Crowd Scene Anomaly Detection in Online Videos

Kaizhi Yang, Alper Yilmaz

Photogrammetric Computer Vision Lab, The Ohio State University, Bolz Hall Suite 233, 2036 Neil Ave, Columbus, OH 43210, United States – yang.6196@osu.edu, yilmaz.15@osu.edu

Keywords: Object Detection, Anomaly Detection, CNN, Crowd surveillance, Geometric Rectification.

Abstract

The prevalence of surveillance cameras in public places has led to an extremely pressing need for effective position and crowd monitoring, as well as anomaly detection. This paper tends to exhibit an incorporated approach that combines state-of-the-art computer vision techniques for comprehensive crowd surveillance. The main features of our approach are summarized into four steps: (a) Object detection and tracking; (b) Geometric rectification for positioning; (c) Motion extraction; and (d) Anomaly detection. First, this uses YOLOv5's Convolutional Neural Network (CNN) model in making efficient detection of objects, focusing on spotting individuals within crowded scenes. After detection, a strong mechanism for tracking is established with the help of the DeepSORT algorithm, which can track the person across frames. It must gain the people's position in the video frame and analyze motion data with the guarantee of capture of camera-scene geometry. Each frame thus gets converted from the 3D perspective to a 2D bird's eye view within the surveillance video, giving a guarantee of capture of the geometry of a camera scene. Motion anomaly detection is addressed through statistical methods, with Kernel Density Estimation (KDE) being employed to identify deviations from normal motion patterns. Extensive experiments conducted on different online crowd scene video datasets validate the effectiveness of the proposed anomaly detection mechanism. Overall, this integrated approach proposes a promising solution to crowd surveillance, further development of object detection, tracking, and anomaly analysis for monitoring public spaces.

1. Introduction

The surveillance camera was widely used to monitor human’s behavior in the public spaces such as the shopping mall, football game event, and airport (Arroyo et al., 2015). Ensuring the safety and security of public spaces depends on the ability to identify unexpected movements, irregular patterns, and anomalies (Klauser, Ruegg, November, 2008). There is no standard definition of anomaly. The anomaly in our context refers to the motion data patterns of individuals that do not follow the main flow of the data patterns, which is also called local anomaly, as well as any means of transportation that people use to move. For example, most of the people are walking across the street from left to right while there are minor amount of people walking though from right to left. The individuals who were walking against the main flow were defined as anomalies. Besides, the anomalies are recognized not only by the human themselves. If a person is riding a bike or driving a car, he is moving with different speed and direction. This person can also be defined as an anomaly. Although there were decent amount of work conquering this challenge, the anomaly detection task was still a valuable and significant research field. With the dramatic growth of the installation of surveillance cameras in public spaces, the volume of video frame data being generated is increasing rapidly (Nayak, Pati, and Das, 2021). A efficient behavior monitoring system is pivoted to handle this large amount of information. The paper proposed an integrated approach that combines the state-of-the-art YOLOv5 object detection model, DeepSORT algorithm, geometric rectification technique, and KDE method to detect and track anomaly efficiently in video streams.

2. Related Work

The purpose of this section is to review the previous work on anomaly detection. The related studies are categorized into three areas: Supervised Learning, Unsupervised Learning, and Semi-supervised Learning. This categorization was bult on the participation of human intervention which is the label usage during the training of the model (Mohindru and Singla, 2021).

The supervised anomaly detection approach is utilizing the presence of label for identifying the normal and abnormal activity. During the training phase of the model, datasets with labels were used. Each frame in the datasets contained one or multiple annotated objects with either normal or abnormal label attached to the object. The model was trained to detect the abnormal object (Mohindru and Singla, 2021). One of the very common uses is the use of Deep Convolutional Neural Networks (CNNs) in video surveillance for anomaly detection. For example, multi-instance learning has been used to deal with weak labels like in video-level data, where the labels are not provided for each instance in the video but rather on the video level only (Sultani et al., 2018). Their model learns to localize anomalies by learning the detection between normal and abnormal events without frame-level annotations.

The unsupervised anomaly detection approach is exploiting the unlabeled datasets to detect the anomaly. Since the datasets are unlabeled, the definition of anomaly must be well defined (Mohindru and Singla, 2021). The multi-level representation framework provided by Conditional GANs is also a common application—for example, denoising autoencoders and Conditional GANs (cGANs) (Vu et al., 2019). These methods try to maximize robustness and accuracy in detecting anomalies by exploiting both intensity and motion data at various representational levels. The approach significantly improves pixel-level Equal Error Rate on different benchmark datasets and provides a more reliable system in monitoring complex environments, such as public spaces, with traditional unsupervised methods that could fail because of noise and environmental changes.

The semi-supervised anomaly detection approach is using datasets with neither fully labelled nor fully unlabeled. Instead, it employs the weakly labelled datasets to detect anomaly (Mohindru and Singla, 2021). To improve the discrimination between normal and anomalous classifications of the video, Sarker et al. (2021) used the spatiotemporal features extracted by a temporal 3D convolutional neural network (T-C3D) with a new ranking loss function. This feature contributes reasonably to the increase in the score gap of the abnormal and normal videos, aiming at a reduction in the false-negative rate. The proposed methodology attained competitive performance without any model fine-tuning, thus substantiating its strengths towards robust generalization across the wide spectrum of diverse scenarios in the wild. This study is an example of a big step toward advancing semi-supervised anomaly detection methodologies that harness the strengths of supervised learning's accuracy and few requirements for labelling in unsupervised approaches.

3. Methodology

The purpose of this paper is to detect the anomalies that is defined as the difference in behavior of an individual from other. To achieve this, following steps were taken.

3.1 Object Detection and Tracking

The initial step of our methodology involves object detection using YOLOv5 (Jocher, 2020), with a focus on detecting all individuals. After acquiring the bounding boxes of all individuals, we input these bounding boxes to DeepSORT algorithm to keep track of these people and assign ID to them. This process is the foundation of our approach since it allows us to detect and track individuals.

3.1.1 YOLOv5: We used a pretrained model for YOLOv5, referred to as YOLOv5x (Jocher, 2020). We nominally trained this model since it satisfied the needs of high accuracy and low inference time. YOLOv5x processes images of 640x640 pixels with a mean Average Precision of 50.7% across various Intersection over Union (IoU) thresholds (from 0.5 to 0.95), indicating a strong ability to accurately predict object locations and classifications. Specifically, at an IoU threshold of 0.5, its accuracy (mAP) reaches 68.9%. This model also has speed of 4.8 milliseconds per image when processing images in batches of 32 (Jocher, 2020).

The original model is provided as a PyTorch model file. We export this model file into an Open Neural Network Exchange model. By doing the format conversion, the model allows us to use in OpenCV-Python library (Itseez, 2015).

3.1.2 DeepSORT: The next major task is to keep track of individual across the video frames using DeepSORT (Wojke, Bewley, and Paulus, 2017). YOLOv5 provides the highly accurate real-time detections. These detections are then fed into DeepSORT to generate trajectories for multiple objects over time, even in scenarios with occlusion or varied object appearances. The advantages of integration of YOLOv5 and DeepSORT include enhanced detecting and tracking stability and accuracy as DeepSORT uses deep learning algorithms to differentiate between objects effectively. The tracked persons were assigned their IDs. These IDs would not disappear until the video ends, or the tracked person never shows again.

3.2 The Geometric rectification for Positioning

After the detection and tracking, the foundation of the algorithm was done. Since the view in the video is 3 dimensional and oblique, we require a 2 dimensional, plain, and bird's eye view in order to collect more accurate data from detected targets (Song et al., 2021). To convert the video's 3D view into a 2D bird's eye perspective, we employed the homography transform. This transformation allows us to create a 2D representation of the scene, optimizing tracking accuracy and data recording for individuals' movements (Chen, Hou, and Zhang, 2022). The homography transform is given as:

$$\begin{bmatrix} x' \\ y' \\ s' \end{bmatrix} = H \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (1)$$

where s' = a scaling factor
x, y = coordinates of a point in original image
x', y' = coordinates of a point in destination image
H = 3x3 homography matrix

S' is a scaling factor, often set to 1 (s=1) for non-projective transformations. $\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$ are the homogeneous coordinates of the corresponding point in the destination image. $\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$ are the homogeneous coordinates of a point in the source image. To calculate *H*, OpenCV-Python library was used.

To complete this process, we marked four source points from the source image and four destination points from the destination image to calculate *H*. The four points from source and destination image were top left, top right, bottom right, and bottom left point of the plane we intend to project.

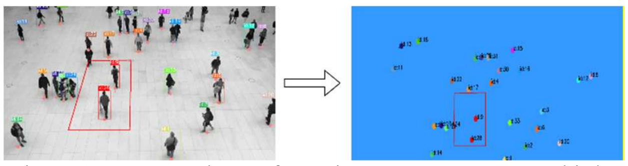
Figure 1. Homograph transformation process to generate bird's eye view.

Fig. 1 shows the homograph transformation process of a frame. The four source points selected were the top left, top right, bottom right, and bottom left corner of the floor in the site. To compute four destination points, we use:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \quad (2)$$

where d = Euclidean distance of two points
x_1, y_1 = coordinates of the first point
x_2, y_2= coordinates of the second point

We first compute the w_1 by using top right and top left points selected from original frame. w_1 is the Euclidean distance between top right and top left points, which is also the candidate width of the right image in Fig 1. w_2 can be computed in the same way as w_1 with bottom right and bottom left points. After that, we choose the longer width W' among w_1 and w_2 as the actual width of the bird's eye view. Besides, we also need the longer height H' among h_1 and h_2 as the actual height of the bird's eye view. h_1 and h_2 can be calculated by using top right, bottom right, top left, and bottom left points. Therefore, the four destination points is the following.

$$dst = [(0,0) \quad (W',0) \quad (W',H') \quad (0,H')] \tag{3}$$

where dst = [(top left) (top right) (bottom right) (bottom left)]
W' = the longer width between w_1 and w_2
H' = the longer height between h_1 and h_2

dst is the destination points array containing top left, top right, bottom right, and bottom left points accordingly. The original frame on the left in Fig. 1 contains detected and tracked individuals and a red rectangle area. The coordinates of each person's position in the original image are taken from the bottom center of their bounding boxes which are generated by YOLOv5 and tracked by DeepSORT.

$$P_{bottom\ center} = (\left(\frac{x+w}{2}\right),\ (y+h), 1) \tag{4}$$

where $P_{bottom\ center}$ = coordinates of a people's position in original image
$x,\ y$ = the top left coordinates of the bounding box
w = the width of the bounding box
h = the height of the bounding box

The Eq. (4) shows the calculation of the bottom center of the bounding box. We added a "1" after (x,y) 2D coordinates to make it (x,y,1) 3D since the calculation of person's position within the homography transform involved a matrix multiplication between a 3x3 H matrix and a 3x1 $P_{bottom\ center}$ matrix, which is necessary.

$$\begin{bmatrix} x' \\ y' \\ s' \end{bmatrix} = H \cdot P_{bottom\ center} \tag{5}$$

$$P_{homo} = (\frac{x'}{s'}, \frac{y'}{s'}) \tag{6}$$

where s' = a scaling factor
x', y' = coordinates of a point after homography transform
H = 3x3 homography matrix
$P_{bottom\ center}$ = coordinates of a people's position in original image
P_{homo} = coordinates of people's position in destination image

To obtain people's position P_{homo} after homography transform, we applied Eq. (5) and (6) with the H matrix by $P_{bottom\ center}$. Thus, we gained the individual's position on the bird's eye view.

After homography transformation, the resulting bird's eye view on the right in Fig 1 includes the individual's transformed position marked as colored dots with corresponding IDs. By doing the homograph transform, the position of the detected person can become more precise later in the calculation of the motion vector.

3.3 Motion Extraction

This section describes how motion data is extracted from tracked individuals in the crowd scenes. This process involves calculating motion vectors that represent the direction and speed of everyone over time.

The process of extracting motion is fundamental to what will be understood as the dynamic movements of people in a crowd, which may be seen as possible anomalies. Having obtained the exact positions of everyone through the geometric rectification process in the bird's eye view, we proceed to calculate their motion vectors, which are vectors that encapsulate the direction and speed of movement over time.

The motion vector of everyone is calculated by comparing the position from the current frame with the position of the next frame using Eq. (7).

$$\vec{v}_{i,t} = (x_{i,t+1} - x_{i,t},\ y_{i,t+1} - y_{i,t}) \tag{7}$$

where $\vec{v}_{i,t}$ = motion vector for person *i* at frame *t*
$x_{i,t}$, $y_{i,t}$ = coordinates of individual *i* at frame *t* in bird's eye view
$x_{i,t+1}$, $y_{i,t+1}$ = coordinates of individual *i* at frame *t+1* in bird's eye view

These vectors are calculated for each individual across all available frames, allowing us to track the trajectory and speed of movement throughout the video sequence.

3.4 Anomaly Detection Using KDE

Kernel Density Estimation (KDE) is employed in our approach to effectively detect anomalies in motion data deduced from crowd scenes. KDE is a non-parametric method used to estimate the probability density function of a random variable (Rosenberger et al., 2022). This method is able to ascertain the abnormal patterns within the motion data, which are far away from the motions average that typically occurs in the dataset (Kerpicci, Ozkan, and Kozat, 2021).

KDE functions by placing a continuous kernel at every data point and then summing up these kernels over to produce a smooth estimation of the underlying distribution. In the context of our study, any motion vector derived from the surveillance footage shall be taken as a data point. The KDE is applied to these vectors to estimate the density distribution of typical movements within the crowd. Based on this distribution, one has the possibility to detect the outliers as movements which are far away from typical (Sadeghi-Tehran and Angelov, 2012).

The steps for KDE anomaly detection are:

1. **Data Preparation**: Motion vectors are extracted and ready to be analyzed.
2. **KDE Application**: KDE applies to the motion vector data. We used a common Gaussian kernel known for being smooth and effective in highlighting the outliers.
3. **Threshold Setting**: Based on the density values obtained from KDE, a threshold for anomaly detection is set. Movements whose density values go below this threshold are considered anomalies.
4. **Anomaly Identification**: All points that fall below the threshold set are marked as possible anomalies. Those are the anomalies of movements that do not belong to a typical pattern in the crowd.

Seaborn library (Waskom, 2021) in Python was utilized to visualize KDE on motion vectors. However, the kernel density value of a given motion vector cannot be extracted from Seaborn's implementation. We, therefore, used SciPy library (Virtanen et al., 2020) to extract the kernel density value of a specific motion vector. The bandwidth as a major parameter was set to be "scott" for both methods. In addition, the weights as another main parameter were set to be "None" along with all the other parameters set to be their default value.

Before setting a threshold, it is also essential to understand the distribution of the data. Visualize the KDE of the motion vectors to understand where most data points occur and where density starts to thin out, indicating less common movements. This is where one has to decide what percentile best determines the "normal" data. After that determination, the threshold would be set at that percentile to flag all data below that as anomalous.

$$P_i = \sum_{t=0}^{N} \frac{1[KDE(\vec{v}_{i,t}) \geq threshold]}{N} \tag{8}$$

where P_i = probability of a person i who is an anomaly
N = total number of motion vectors for person i
$\vec{v}_{i,t}$ = t-th motion vector for person i
$KDE()$ = a function to get the kernel density value of a given motion vector
$threshold$ = the threshold to check if a motion vector is a anomaly
$1[condition]$ = the indicator function returns 1 if $condition$ is true, 0 otherwise

Each motion vector in a frame, which represents the movement of a person, is evaluated against a predefined threshold. The assessment aims to determine if the motion, while typical, deviates significantly from the expected motion by indicating an anomaly. If any motion vector is found to be anomalous in that it deviated abnormally over the threshold, the count of that particular individual is thus increased. This is done to every motion vector associated with the person across all the frames they appear in. After evaluating all anomalous motion vectors related to the person, the overall count of the anomalous vectors related to the person is then compared to the total number of all motion vectors related to the person, as per Eq. (8). Then, the calculated ratio P_i is compared against a set threshold to decide whether or not an individual is generally behaving anomalously. This ratio P_i is equal to or larger than 0.5 or 0.6, depending on the stringency of the anomaly detection required.

Anomalies detected by KDE could be used to detect a person moving differently from general flow either at a much higher or much lower speed than the surrounding individuals, or even taking irregular paths in terms of common or normal trajectories. This is shown in the surveillance footages through visible signs that show these individuals with abnormalities.

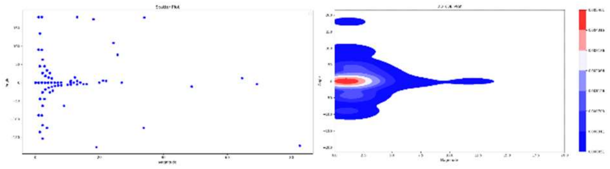
Figure 2. The scatter plot and the KDE plot of a video

Figure 2 illustrates the distribution of the motion vector and its KDE plot of a testing video. The x-axis represents speed (or magnitude), while the y-axis denotes the direction of movement (or angle). Points falling in the low-density regions of this plot are marked and considered anomalies in the video frames.

Figure 4 shows an individual with ID 23 and pink trajectory detected as an anomaly. This person was riding a bike across from the top left to the bottom right of the street with a different direction and speed compared to others. The motion vectors of this person fall into the low-density regions of the KDE plot and then being considered as anomaly.

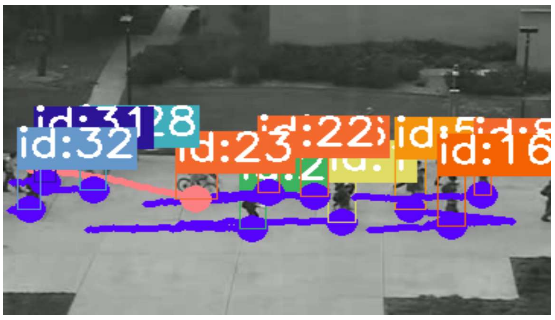

Figure 3. Anomaly detected within one of video frames in UCSD Anomaly Detection Dataset

In this regard, KDE for the detection of anomalies in scenes with crowds has the following advantages:

- Flexibility: KDE does not need to know beforehand what model of what is considered normal behavior; hence, it can apply to different environments and situations.
- Sensitive: This method is, therefore, sensitive to small changes in data—a very important requisite when it comes to finding out anomalies in densely populated areas.

This anomaly detection approach with KDE complements very well the geometric rectification and tracking method proposed earlier, thus providing a very robust system for continuous monitoring and hence the security of public spaces through advanced video analytics. This is an added approach that consolidates an all-encompassing solution for the issues of crowd surveillance, as well as anomaly detection, in dynamic environments.

4. Result

4.1 Dataset

The dataset used for its test was the UCSD Anomaly Detection Dataset (Mahadevan et al., 2010), a well-accepted benchmark in video surveillance for testing the performance of our method. We chose this dataset with comprehensive frame-level annotations for normal and anomalous scenarios of crowd scenes specifically to test the robustness and accuracy of our proposed method.

The dataset contains two sets, Peds1 and Peds2. In each set, video samples are single frames saved in TIFF format. More specifically, Peds1 includes 36 testing video samples, and Peds2 consists of 12 testing video samples. These testing video clips are useful in a way that they contain different scenarios, and each video, at least, has some frames that are classified as anomalous, hence providing quite a robust framework for assessing detection accuracy.

Since these individual frames were stored in TIFF format, a preliminary step in our process involved the conversion of these into a continuous video format. This obviously was necessary, as it will allow more dynamic analysis using our video processing tools. So, each sequence of TIFF images was integrated into single .mp4 video files, which subsequently allowed more fluid testing of anomaly detection with sequential frame analysis.

Frame-level annotations for abnormal events are detailed in MATLAB (.m file) format within the test folders. The 'gt_frame' field in these files is particularly important as it indicates the frames containing anomalies. This kind of precise annotation enables one to make a focused assessment, focusing his analysis only on the frames that reveal abnormal activity.

Our evaluation process exclusively utilized the testing portions of both Peds1 and Peds2. This allowed us to address only the capability of the system to detect anomalies without compromising the system with learning data on normal behavior. The evaluation at the frame level was kept only for the verification of whether the anomalies were detected within each individual frame, reflecting our model's sensitivity to changes within such a highly dynamic setting.

4.2 Anomaly Detection Result

In evaluating our anomaly detection system using the UCSD Anomaly Detection Dataset, we focused specifically on the performance of the system at detecting frame-level anomalies.

The UCSD dataset was provided with ground truth data that contained frame-level annotations denoting whether each frame contained an anomaly. This data was used as a benchmark for evaluating our system. During our evaluation, we compared each processed frame of the sequence against this ground truth to find out if it correctly detected the presence of an anomaly or not.

First, in our analysis, we computed the key metrics, pertained to the quantification of system performance.

$$Precision = \frac{TP}{TP+} \tag{8}$$

$$Recall = \frac{TP}{TP+FN} \tag{9}$$

where TP = correctly identified anomaly frames
FP = incorrectly identified anomaly frames, where they are normal
FN = failed to identify anomaly frames

UCSD Anomaly Detection Dataset	Correctly detected frames	Incorrectly detected frames		Result	
	TP	FP	FN	Precision	Recall
Peds1	4062	482	443	0.89	0.90
Peds2	1453	115	195	0.93	0.88

Table 1. Anomaly frames detection result

As Table 1 showed, the system exhibited a high ability to correctly identify anomalies in the Peds1 subset. The model successfully detected 4062 true positive frames, where anomalous activities were correctly flagged according to the provided ground truth. On the other hand, the system has given 482 false positives, whereby a normal activity has been considered as an anomaly. Moreover, the system has missed 443 anomalous frames, hence giving a false negative. This subset resulted in a precision of 0.89, meaning the system successfully identified the frames and found that 89% were anomalies. This was closely followed by recall at 0.90, i.e., 90% of all actual anomalies in the dataset were detected.

The peds2 subset showed another facet of the performance of the system. In such case, there were 1453 true positive frames identified properly by the system. Meanwhile, the number of false positive cases which were flagged was lesser than that of Peds1 and registered 115 frames that were flagged incorrectly. However, it did miss more anomalies in the system, recording 195 false negatives. The precision in Peds2 was higher at 0.93, meaning that in case an anomaly is detected, there lies high chances of this being a true anomaly, unlike in Peds1. However, the recall was a bit lesser at 0.88.

The comparative results between Peds1 and Peds2 show effective operations of the system, where both subsets presented good precision values with a clear efficacy in detecting true anomalies.

5. Discussion and Conclusion

The UCSD Anomaly Detection Dataset supported the robust validation of the anomalies' detection from the efficacy of our integrated surveillance system inside crowded public spaces. But those are the intrinsic challenges associated with human motion, where the natural oscillating patterns of stepping left and right foot introduced a complex situation in correctly differentiating between normal and anomalous behavior.

Our system is based on the tracking algorithm and derives the motion vectors as they follow an individual from frame to frame. These vectors are crucial for identifying movement directions and speeds. However, human walking exhibits a natural oscillation, which ultimately gives noisy data. This fluctuation, though normal, complicates the ability of the system to maintain a consistent tracking vector, which further influences the analysis through KDE.

It is precisely at this level that the advanced KDE needs to be tuned to a high precision to be able to discriminate between the density clusters formed by common movements and those created by truly anomalous behaviors. If the KDE bandwidth is too wide, some subtle anomalies would be obscured in the normal motion densities; if it is too narrow, then the normal variability in human gait would tend to produce excessive false detections.

In conclusion, this paper proposed a holistic system to monitor crowds and detect their anomalies using a combination of new technologies in computer vision and innovative statistical methods in a public place. The solid performance of the system applied to the UCSD Anomaly Detection Dataset speaks to its potential under real-world deployment in varied surveillance scenarios.

The integration of YOLOv5 and DeepSORT with geometric rectification and KDE-based anomaly detection gives the system a strong base against reliable object detection and tracking, making it sensitive enough to work with accuracy. Such a system not only improving security by enabling the early detection of potential threats but also offers scalability and adaptability to different environments and conditions.

Future works are going to keep refining the capabilities of the anomaly detection by overcoming the challenge of human gait in the motion extraction stage, and the dataset will be extended to cover larger and more varied environments to test the generalizability of the system. This will help ensure its operational effectiveness as a proactive tool in public safety and surveillance systems.

Acknowledgements

I would like to thank Dr. Alper Yilmaz for giving me opportunity to pursue research in PCV Lab and guidance for my paper. I am extremely grateful to all the support and help from Dr. Alper Yilmaz.

References

Arroyo, R., Yebes, J. J., Bergasa, L. M., Daza, I. G., & Almazán, J. 2015: Expert video-surveillance system for real-time detection of suspicious behaviors in shopping malls. *Expert Systems with Applications*, *42*(21), 7991–8005. https://doi.org/10.1016/j.eswa.2015.06.016

Chen, H., Hou, L., & Zhang, G. 2022: Social distance monitoring of site workers for COVID-19 using context-guided data augmentation, Deep Learning, and homography transformation. *IOP Conference Series: Earth and Environmental Science*, *1101*(3), 032035. https://doi.org/10.1088/1755-1315/1101/3/032035

Itseez. 2015. Open source computer vision library (Version 4.9.0). https://github.com/itseez/opencv

Jocher, G. 2020. YOLOv5 by Ultralytics, Version 7.0. https://doi.org/10.5281/zenodo.3908559

Kerpicci, M., Ozkan, H., & Kozat, S. S. 2021: Online anomaly detection with bandwidth optimized hierarchical kernel density estimators. *IEEE Transactions on Neural Networks and Learning Systems*, *32*(9), 4253–4266. https://doi.org/10.1109/tnnls.2020.3017675

Klauser, F. R., Ruegg, J., & November, V. 2008: Airport surveillance between public and private interests. CCTV at Geneva International Airport. Politics at the Airport, 105-126.

Mahadevan, V., Li, W., and Vasconcelos, N. 2010: Anomaly Detection in Crowded Scenes. *In Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*

Mohindru, V., & Singla, S. 2021: A review of anomaly detection techniques using Computer Vision. *Lecture Notes in Electrical Engineering*, 669–677. https://doi.org/10.1007/978-981-15-8297-4_53

Nayak, R., Pati, U. C., & Das, S. K. 2021: A comprehensive review on Deep Learning-based methods for video anomaly detection. *Image and Vision Computing*, *106*, 104078. https://doi.org/10.1016/j.imavis.2020.104078

Rosenberger, J., Müller, K., Selig, A., Bühren, M., & Schramm, D. 2022: Extended kernel density estimation for anomaly detection in streaming data. *Procedia CIRP*, *112*, 156–161. https://doi.org/10.1016/j.procir.2022.09.065

Sadeghi-Tehran, P., & Angelov, P. 2012: A real-time approach for novelty detection and trajectories analysis for anomaly recognition in Video Surveillance Systems. *2012 IEEE Conference on Evolving and Adaptive Intelligent Systems*. https://doi.org/10.1109/eais.2012.6232814

Sarker, M. I., Losada-Gutiérrez, C., Marrón-Romera, M., Fuentes-Jiménez, D., & Luengo-Sánchez, S. 2021: Semi-supervised anomaly detection in video-surveillance scenes in the wild. *Sensors*, *21*(12), 3993. https://doi.org/10.3390/s21123993

Song, L., Wu, J., Yang, M., Zhang, Q., Li, Y. and Yuan, J., 2021: Stacked homography transformations for multi-view pedestrian detection. *In Proceedings of the IEEE/CVF International Conference on Computer Vision*, 6049-6057.

Sultani, W., Chen, C., & Shah, M. 2018: Real-world anomaly detection in surveillance videos. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. https://doi.org/10.1109/cvpr.2018.00678

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R., Jones, E., Kern, R., Larson, E., … Vázquez-Baeza, Y. 2020. SciPy 1.0: Fundamental algorithms for scientific computing in python. *Nature Methods*, *17*(3), 261–272. https://doi.org/10.1038/s41592-019-0686-2

Vu, H., Nguyen, T. D., Le, T., Luo, W., & Phung, D. 2019: Robust anomaly detection in videos using multilevel representations. *Proceedings of the AAAI Conference on Artificial Intelligence*, *33*(01), 5216–5223. https://doi.org/10.1609/aaai.v33i01.33015216

Waskom, M. 2021. Seaborn: Statistical data visualization. *Journal of Open Source Software*, *6*(60), 3021. https://doi.org/10.21105/joss.03021

Wojke, N., Bewley, A., Paulus, D., 2017. Simple Online and Realtime Tracking with a Deep Association Metric. *2017 IEEE International Conference on Image Processing (ICIP)*, 3645–3649. doi:10.1109/ICIP.2017.8296962