et al. [1] implements an end-to-end pipeline. Beginning with a dataset of surveillance footage from a public space, the system uses an object detection model to detect humans from each frame of video. These positions are converted into a top-down coordinate system using homography, on which distances between

people are calculated. Their full implementation is available on a github link in their original work. We used their pipeline implementation as skeleton code to create our own pipeline. To compare their results and our own, we use similar plotting functions to better illustrate the differences between our works.

A similar pipeline approach is used by Landing AI [2] for the purposes of helping companies ensure social distancing in the workplace. Although the methods from [1] and [2] are able to detect violations, both consider every human within the frame as strangers. In reality, this is not the case, since families, couples, and friends tend to walk together. Upon observation, we find that a majority of the detected violations from the two systems are between such groups. This leads to a primary goal of our project — being able to account for these groups of individuals to decrease the high rate of false positives.

3.2 Pedestrian Detection

The pedestrian detection problem falls under the general umbrella of object detection methods that seek to identify objects in images/video. Specifically, pedestrian detection is dedicated to identifying pedestrians in two main scenarios: surveillance and self-driving. An in depth survey on the topic of pedestrian detection that considers models, datasets, and challenges can be found in [3]. For the case of social distancing detection, we consider only surveillance related methods to be relevant as that is the primary data source. Additionally, we consider the general case of object detection to provide a stable baseline approach that can be easily replicated. A general survey of object detection can be found in [4], which details object detection method advances for the past 20 years.

The general taxonomy of pedestrian (and object) detection models can be split into hand-crafted models and deep learning models. Most hand-crafted techniques use feature extractor algorithms such as SIFT [5], Haar wavelets, and HOG to extract image features for various regions in the image, which are then passed to a classification algorithm, often using SVM [6] or a cascade of classifiers using AdaBoost [7], to classify whether the desired object is present, as in [8] and [9]. One of the most popular and seminal techniques was introduced by Dalal in [10] which utilizes a Linear SVM to classify humans based on HOG features.

Generally, the hand-crafted object detection techniques were superseded by deep learning based approaches that could be trained end to end and could generalize to general object detection without the need for hand crafted features. Deep learning based object detection models can be further reduced into two categories: one-shot and two-shot. One shot methods such as the YOLO (You Only Look Once) family [11, 12, 13], SSD (Single Shot Detector) [14], and RetinaNet [15], are directly applied over regions in an image that are densely distributed with varying scales and aspect ratios. In contrast, two-shot methods such as Faster-RCNN [16], Mask-RCNN [17], and FPN (Feature Pyramid Network) [18] are based on an R-CNN [19] approach which involves a region proposal network that proposes a sparse set of regions, which are then classified as background or foreground, the foreground regions are then classified into object classes and their bounding boxes are regressed. Modern methods have been shown to perform well on object detection benchmarks such as Pascal VOC [20] and MS COCO [21], and provide robust and generalizable implementations for use in many

object detection contexts.

3.3 Optical Flow

Optical flow is a method used for estimating the movement of objects within consecutive frames of an image. There are several methods for calculating this, with one widely used method being Lucas-Kanade [22]. It assumes that the displacement of objects between two frames is small and the same within a neighbourhood of the point under consideration. By solving for a region of pixels rather than a single point, Lucas-Kanade is less affected by noise. These characteristics make Lucas-Kanade a potential method to be used for person tracking within our pipeline.

4 Methods

The general pipeline for our method which accounts for social groups is shown in Figure 1. In this section, we begin by discussing the datasets we experimented with. Section 4.2 describes the 4 object detection models used for our human detection method. Section 4.3 explains the required methods to achieve a bird's-eye-view of our dataset. Section 4.4 introduces 2 methods used for tracking humans across frames. Lastly, Section 4.5 describes the method to determine social groups across frames.

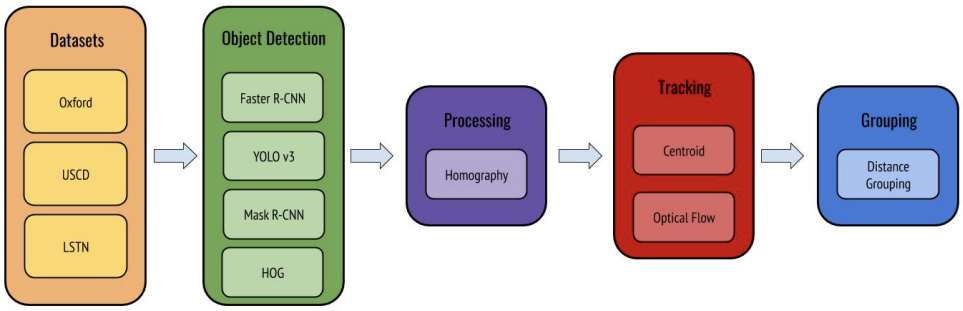


Figure 1: Overview of our project pipeline

4.1 Datasets

Three datasets were used for experiments, the Oxford Town Centre Dataset [23], the UCSD Pedestrians dataset [24], and the LSTN Fudan-ShanghaiTech dataset [25]. All of them consist of surveillance footage in a public space and are used for crown counting and pedestrian detection tasks. A sample frame and other information for the datasets are shown in Table 1. Each frame from the datasets is considered an image, $\mathbf{I}$, and the entire dataset is a video, $\mathbf{V} = \{I_1, I_2, ...\}$.

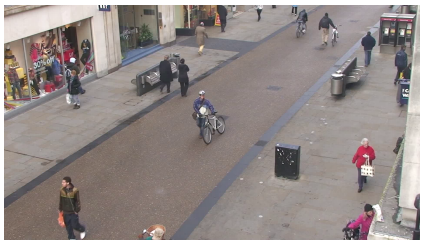
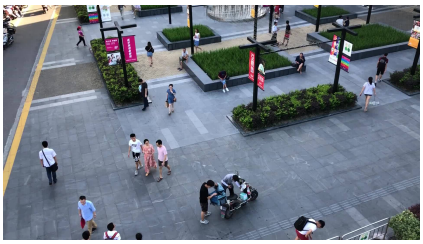
Oxford Town Centre	UCSD Pedestrian Dataset	LSTN Dataset
1920 ×1080 @ 25 FPS Colour	238 ×158 @ 10 FPS Black and White	1920 ×1080 @ 25 FPS Colour
		

Table 1: Statistics and sample frame from the 3 datasets used in our experiments.

4.2 Pedestrian Detection

For the subtask of pedestrian detection, we seek to compare 4 different object detection models for detection and computational performance. The models we consider are HOG+SVM from [10], which represents an advanced hand-crafted model, along with deep learning based models YOLOv3 [13], Faster-RCNN [16], and Mask-RCNN [17]. These models are chosen both for their performance, which has been verified over time in research and application domains, and their availability for use. Each model is considered as a function $f_{model} : \mathbf{I} \rightarrow \mathbf{D}$. Where $\mathbf{I}$ is an input image/frame and $\mathbf{D} = \{D_i\}_n$ is a set of n detected objects, $D_i = (\mathbf{b}_i, l_i, s_i)$. Here $\mathbf{b}_i = [x_1, y_1, x_2, y_2]$ is a bounding box for the object, l_i is the label associated with the object, and s_i is a confidence score pertaining to the prediction for the detected object.

For models that predict multiple labels, i.e. $l_i \in \{object_1, object_2, ...\}$, we only consider the set of predictions for *humans* and with confidence scores above a certain threshold, ϵ: $\hat{\mathbf{D}} = \{D_i, D_i \in \mathbf{D} | l_i = human, s_i > \epsilon\}$. For each detection we now define the principal point for that human, $\mathbf{p}_i^{img} = [x_1 + x_2, y_2]/2$, as the center point of the bounding box's bottom edge in the image space.

4.3 Bird's Eye View

From the previous step, we obtain detected pedestrians as coordinates in the image space. However, to determine the real world distance between humans, the coordinates must be transformed to a real world space measured in meters. This transformation is a perspective transformation from the camera view to the bird's eye view (BEV). We define $h : \tilde{\mathbf{p}}^{img} \rightarrow \tilde{\mathbf{p}}^{bev}$, as a homography transformation from the homogeneous image coordinate, $\tilde{\mathbf{p}}^{img} = [p_x^{img}, p_y^{img}, 1]$ to 2-dimensional BEV coordinates, $\tilde{\mathbf{p}}^{bev} = [p_x^{bev}, p_y^{bev}, 1]$, with the ground plane considered as $z = 0$. The complete transformation is then the well known homography transform **cite szeliski book**:

$$h(\tilde{\mathbf{p}}^{img}) = \tilde{\mathbf{p}}^{bev} = \mathbf{H}\tilde{\mathbf{p}}^{img}$$

Here, $\mathbf{H} \in \mathbb{R}^{3\times3}$, is the homography matrix describing the full projection from image view to BEV. The final BEV point is: $\mathbf{p}^{bev} = [p_x^{bev}, p_y^{bev}]$ for every human detected.

4.4 Tracking

4.4.1 Why is tracking necessary for grouping?

As described in section 4.2, the output of all 4 object detection models is a list of (x, y) positions for detected humans in the frame. To define a social group, the movement and positions of the detected humans must be tracked and analyzed across multiple frames. Hence, before performing group detection, we must develop a method that assigns a unique human ID to each individual and consistently tracks this information across all frames, while accounting for people leaving and entering the frame.

4.4.2 Why is tracking a non-trivial task?

There are a number of reasons why tracking is non-trivial. First, we cannot assume that the object detection model returns the positions of humans in the same order between frames. An example of this problem is shown in Figure 2. In frame i, the output from the detection model is $[P_3, P_2, P_1]$. In frame $i+1$, the output from the detection model is $[P_3, P_1, P_2]$. As a result, we cannot simply rely solely on the order of the model output.

To solve this problem, we develop two methods, centroid tracking and optical flow tracking, as described in Sections 4.4.3 and 4.4.4, respectively. The key intuition behind both methods is that between a single frame, humans move a small amount. This assumption is valid in our case, as only 40ms occur between each frame for the Oxford and LSTN dataset and 100ms for the UCSD dataset.

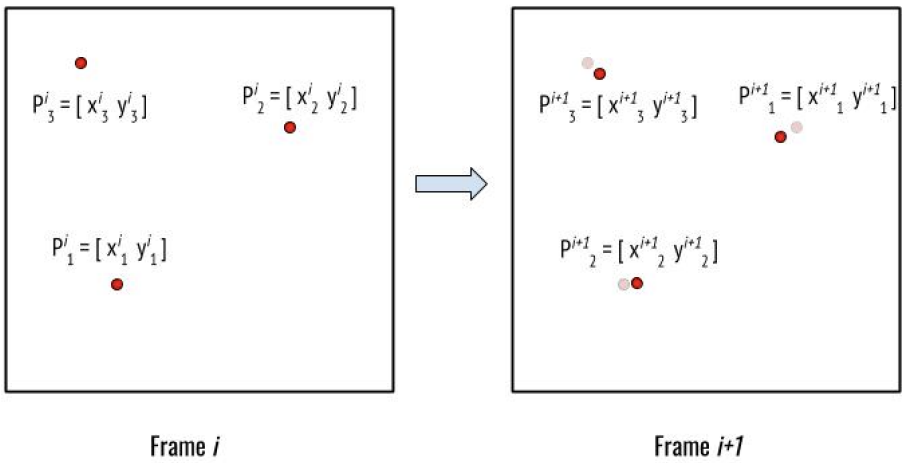


Figure 2: Illustration of the problem of determining human ID solely from the order of predictions of object detection model.

4.4.3 Centroid Tracking

The centroid tracking algorithm is defined is defined at a high level as follows. For the first frame $i = 0$, assign every human a distinct human ID from 0 and increasing by one. For every human in frame $i \neq 0$, we assign them the human ID of the closest human in the previous frame $i - 1$ to their current position in frame i. If a human in frame i has no one in the previous frame that was close by to them (i.e. under a certain threshold), there are two possible scenarios:

1. **Scenario 1:** They are a human that we have never encountered before as they just entered the region of interest. Assign them a new human ID.
2. **Scenario 2:** This human is not new, but for some reason was not detected in frame $i - 1$. We call this problem *phasing*. This occurs when a person is correctly detected in frame $i - 2$, not detected by the model in frame $i - 1$, but then detected again in frame i. In this case, this person does not fall under Scenario 1 as they are not a new person and we do not want to assign them a new ID. In this case, we look at the past 15 frames and see if someone was consistently in the past 15 frames, except the very last one. If such an individual exists, assign them that human ID. If there exists no such individual, they fall under Scenario 1.

Repeating this process for all frames, we now have a human ID for each detected human across the entire dataset. We can now use this data to determine groups, which is described in Section 4.5.

4.4.4 Optical Flow

The second method for tracking the movement of individual people is by using optical flow. At a high level, the object detection model, specifically Mask R-CNN, is run at intervals of every k frames to update the locations of every person. In between these update frames, optical flow is used to track the movement of people instead.

For optical flow to work, each person must have a set of features on them for the algorithm to track. This occurs during the update frame when the objection detection model is run. The model returns a list of masks corresponding to individual humans. For each mask, it is given as input to the function `cv2.goodFeaturesToTrack()`. Essentially, this function does Shi-Tomasi Corner Detection within the mask region of the frame to select multiple corners that are ideal for optical flow. The corresponding set of tracked points for every detected human is stored in memory.

For the next $(k - 1)$ frames, the tracked points for every person is given as input to the optical flow implementation. In our project, we used `cv2.calcOpticalFlowPyrLK()`. The return of this function are the locations of the input points within the next frame. This new set of points is stored to be used for the next frame.

Multiple points for each person is used for robustness. However, for visualization purposes, each person should only have one point associated to them. This single point is calculated by taking the centroid of the person's tracked points. The centroid calculated from each

frame for every person is stored in memory so that the entire path of a person can be analyzed.

On the next update, we are provided with a new list of masks. To map this new set of masks to our current set of tracked points, we determine if a centroid overlaps with a new mask. If it does, then that new mask is attributed to this existing person. If a new mask does not have an overlapping centroid, this is considered as a potential new person. Before assigning the mask a new human ID however, we first account for phasing.

Phasing in Optical Flow As first described in Section 4.4.3, phasing can also occur during optical flow tracking as well, where the object detection model fails to detect a specific human for one frame but detects it again in a later frame. To ensure a person is continuously tracked despite phasing, we store the set of deleted points to look back on. When a centroid does not overlap with any mask during the update frame, we remove the corresponding points from being tracked. However, we store it in memory along with information regarding its position and last tracked frame.

On an update frame, if there is a mask that does not overlap with any tracked centroids, this suggests a new person has entered the frame. However, it can also be the case that the person has just phased back in. Before assigning the mask a new human ID, we first check our list of removed centroids. If any of them are close to the mask in both distance and time, then we assign the old human ID of the deleted centroid back to the mask. As a result, if a person phases out, their tracked points are removed but not forgotten. Once they phase back in, we are then able to assign their old human ID back to them.

4.5 Grouping

Using the human IDs from either centroid or optical flow tracking, we are then able to determine groups of 2 or more people. Our grouping function first assigns every pair of two humans X and Y as members of the same group if all of the following conditions are met:

1. Human X and human Y are both in the region of interest together for at minimum 80% of their individual time in the region of interest
2. During human X and human Y's frames where they are both in the region of interest, they must stay at a relatively close and constant distance
3. During human X and human Y's frames where they are both in the region of interest, they must travel in the same general direction

The first condition captures the time aspect of a group where the humans in a group are likely to enter and exit the region of interest at the same time. The second condition is intuitive as a couple or a family are likely to walk together, keeping at maximum a meter distance between one another. Furthermore, we expect a group to behave in unison and not have a member drift off. The last condition is an additional check to account for any drifting that could occur within a group.
By looking at every single pair of possible humans X and Y, we can determine the groups of

humans across all N frames. It is important to note that our grouping is transitive, meaning that if X and Y are in a group and Y and Z is as well, by the transitive property, X and Z are in the same group. This is important for groups of 3 or more people walking side by side, where the humans on the very edges of the group might not satisfy the second condition due to them being far away from each other. This property allows for such scenarios to be counted as a single group, and not multiple sub-groups which would result into constant false social distancing detections.

5 Experiments

We devise a multitude of experiments to test the various steps in our pipeline and the various methods used in each step. First, we note the general experimental setup relating to the datasets, homography transforms, computational hardware, and model implementations, . Then we discuss experiments to test and compare **(1)** Pedestrian Detection, **(2)** Centroid Tracking and Optical Flow Tracking, **(3)** Grouping, and **(4)** End to End results.

5.1 Experimental Setup

For the general experimental setup, we utilize three sources of pedestrian data from surveillance video, as noted in Section 4.1. The Oxford dataset provides a homography transformation matrix from the image space to BEV, and is used directly in the demos of [1] and [2]. Thus, it can be used for direct comparison against previous methods and provides a reliable perspective transformations to measure distance. For the UCSD and LSTN datasets, we approximate the homography transform by estimating the length of bicycles in both datasets and using that to approximate the BEV size of rectangular patterns on the ground. The rectangle corners in the image space and the BEV space are taken as key points and fed into `cv2.findHomography()` to compute the homography matrix, $\mathbf{H}$.

The experiments are all run on a PC with an AMD Ryzen 2700x @ 4.0 GHz, an Nvidia GTX 1660, with 16GB of RAM and running Ubuntu 20.04. The object detection models are all pretrained and their respective implementations (and weights) are used and/or adapted from the following sources, **OpenCV**: HOG+SVM, Github user **eriklindernoren** [26]: YOLOv3, and **TorchVision**: Faster-RCNN and Mask-RCNN.

5.2 Pedestrian Detection on Various Datasets

First we compare pedestrian detection and computational performance for the object detection models in question. This establishs which models we can consider for further analysis in the object tracking and grouping experiments. Additionally, we determine the variation in performance across the three datasets to see how robust object detection can be for varying camera angle, resolution, color space, and pedestrian density.

Detector	Avg. Inference Time (s)	mAP (%)
HOG + SVM	0.12 — 0.05 — 0.12	N\A
YOLOv3	0.05 — 0.03 — 0.05	43.0
Faster-RCNN	0.17 — 0.15 — 0.17	42.7
Mask-RCNN	0.23 — 0.18 — 0.26	60.0*

Table 2: Performance of detectors on MS COCO and average inference time for each of the datasets (Oxford Town — UCSD — LSTN). ** Mask AP, the box AP is similar to Faster-RCNN*

.

5.3 Centroid and Optical Flow Tracking

The next experiment pertains to inspecting the performance of the pedestrian tracking methods. Here we analyze the tracking performance for detected pedestrians using Faster-RCNN on the Oxford dataset. This will provide a direct comparison to [1]'s results by holding both the object detection model and dataset consistent. A presentation of results and a discussion of difficult and failed cases is provided to highlight the strengths and weaknesses of both methods.

5.4 Grouping

Further study concerns the finding of pedestrian groups that are consistently together in frames. A brief presentation of grouping results is provided to determine if the algorithm can correctly and reliably predict pedestrian groups after tracking takes place.

5.5 End to End Results

Finally, we run the entire social distancing detection pipeline shown in Figure 4 on the datasets. Here we study the accuracy of group detection on the false positive rate and the overall computational performance of the pipeline. Results are presented for the Oxford datset to compare directly against the Yang Yurtsever results, and results are presented for the LSTN dataset to showcase the detection of larger groups in denser areas.

6 Results

6.1 Pedestrian Detection

To establish a baseline comparison for the pedestrian detection models we use, we consider their average inference time on each of the datasets using the hardware listed in Experimental Setup and their stated performance on the MS COCO dataset from the reference implementations. The detection performance for the HOG based detector is omitted since it precludes testing the MS COCO dataset. The results are summarized in Table 2. Note that only YOLOv3 consistently performs near real-time for all datasets considered, and all the deep learning based detectors have strong performance for object detection.

To compare the pedestrian detection performance on the datasets used in this analysis, we show some example detections with bounding boxes or masks in Figure 3. Note that HOG+SVM fails to detect many pedestrians in the image, whereas the deep learning based models have near perfect detection accuracy. This results holds across the UCSD and LSTN datasets as well, with HOG+SVM failing to detect any pedestrians in the UCSD dataset due to the image being low resolution and grayscale. An example detection for both UCSD and LSTN is shown for YOLOv3 in Figure 4 to demonstrate that pedestrian detection is robust across dataset resolution, image colour, and camera angle. We note that all the pedestrians are correctly detected in the UCSD dataset, a result that holds fairly consistents across all the frames, and that most pedestrians are detected in the LSTN dataset despite the difficult camera angle, variety in human pose, and occlusions.

(a) HOG + SVM

(b) YOLOv3

(c) Faster-RCNN

(d) Mask-RCNN

Figure 3: Example detections for all detection models on the Oxford Town Centre dataset.

6.2 Centroid Tracking

Figure 5 shows a visualization of centroid tracking on frame 4 of the Oxford dataset. Each colour corresponds to a different person with a unique human ID. Each human has 4 points, 3 of which are their past positions in the 3 previous frames to show a small portion of their path. We see that points of the same colour are grouped tightly together, showing that centroid tracking works properly. A full animation of 500 frames of the Oxford dataset is available here.

Although centroid tracking works relatively well, there are scenarios that are difficult to account for. When a human is about to exit the region of interest and another human enters at the exact same position and time, instead of assigning that new human a new ID, they

(a) UCSD

(b) LSTN

Figure 4: Example detection by YOLOv3 on the UCSD and LSTN datasets.

pick up the ID of the human who just exited. We have found that this is less of an issue the closer this scenario occurs to the camera as it is easier to distinguish what is going on. Furthermore, the bounding box of a human can sometimes include noise, such as a human's bike as shown in Figure 6. This leads to large shifts of the bounding box in consecutive frames which causes the centroid tracking algorithm to believe that this human appeared out of nowhere and should be assigned a new ID. The rightmost image of Figure 6 also shows a case where the model mistakes a mannequin for a human.

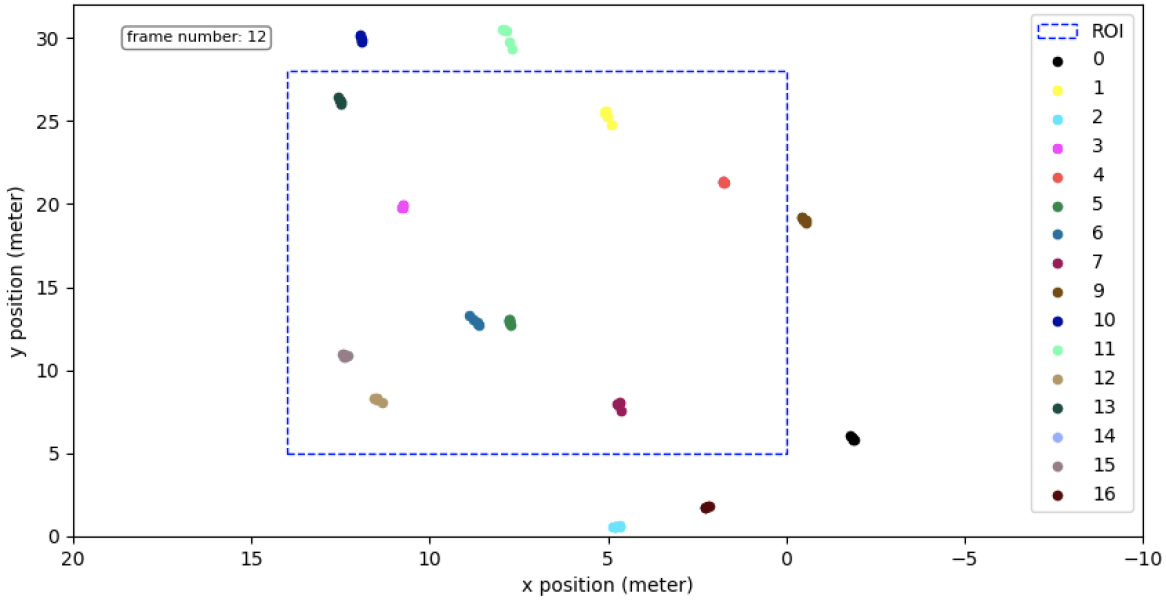


Figure 5: Humans within ROI in Oxford dataset at frame 4. Each human has 4 points, indicating their position at frame 4,3,2 and 1.

Figure 6: Illustration of how objects such as a bike can be caught in the bounding box. Furthermore, the object detection model sometimes picks up background noise such as mannequins as humans

6.3 Optical Flow Tracking

Tracking with optical flow works quite well for most of our experiments. Human labels remain consistent throughout the frames despite phasing and some cases of occlusion. Figure 7 shows an example of optical flow tracking working robustly for a case of occlusion. However, as we will see next, some instances of occlusion may lead to failure cases.

Figure 7: Sample frames going chronologically from left to right demonstrating person tracking using optical flow. As seen, optical flow provides a reliable alternative for tracking human movement. Click here for a corresponding animation.

6.3.1 Optical Flow Failure Case

Here, we see a case where optical flow fails to account for occlusion. If Person B walks in front of Person A, the set of tracked points on Person A now lies on Person B. This means that optical flow will now start tracking Person B with the points from Person A. As a result, Person A's label is now assigned to Person B while Person A receives either a new label or Person B's old label. An example from our experiments is shown in Figure 8.

However, this does not occur all the time. If the occlusion were to occur during an update frame, as is the case in Figure 7, the object detection model would be unable to detect Person A behind Person B, which leads to our algorithm removing Person A's label and treating it as a case of phasing. As a result, once Person A comes back into view, the same label will be assigned back to them.

Figure 8: 3 frames demonstrating a failure case of optical flow tracking due to occlusion. In the leftmost frame, two people are labelled with IDs 7 and 43. In the middle frame, Person 43 walks in front of Person 7 and picks up Person 7's tracker. Once they separate in the rightmost frame, the IDs are now swapped. Click here for the corresponding animation.

6.4 Grouping

An example of grouping is shown in Figure 9, where black dots represent humans that are alone, while non-black dots of the same colour represent a single group. Shown in this Figure is a single group consisting of a couple walking upwards near the bench on the left. In the bird's-eye-view plot on the right, we can see a corresponding pair of green dots. To view more groups, an animation of all 1000 frames is available here.

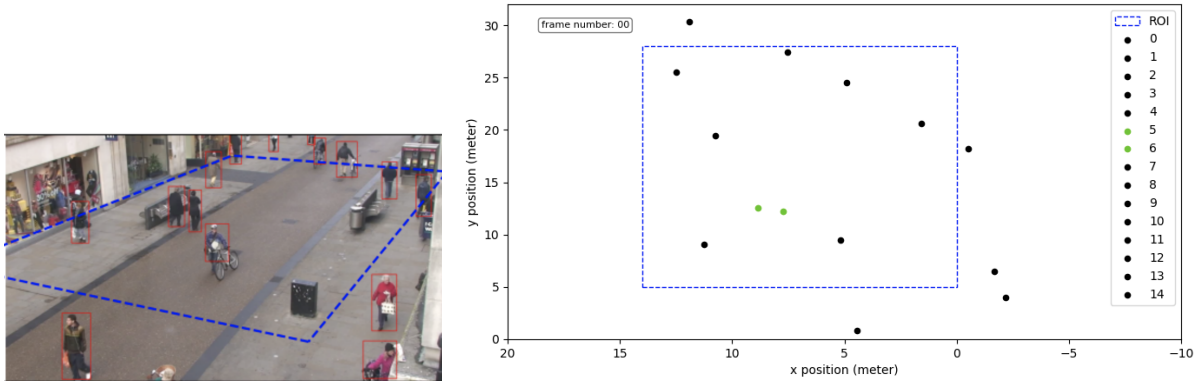


Figure 9: Example of grouping: the couple next to the bench on the left are indicated as a pair of green dots on the right

6.5 End to End Results

Figure 10 shows a frame from the final result of our pipeline for the Oxford dataset using centroid tracking with Faster-RCNN. This configuration is originally used by [1], so that the centre plot of Figure 10 shows the related work's replicated results. However, we see that without group detection, the couple in the frame is falsely classified as violating social distancing, which should not be the case. This is corrected by using our group detection method, with results shown in the rightmost plot.

By manually analyzing the first 250 frames, we find that [1]'s implementation detects a total of 418 violations, while our implementation detects a total of 156 violations. Of those violations, 262 of the 418 violations (62.68%) from [1] are false positives due to not accounting

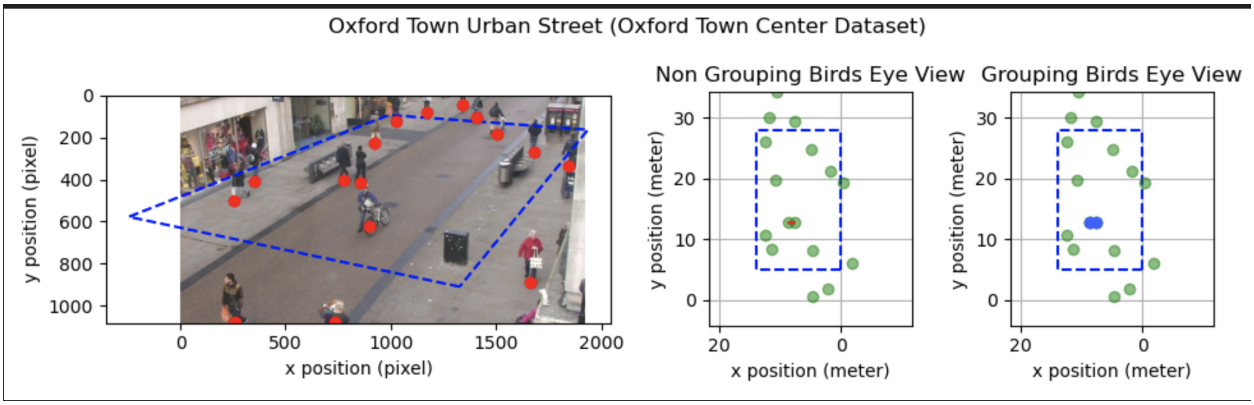


Figure 10: Single frame of social distancing detection on the Oxford dataset using Faster-RCNN with centroid tracking. (Left) A frame of the dataset where red dots are centred at the bottom of each person's bounding box. (Centre) Bird's-eye-view representation without group detection. Humans are represented by green dots and a red line is drawn if social distancing is violated. (Right) Bird's-eye-view representation with social group detection, where individuals are shown in green and groups are marked by non-green dots. Violations are again marked with red lines.

for groups or double counting. Specifically, of the 262 false violations, 249 of them are caused by the couple walking together next to the bench. On the other hand, our implementation does not double count violations nor detect violations between groups, resulting in a false positive rate of 0%. An animation of those 250 frames are available here.

Since Faster-RCNN is unable to be run in real time with our GPU setup, we conducted the same analysis for YOLOv3, which is more fit to be run in real time. The resulting false positive rate was 8.8% for our approach and 65% for [1]'s approach. Again, our method performs significantly better. Our non-zero false positive rate is due to YOLOv3 falsely detecting mannequins during some frames, resulting in a violation when a human would walk the storefront. Additional frames are available *here*.

Furthermore, the results of our experiment on the LSTN dataset using YOLOv3 with centroid tracking is shown in Figure 11. We can see 3 different groups, 2 of them having 3 members and another being a pair of people. We also see that violations across different groups are detected, but not within.

Experiments were also done using optical flow tracking with Mask-RCNN on the Oxford dataset. For optical flow, although we do account for grouping, it does not perform as well as centroid tracking. This is explained by Figures 12 and 13. Recall that with optical flow tracking, the object detection model is run at intervals to update the positions of people. This leads to a recalculation of points to track on each person, which can be at a different position on the body of the person. For example, in Figure 12, the rightmost plot shows the couple next to the bench as blue dots, indicating that they are in the same group. In Figure 13, however, the tracked features of the woman within the couple goes from her head to the bottom of her back. This makes the algorithm believe that she is a new individual that appeared out of nowhere, assigning her a new human ID. This results in the couple no longer being in the same group. In turn, we see a red violation line between them on the rightmost plot.

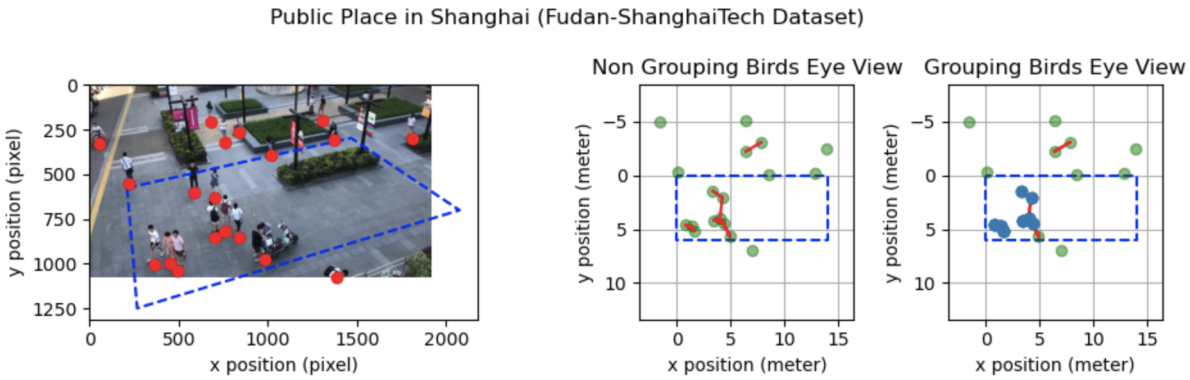


Figure 11: Single frame for performing social distancing detection for the Fudan dataset using YOLO human detection and centroid tracking. Notice our grouping algorithm can group groups of 2 and 3's, and do no detect violations within groups, only between groups.

Pipeline	Time(s)/100 Frames	FPS
YOLOv3+Centroid+Grouping	**5.20**	**19.2**
Faster-RCNN+Centroid+Grouping	17.2	5.81
Mask-RCNN+Optical Flow+Grouping	23.1	4.33

Table 3: End to end performance comparison of different object detection models, tracking methods, and grouping.

Finally, we consider the overall computational burden of implementing grouping onto the full pipeline. Here, we consider the Oxford dataset again, and showcase the performance for each model considered (YOLOv3, Faster-RCNN, Mask-RCNN) along with the performance for each tracking method (centroid, optical flow), and finally the performance penalty for finding and tracking groups. The results are presented in Table 3. Given that the Oxford and LSTN dataset are 25 FPS, we see that using YOLOv3 with centroid tracking and group detection performs nearly at real time at 19.2 FPS. In fact, with either a better GPU setup or by downsampling the frame rate, our method is able to run in real time. Additional frames are shown *here*.

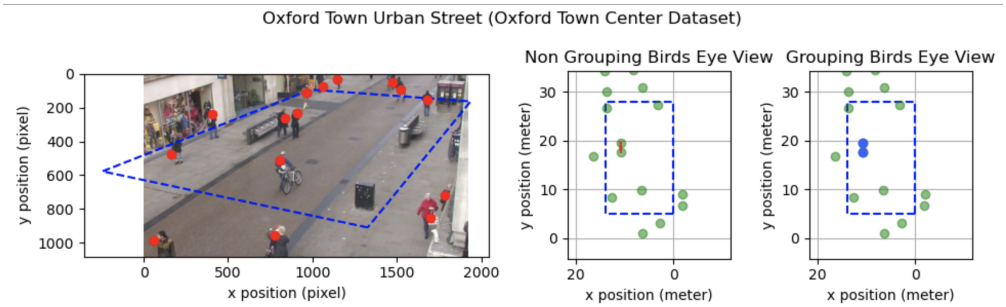


Figure 12: Single frame for the Oxford dataset using Mask-RCNN human detection and optical flow tracking. Notice that the centroid of the woman in the couple next to the bench is at her head. Also notice in the leftmost plot that the couple are considered a group, denoted by the blue dots.

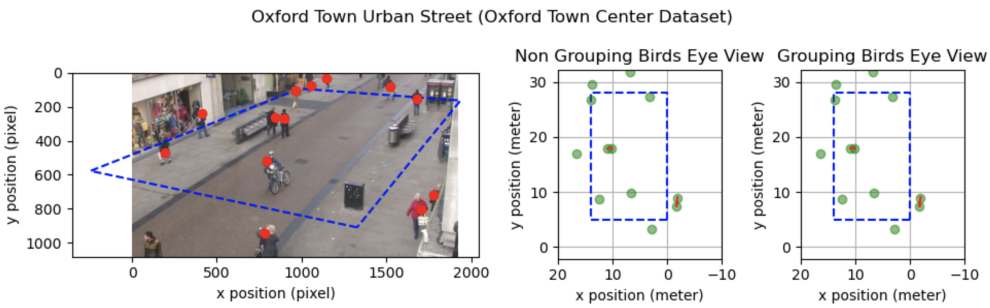


Figure 13: Same as Figure 12 but 1 frame later. Notice that the centroid of the woman in the couple next to the bench is now at the base of her back. This inconsistency results in a new human ID for her, which falsely causes a violation to be detected between the couple.

7 Conclusion

In conclusion, our work expands on previous social distancing detection work in multiple avenues. First, we account for individuals that are in the same social bubble such as families and couples to avoid a significant amount of false positives. Group detection is accomplished by two different human tracking methods, centroid tracking and optical flow tracking. Second, we expand on previous work by evaluating our results on 2 additional datasets and 2 additional human detection models. Overall, our results show that a significant portion (63% - 65%) of previous works' social distancing violations are false positives. On the other hand, our method results results in a false positive rate of 0% to 8.8%, a significant improvement. Our results were tested across a variety of datasets, object detection models, and tracking methods to ensure the generalizability of our methods. With a better setup, it can also be run in real time.

Author Contribution

Name	Contribution to Final Report	Contribution to Project
Gabriel Deza	• Contributed Equally (ie: 1/3)	• Centroid Tracking • Grouping
Danial Hasan	• Contributed Equally (ie: 1/3)	• Datasets • Homography • Object Detection (HOG, YOLO, Faster R-CNN)
Mengyu Yang	• Contributed Equally (ie: 1/3)	• Optical Flow • Mask R-CNN

References

[1] D. Yang, E. Yurtsever, V. Renganathan, K. A. Redmill, and Özgüner, "A Vision-based Social Distancing and Critical Density Detection System for COVID-19," *arXiv:2007.03578 [cs, eess]*, July 2020. arXiv: 2007.03578.

[2] "Landing AI Creates an AI Tool to Help Customers Monitor Social Distancing in the Workplace," Apr. 2020.

[3] J. Cao, Y. Pang, J. Xie, F. S. Khan, and L. Shao, "From handcrafted to deep features for pedestrian detection: A survey," *arXiv:2010.00456*, 2020.

[4] Z. Zou, Z. Shi, Y. Guo, and J. Ye, "Object detection in 20 years: A survey," 2019.

[5] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the seventh IEEE international conference on computer vision*, vol. 2, pp. 1150–1157, Ieee, 1999.

[6] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[7] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of computer and system sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[8] C. Papageorgiou and T. Poggio, "A trainable system for object detection," *International journal of computer vision*, vol. 38, no. 1, pp. 15–33, 2000.

[9] P. Viola, M. J. Jones, and D. Snow, "Detecting pedestrians using patterns of motion and appearance," *International Journal of Computer Vision*, vol. 63, no. 2, pp. 153–161, 2005.

[10] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, vol. 1, pp. 886–893, IEEE, 2005.

[11] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," 2016.

[12] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," 2016.

[13] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv*, 2018.

[14] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," *Lecture Notes in Computer Science*, p. 21–37, 2016.

[15] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," 2018.

[16] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," 2016.

[17] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," 2018.

[18] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," 2017.

[19] R. Girshick, "Fast r-cnn," 2015.

[20] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, pp. 303–338, June 2010.

[21] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, "Microsoft coco: Common objects in context," 2015.

[22] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of the 7th international joint conference on Artificial intelligence - Volume 2*, IJCAI'81, (San Francisco, CA, USA), pp. 674–679, Morgan Kaufmann Publishers Inc., Aug. 1981.

[23] B. Benfold and I. Reid, "Stable multi-target tracking in real-time surveillance video," *CVPR 2011*, pp. 3457–3464, 2011.

[24] A. B. Chan and N. Vasconcelos, "Modeling, clustering, and segmenting video with mixtures of dynamic textures," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 5, pp. 909–926, 2008.

[25] Y. Fang, B. Zhan, W. Cai, S. Gao, and B. Hu, "Locality-constrained spatial transformer network for video crowd counting," *arXiv preprint arXiv:1907.07911*, 2019.

[26] E. Lindernoren, "Pytorch-yolov3." `https://github.com/eriklindernoren/PyTorch-YOLOv3`, 2017.