Auto-SDA: Automated Video-based Social Distancing Analyzer

Mahshid Ghasemi, Zoran Kostic, Javad Ghaderi, Gil Zussman
Electrical Engineering, Columbia University
New York, NY
{mahshid.ghasemi,zk2172,jghaderi,gil.zussman}@columbia.edu

Abstract

Social distancing can reduce infection rates in respiratory pandemics such as COVID-19, especially in dense urban areas. To assess pedestrians' compliance with social distancing policies, we use the pilot site of the PAWR COSMOS wireless edge-cloud testbed in New York City to design and evaluate an **Auto**mated video-based **S**ocial **D**istancing **A**nalyzer (**Auto-SDA**) pipeline. Auto-SDA derives pedestrians' trajectories and measures the duration of close proximity events. It relies on an object detector and a tracker, however, to achieve highly accurate social distancing analysis, we design and incorporate 3 modules into Auto-SDA: (i) a *calibration module* that converts 2D pixel distances to 3D on-ground distances with *less than 10 cm* error, (ii) a *correction module* that identifies pedestrians who were missed or assigned duplicate IDs by the object detection-tracker and rectifies their IDs, and (iii) a *group detection module* that identifies affiliated pedestrians (i.e., pedestrians who walk together as a social group) and excludes them from the social distancing violation analysis. We applied Auto-SDA to videos recorded at the COSMOS pilot site before the pandemic, soon after the lockdown, and after the vaccines became broadly available, and analyzed the impacts of the social distancing protocols on pedestrians' behaviors and their evolution. For example, the analysis shows that after the lockdown, less than 55% of the pedestrians violated the social distancing protocols, whereas this percentage increased to 65% after the vaccines became available. Moreover, after the lockdown, 0-20% of the pedestrians were affiliated with a social group, compared to 10-45% once the vaccines became available. Finally, following the lockdown, the density of the pedestrians at the intersection decreased by almost 50%.

CCS Concepts

• **Computing methodologies** → **Object detection**; **Activity recognition and understanding**; **Tracking**.

Keywords

Social distancing, COVID-19, Object detection, Tracking, Smart city

ACM Reference Format:
Mahshid Ghasemi, Zoran Kostic, Javad Ghaderi, Gil Zussman. 2022. Auto-SDA: Automated Video-based Social Distancing Analyzer. In *3rd ACM Workshop on Hot Topics in Video Analytics and Intelligent Edges (HotEdgeVideo'21), January 31-February 4 2022, New Orleans, LA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3477083.3480154

HotEdgeVideo'21, January 31-February 4 2022, New Orleans, LA, USA

ACM ISBN 978-1-4503-8700-2/22/01…$15.00
https://doi.org/10.1145/3477083.3480154

Figure 1: The NSF PAWR COSMOS pilot site at 120th St. and Amsterdam Ave. intersection, NYC. Cameras are deployed on the Columbia University's Mudd building and connected to the edge-cloud servers via dedicated fibers.

Figure 2: Different stages in the Auto-SDA pipeline.

1 Introduction

Social distancing is one of the primary tools to reduce the transmission of the SARS-CoV-2 virus that causes COVID-19. A way to obtain information about pedestrian density and behavior, which can be critical in assessing and controlling respiratory pandemics such as COVID-19, is through smart-city infrastructures. In this paper, we use the NSF PAWR COSMOS wireless edge-cloud testbed, which is being deployed in West Harlem, New York City (NYC) [18, 25], to develop a video-based object detection, tracking, and density estimation pipeline, that can provide such information. Specifically, we use one of the cameras in the testbed's pilot site (see Fig. 1) to design and evaluate a fully **Auto**mated video-based **S**ocial **D**istancing **A**nalyzer (**Auto-SDA**) pipeline (shown in Fig. 2). This pipeline measures the distance between unaffiliated pedestrians (i.e., the pedestrians who do not walk together as a social group) and assesses if they maintain 6 ft distance.

The main goal of this paper is to design a highly accurate social distancing analyzer pipeline, whose performance is *not* sensitive to the camera's viewpoint and scene dynamics. As illustrated in Fig. 2, the pipeline includes an object detector model (YOLOv4 [4]) and a tracker model (Nvidia DCF-based tracker) that extracts the trajectory of pedestrians. These trajectories are eventually used to compute the proximity duration of each two unaffiliated pedestrians separately. While these are off-the-shelf components, achieving accurate social distancing analysis calls for the design of tailored components. Specifically, we achieve this goal by incorporating three modules in Auto-SDA, as outlined below and in Section 3:

Table 1: A comparison of prior work to Auto-SDA

Framework	Object Detection	Tracking	Calibration Method	On-Ground Distance Computation Error	Correction	Group Detection	Real-World COVID-19 Pandemic Impact Analysis
[24]	✓	X	Homography trans.	$\gg$ 10 cm	X	X	X
[17]	✓	✓	Depth information	$\gg$ 10 cm	X	X	X
[2, 3, 6, 7]	✓	X	Planar camera persp. trans.	$\gg$ 10 cm	X	X	X
Auto-SDA	✓	✓	Multi-area calibration	$<$ 10 cm	✓	✓	✓

- **Camera calibration module:** Our measurements show that using a single set of photogrammetry parameters for the whole scene leads to imprecise on-ground distance computation. Therefore, this module breaks the view of the camera into multiple areas and computes the corresponding photogrammetry parameters for each area individually. These parameters are then used to convert the 2D on-image distances into 3D on-ground distances with less than 10 cm error.
- **ID correction module:** It compensates for the inaccuracies of the object detector and tracking model caused by the camera's tilt angle and the obstacles on the road. For instance, if multiple IDs are assigned to a single pedestrian, this module removes the redundant IDs and derives the entire trajectory of that pedestrian.
- **Group detection module:** This module detects the pedestrians affiliated with a single social group (e.g., members of a family) and excludes them from social distancing violation.

We applied Auto-SDA to our dataset collected by a camera located on the 2nd floor of Columbia's Mudd building (see Fig. 1)[1]. The dataset consists of 180 sec videos recorded at different times of the day (9 AM, 2 PM, 5:30 PM, 7:30 PM, and 10 PM) in about one month periods, soon after the lockdown (June 17 to July 20, 2020), and after the vaccines became broadly available (May 2021). In addition, the dataset includes 16 videos collected (less methodically) before the pandemic (June 2019) which are used as a reference point[2]. The detailed results (described in Section 4) show, for example, that after the lockdown, the density of pedestrians seen at the intersection decreased by almost 50%. Moreover, after the lockdown, less than 55% of the pedestrians violated the social distancing protocols compared to 65% post-vaccine. The results also show that the fraction of pedestrians walking as a social group has grown from 0-20% (after the lockdown) to 10-45% (post-vaccine). We discuss potential extensions and future work in Section 5.

2 Related Work

Auto-SDA uses an object detector and a tracker. Several detectors are available, including Mask R-CNN [9], SSD [12], YOLOv3 [19], and (YOLOv4) [4]. Auto-SDA uses the state-of-the-art object detector YOLOv4, which provides adequate speed and accuracy for social distancing analysis. Auto-SDA also uses the NVIDIA DCF tracker, which based on our experiments provides higher accuracy than other trackers, such as DeepSORT [23].

We now review related work that focuses on monitoring the COVID-19 pandemic and compare Auto-SDA to social distancing frameworks (see Table 1). A survey on enabling wireless technologies for monitoring social distancing appears in [14]. The work [8] explores using smart city technologies in pandemic management. The work [24] proposes the use of monocular cameras and deep learning-based object detectors to monitor social distancing and emit warnings, however since it does not use a tracker, it can only provide instantaneous warnings. Moreover, [24] uses homography transformation to convert 2D on-image coordinates to their 3D counterparts, which can only be used to estimate the camera pose for planar objects. Thus, a more advanced method is required to calibrate the cameras and compute the on-ground distances from the pixel distances on an image. The framework in [17] uses YOLOv3 for object detection and DeepSORT for tracking. The obtained bounding boxes are utilized to obtain depth information of the pedestrians (i.e., their distance from the camera lens) and identify clusters of pedestrians violating social distancing. However, the depth information-based method is not sufficiently accurate for measuring the distance between pedestrians, and a more precise camera calibration along with group detection is needed.

The frameworks in [2, 3, 6, 7] employ an object detector, but do not use a tracker to derive trajectories. Moreover, they perform planar camera perspective transformation for calibration, which yields an inaccurate estimation of the on-ground coordinates, thereby limiting the social distancing measurements accuracy. Finally, [2, 3, 6, 7, 17, 24] do not provide evaluations on real-world data recorded during the COVID-19 pandemic. Table 1 summarizes the main features of Auto-SDA compared to the prior work.

3 Pipeline Modules

3.1 Camera Calibration

Camera calibration is a necessary step for extracting on-ground distances between pedestrians. The goal is to determine the intrinsic and extrinsic parameters of the camera to convert the 2D on-image coordinates, viewed by the camera, to the 3D on-ground coordinates. Intrinsic parameters are (i) principal point (c_x, c_y), (ii) focal length in pixel units (f_x, f_y), (iii) radial distortion coefficients $(k_1, k_2, ..., k_6)$, and (iv) tangential distortion coefficients (p_1, p_2). Extrinsic parameters are (i) rotation matrix R, and (ii) translation vector t.[3]

Since the testbed cameras are fixed, we needed to calibrate them once. As part of this process, we captured multiple photos of a checkerboard with known square sizes, posed in different tilt and rotation angles (see Fig. 3). Then, we fed the 2D on-image pixel coordinates of the checkerboard corners and their corresponding 3D coordinates into OpenCV [5], that runs the global Levenberg-Marquardt optimization algorithm, to calculate the required parameters.

Moreover, we split the view of the intersection into 10 areas (as shown in Fig. 4) and for each area, we determined the extrinsic parameters individually. This can further mitigate the impact of

[1]The use of the videos by Columbia researchers is IRB-exempt. The videos are solely used for research-related purposes, and they will not be shared in any way.

[2]You can find a sample video in https://bit.ly/2Rt36S2.

[3]The effects of higher order coefficients are negligible, see [22].

Table 2: A comparison of calibration methods used in the prior work to Auto-SDA's multi-area calibration

Pixel Coordinates of a Pair of Points on a 4 K Frame	On-Ground Distance (cm)	Distance Calculated by Multi-area Calibration (cm)	Distance Calculated by Homography Trans. [24] (cm)	Distance Calculated by Planar Camera Persp. Trans. [2, 3, 6, 7] (cm)
[1093, 715], [1065, 685]	320	325	209	339
[1785, 572], [1862, 566]	183	178	140	128
[1680, 582], [1588, 552]	503	508	368	457
[2153, 598], [2077, 582]	259	256	201	146
[1121, 746], [1093, 714]	320	314	201	229

camera distortion and obtain the on-ground distances with less than 10 cm error (ground-truth obtained from actual distance measurements on the COSMOS pilot site). The number of areas can increase to improve the accuracy but for our use-case, using 10 areas proved to be adequate. For each area, we determined a few points on the ground with known coordinates (w.r.t. a predefined center point which should be on the edge of the area) and found their corresponding pixel coordinates from the perspective of the camera. These sample points along with the intrinsic parameters of the camera are then used to determine the extrinsic parameters (using OpenCV).

Auto-SDA plugs these parameters into the photogrammetry equations [1, 5, 11], given below, and completes the 2D-3D coordinates conversion:

$$\begin{bmatrix} x & y & z \end{bmatrix}^T = \boldsymbol{R} \begin{bmatrix} X & Y & Z \end{bmatrix}^T + t, \quad x' = \frac{x}{z}, \quad y' = \frac{y}{z}$$

$$x'' = x' \frac{1 + k_1 r^2 + k_2 r^4 + k_3 r^6}{1 + k_4 r^2 + k_5 r^4 + k_6 r^6} + 2p_1 x' y' + p_2 (r^2 + 2x'^2)$$

$$y'' = y' \frac{1 + k_1 r^2 + k_2 r^4 + k_3 r^6}{1 + k_4 r^2 + k_5 r^4 + k_6 r^6} + p_1 (r^2 + 2y'^2) + 2p_2 x' y'$$

$$r^2 = x'^2 + y'^2, \quad u = f_x x'' + c_x, \quad v = f_y y'' + c_y.$$

In the equations above, $[u, v]$ are the 2D pixel coordinates and $[X, Y, Z]$ are the 3D on-ground coordinates. Since there are no closed-form equations to map the 2D points to 3D points, Auto-SDA uses Newton's method to solve the above system of equations (it sets the ground level to $Z = 0$ and solves for X and Y).

In Table 2, we compare the accuracy of on-ground distance calculation of the multi-area calibration method used in Auto-SDA with the calibration methods used in [2, 3, 6, 7, 24]. As the results show, there could be more than 1 m error in calculating the on-ground distances when using the homography and planar camera perspective transformation method, used in previous work. While it may be sufficient for other applications, such an error is inadequate for social distancing monitoring. Moreover, in [17] the distance between pedestrians is determined by using a method proposed in [16]. In this method, first, the distance of a pedestrian from the camera lens is obtained using the coordinates, width, and height of its bounding box provided by an object detector. Then, the distance between every two pedestrians is calculated. In Fig. 5 we represent the results of calculating the pedestrians distances from the camera lens using the calibration method proposed in [16]. The camera's (vertical and horizontal) distance from the pedestrians is more than 10 m. However, due to the oblique view of the camera, the calculated distances (displayed near the bounding boxes) are inaccurate, and one cannot simply fix them (e.g., using scaling factor).

3.2 Pedestrians Detection and Tracking

Auto-SDA uses the YOLOv4 object detector [4] to detect the pedestrians. It is also equipped with a tracker (NvDCF) that extracts the trajectory of each pedestrian and uses that to trace the number of pedestrians they are in contact with (within a radius of 6 ft) and the duration of each contact. Both models are set as building blocks inside the Deepstream pipeline which is an optimized architecture built using the Gstreamer framework [15].

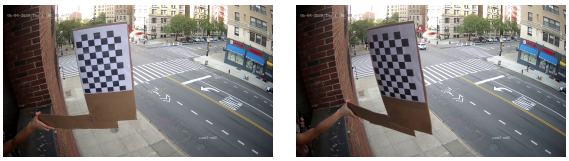

Figure 3: Calibration of the COSMOS cameras using a checkerboard: more than 20 images of the checkerboard in different poses were provided to the OpenCV library to obtain the intrinsic parameters of the camera.

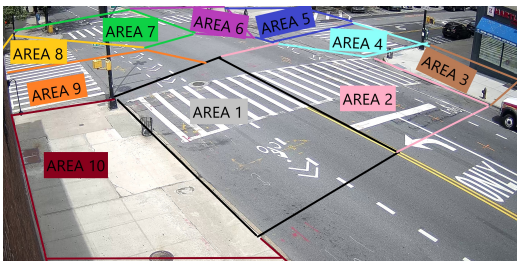

Figure 4: Division of the camera scene into 10 areas. The extrinsic parameters of the camera were calculated for each area individually.

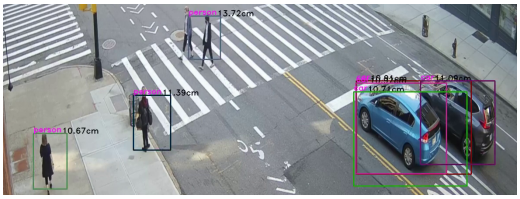

Figure 5: Computed distances of pedestrians from the camera using objects' depth information proposed in [16]. The real distances (both vertically and horizontally) of the pedestrians from the camera is more than 10 m. However, due to the oblique view of the camera, the obtained distances deviate from their real values and it is not straightforward to rectify them (e.g., using a scaling factor.)

3.3 ID Correction

The COSMOS cameras are located at relatively high altitudes and have an oblique view of the intersection. Therefore, the pedestrians are small and might be blocked by the obstacles such as vehicles, traffic lights, and other pedestrians. As a result, the object detector and tracker have degraded performance (i.e., it is likely that the

Algorithm 1 ID Correction

Input: $ID_{\text{vec}}, e_1, e_2, n$ ▷ ID_{vec} is the output of NvDCF tracker
Output: corrected ID_{vec}
for $id \in ID_{\text{vec}}$ **do**
  Compute $id.Trj$ ▷ vector of points on id's path
  Compute $id.TimeStamp.StartTime$ ▷ detection time
  Compute $id.TimeStamp.StopTime$ ▷ Lost time
  Compute $(id.TailEst, id.TailDir)$ ▷ Linear Regression of $id.Trj.tail(n)$
  Compute $(id.HeadEst, id.HeadDir)$ ▷ Linear Regression of $id.Trj.head(n)$
for $(id_1, id_2) \in ID_{\text{vec}}$ **do**
  $t1 \leftarrow id_1.TimeStamp.StopTime$
  $t2 \leftarrow id_2.TimeStamp.StartTime$
  $p_1 \leftarrow id_1.TailEst.\text{at}(t = t_2)$, $p_2 \leftarrow id_1.Trj.\text{at}(t_2)$
  $v_1 \leftarrow id_1.TailDir$, $v_2 \leftarrow id_2.HeadDir$
  if $t_2 - t_1 < e_1$ && $|p_1 - p_2| < e_2$ && $\angle(v1, v2) < 90°$ **then**
    id_1 and id_2 belongs to same person

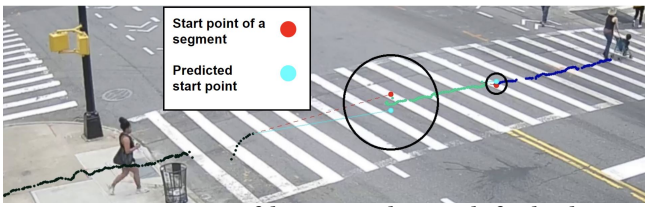

Figure 6: Demonstration of detection and removal of redundant IDs by the ID Correction algorithm when the tracker assigns 3 IDs to a single pedestrian.

tracker loses a pedestrian along the way or assigns multiple IDs to a single person).

The ID Correction module is designed to mitigate this. It detects the IDs that belong to a single pedestrian and extracts their entire trajectory. Algorithm 1 describes our ID Correction algorithm. It receives the results of the object detector and tracking module as its input, and, for each ID, it creates a structure in which it keeps the trajectory ($id.Trj$), the first and last time it was detected by the tracker ($id.TimeStamp$), and the parameters of the Linear Regression approximation of the tail and head of the trajectory. The algorithm then uses this information to predict the pedestrian's location before it was detected and after it was lost.

For each ID pair (id_1, id_2), the ID Correction algorithm then verifies three conditions to determine whether they are associated with a single pedestrian or not. First, the gap between id_1 lost time (t_1) and id_2 detected time (t_2) must be small enough (less than a predefined threshold e_1). Second, the distance between predicted location of id_1, at the time that id_2 was first detected (t_2) using the Linear Regression approximation for the tale of id_1 trajectory, and the location of id_2 at that time (t_2) has to be less than a specified threshold (e_2). Third, it measures the angle between id_1's tail direction and id_2 head direction. This angle must be less than 90° to ensure that the algorithm does not consider two pedestrians crossing each other in opposite direction as a single pedestrian. If all three conditions hold, then id_1 and id_2 belong to a single person. An example is shown in Fig. 6 where the tracker assigned 3 IDs to a single pedestrian. The ID correction module detects the segments that belong to a single trajectory by using the Linear Regression approximation corresponding to the tail of each segment and comparing the estimated start point and the real start point of the subsequent segment.

3.4 Group Detection

We enhance the social distancing analysis by distinguishing the pedestrians walking together as a social group (e.g., friends/family)

Algorithm 2 Group Detection

Input: $ID_{\text{vec}}, d_{\max}, d_{\max}, \sigma_{\max}$
Output: ID_{vec} Pdestrians belong to a group
for $id \in ID_{\text{vec}}$ **do**
  $id.TimeTrj = \text{map}(id.TimeStepVec, id.Trj)$
for $(id_1, id_2) \in ID_{\text{vec}}$ **do**
  $n = 0$
  for $t = 1 : T$ **do**
    $pos_1 = id_1.TimeTrj(t)$, $pos_2 = id_2.TimeTrj(t)$
    $d = ||pos_1 - pos_2||_2$
    if $d > d_{\max}$ **then**
      $n++$, continue
    $Corr_{\text{vec}}(id_1, id_2).\text{append}(d)$
  if $n > N_{\max}$ **then**
    continue
  $\bar{d} = \text{mean}(Corr_{\text{vec}}(id_1, id_2))$ ▷ calculate the mean distance between two pedestrians
  $\sigma = \text{std}(Corr_{\text{vec}}(id_1, id_2))$ ▷ calculate the standard deviation of instantaneous distances between two pedestrians
  if $\bar{d} < \bar{d}_{\max}$ && $\sigma < \sigma_{\max}$ **then**
    id_1 and id_2 belongs to the same group

and excluding them from social distancing violation. There are several methods proposed for group-detection, e.g., see [10, 13, 20, 21]. All these group-detection methods require details such as velocity, body and head orientation, and exact trajectory. However, in our setting (and in many realistic deployments), the cameras are mounted on a relatively high altitude, viewing the intersection from a corner with a large tilt angle. Moreover, there are various obstacles on the road that might block the view of pedestrians for some periods. Therefore, that kind of detailed information cannot be obtained from these cameras.

We designed a group detection algorithm that can detect pedestrians that belong to a single social group with the limited data that we can derive from cameras such as the ones in the COSMOS pilot site. The Group Detection algorithm is given in Algorithm 2. It uses IDs of the pedestrians rectified in the ID Correction module to derive an approximation of each pedestrian trajectory. Then, it calculates the correlation between these trajectories to check if two pedestrians belong to a single social group. Specifically, the algorithm calculates the distance between each pair of pedestrians (id_1, id_2) on all the frames and then calculates the average distance ($\bar{d}$) and empirical standard deviation (σ). Two pedestrians are labeled as one social group, if their instantaneous distance (d) does not exceed $d_{\max}$ in more than $N_{\max}$ frames, and the mean and standard deviation of their distance are less than $\bar{d}_{\max}$ and $\sigma_{\max}$, respectively. To evaluate the performance of the group detection algorithm and fine-tune its parameters we applied the algorithm on sample videos recorded from the COSMOS pilot site and compared the results against the visually detected social groups.

4 Measurements and Evaluation

After obtaining the real distance between pedestrians, their IDs, and their trajectories, Auto-SDA computes how often an individual breaks the social distancing policies and how long this violation lasts. We applied Auto-SDA to videos recorded from a camera deployed on the 2$^{\text{nd}}$ floor of the Columbia University's Mudd building at the COSMOS pilot site. The camera is configured to record 180 sec (which is two times the signal timing cycle of the traffic lights at the intersection) videos, 5 times a day at 9 AM, 2 PM, 5:30 PM, 7:30 PM, and 10 PM. We deployed Auto-SDA in one of the COSMOS edge servers and applied it on videos recorded between June 17 and July 20, 2020 (after the lockdown), and during May 2021 (after the vaccines became broadly available). We also used 16 sample videos

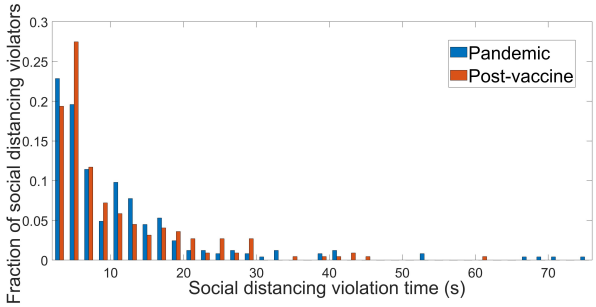

Figure 7: Normalized histogram of the duration of the detected social distancing violation incidents.

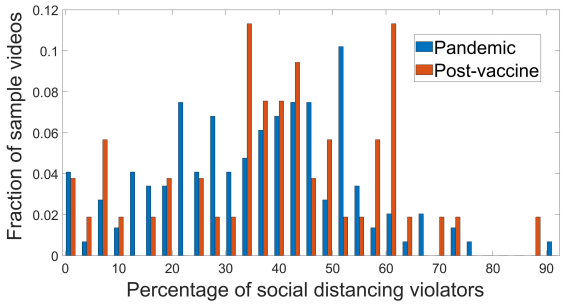

Figure 8: Normalized histogram of the percentage of pedestrians considered social distancing violators in the recorded videos.

recorded before the COVID-19 outbreak (in June 2019) to evaluate the impact of the pandemic on pedestrians' density. Below, we provide the analysis results (results corresponding to June-July, 2020 and May 2021 are labeled as *Pandemic* and *Post-vaccine*, respectively).

Fig. 7 compares the duration of social distancing violation incidents during the pandemic and post-vaccine. As demonstrated, a growth (of around 3 s) is observed post-vaccine. For each video, we calculated the percentage of pedestrians who violate social distancing and plot a normalized histogram of the results in Fig. 8. It can be seen that, after the lockdown, typically, less than 55% of the pedestrians violated social distancing, compared to 65% post-vaccine. Fig. 9 illustrates the normalized histogram of the number of pedestrians who violate social distancing at different times of the day. Fig. 10 shows the fraction of recorded videos in which a certain percentage of pedestrians are walking as a group. One can see that the fraction of pedestrians walking as a social group has grown from 0-20% (after the lockdown) to 10-45% (post-vaccine). Fig. 11 shows the increase in the *maximum* duration of post-vaccine social distancing violation incidents.

We compare the pre-pandemic, lockdown, and post-vaccine density of the crowd at the intersection in Fig. 12. One can observe that density of the pedestrians has decreased by almost 50% after the lockdown (compared to pre-pandemic), while it has slightly increased recently.

5 Conclusions

We presented the Auto-SDA pipeline that evaluates if unaffiliated pedestrians comply with the social distancing policies. It is equipped with a group detection module and is capable of calculating the

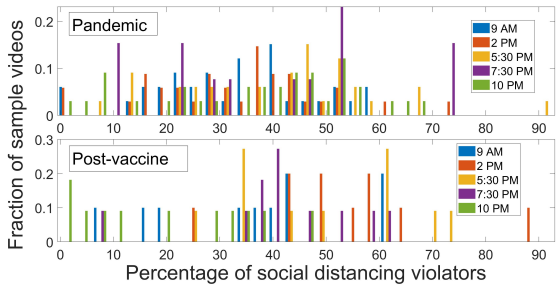

Figure 9: Normalized histogram of the number of pedestrians violating social distancing protocols in different times of the day.

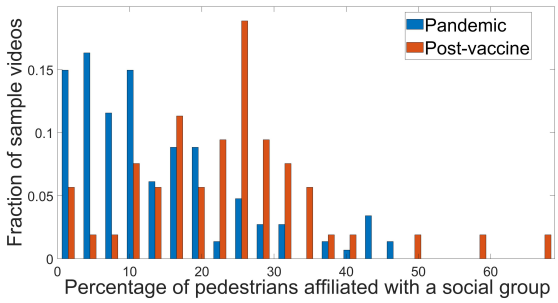

Figure 10: Normalized histogram of the percentage of pedestrians affiliated with a social group.

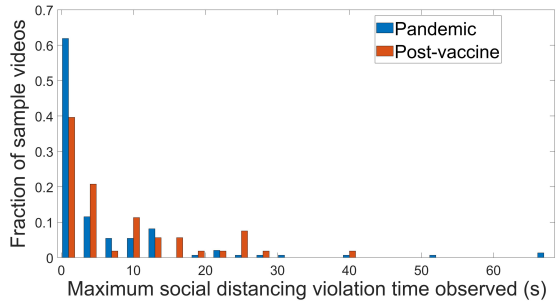

Figure 11: Normalized histogram of the maximum duration of social distancing violation observed.

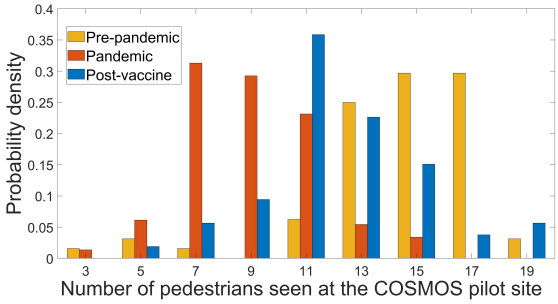

Figure 12: Comparison between the density of pedestrians walking at the COSMOS pilot site in different periods.

on-ground distance between pedestrians with less than 10 cm error. We applied Auto-SDA to the videos recorded by a camera deployed at the COSMOS pilot site. The results demonstrate the impact of the COVID pandemic on pedestrians' behaviors. This work is a first step towards designing systems for evaluating compliance with social distancing and for density assessment. Future work will include the design of privacy preserving methods, integration of information from multiple cameras and sensors, design of real time algorithms, and extensive evaluation as the social distancing policies change.

Acknowledgments

We thank Ivan Seskar, Jakub Kolodziejski, and Michael Sherman (Rutgers/WINLAB). This work was supported in part by NSF grants CNS-1827923, OAC-2029295, NSF-BSF grant CNS-1910757, ARO grant W911NF1910, and AT&T VURI award.

References

[1] Peyman Alizadeh. 2015. *Object distance measurement using a single camera for robotic applications.* Ph.D. Dissertation. Laurentian University of Sudbury.

[2] John Betancourt. 2020. Social Distancing Analyser. https://github.com/JohnBetaCode/Social-Distancing-Analyser.

[3] Deepak Birla. 2020. Social Distancing AI. https://github.com/deepak112/Social-Distancing-AI.

[4] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. 2020. YOLOv4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934* (2020).

[5] Gary Bradski. 2000. The OpenCV Library. *Dr. Dobb's Journal of Software Tools* (2000).

[6] Marco Cristani, Alessio Del Bue, Vittorio Murino, Francesco Setti, and Alessandro Vinciarelli. 2020. The visual social distancing problem. *arXiv preprint arXiv:2005.04813* (2020).

[7] Tom Farrand. 2020. Social Distancing. https://github.com/FarrandTom/social-distancing.

[8] Maanak Gupta, Mahmoud Abdelsalam, and Sudip Mittal. 2020. Enabling and enforcing social distancing measures using smart city and its infrastructures: a COVID-19 Use case. *arXiv preprint arXiv:2004.09246* (2020).

[9] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. 2017. Mask R-CNN. In *Proc. IEEE Int. Conf. Computer Vision.*

[10] Shoichi Inaba and Yoshimitsu Aoki. 2016. Conversational group detection based on social context using graph clustering algorithm. In *2016 12th Int. Conf. Signal-Image Technol. Internet-Based Sys. (SITIS).*

[11] Matthias Jüngel, Heinrich Mellmann, and Michael Spranger. 2007. Improving vision-based distance measurements using reference objects. In *Robot Soccer World Cup.*

[12] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. 2016. SSD: Single shot multibox detector. In *Proc. Eur. Conf. Computer Vision.*

[13] Riccardo Mazzon, Fabio Poiesi, and Andrea Cavallaro. 2013. Detection and tracking of groups in crowd. In *2013 10th IEEE Int. Conf. Adv. Video Signal Based Surveill.*

[14] Cong T Nguyen, Yuris Mulya Saputra, Nguyen Van Huynh, Ngoc-Tan Nguyen, Tran Viet Khoa, Bui Minh Tuan, Diep N Nguyen, Dinh Thai Hoang, Thang X Vu, Eryk Dutkiewicz, et al. 2020. Enabling and Emerging Technologies for Social Distancing: A Comprehensive Survey. *arXiv preprint arXiv:2005.02816* (2020).

[15] John Nickolls, Ian Buck, Michael Garland, and Kevin Skadron. 2008. Scalable parallel programming with CUDA. *Queue* 6, 2 (2008), 40–53.

[16] Paul Pias. 2020. Object detection and distance measurement. https://github.com/paul-pias/Object-Detection-and-Distance-Measurement.

[17] Narinder Singh Punn, Sanjay Kumar Sonbhadra, and Sonali Agarwal. 2020. Monitoring COVID-19 social distancing with person detection and tracking via fine-tuned YOLOv3 and DeepSort techniques. *arXiv preprint arXiv:2005.01385* (2020).

[18] Dipankar Raychaudhuri, Ivan Seskar, Gil Zussman, Thanasis Korakis, Dan Kilper, Tingjun Chen, Jakub Kolodziejski, Michael Sherman, Zoran Kostic, Xiaoxiong Gu, et al. 2020. Challenge: COSMOS: A city-scale programmable testbed for experimentation with advanced wireless. In *Proc. ACM MobiCom'20.*

[19] Joseph Redmon and Ali Farhadi. 2018. YOLOv3: An incremental improvement. *arXiv preprint arXiv:1804.02767* (2018).

[20] Francesco Solera, Simone Calderara, and Rita Cucchiara. 2015. Socially constrained structural learning for groups detection in crowd. *IEEE Trans. Pattern Anal. Mach. Intell.* 38, 5 (2015), 995–1008.

[21] Sebastiano Vascon and Loris Bazzani. 2017. Group detection and tracking using sociological features. In *Group and crowd behavior for computer vision.* Elsevier, 29–66.

[22] Yaming Wang, Y Li, and JB Zheng. 2010. A camera calibration technique based on OpenCV. In *Proc. 3rd Int. Conf. Inf. Sci. Interact. Sci.*

[23] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. 2017. Simple online and realtime tracking with a deep association metric. In *Proc. IEEE ICIP'17.*

[24] Dongfang Yang, Ekim Yurtsever, Vishnu Renganathan, Keith A Redmill, and Ümit Özgüner. 2020. A vision-based social distancing and critical density detection system for COVID-19. *arXiv preprint arXiv:2007.03578* (2020), 24–25.

[25] Shiyun Yang, Emily Bailey, Zhengye Yang, Jonatan Ostrometzky, Gil Zussman, Ivan Seskar, and Zoran Kostic. 2020. COSMOS smart intersection: Edge compute and communications for bird's eye object tracking. In *Proc. 4th Int. Workshop Smart Edge Comput. Netw. (SmartEdge'20).*