Pedestrian Tracking and Analyse Report

Fengrui Yang, Haoyu Sun, Qiyao Zhou, Xiaohang Hu , Zheng Cao

COMP9517 Group Project

The University of New South Wales

July 2022

Abstract

This document is a report of the project: Pedestrian Tracking and Analyse. This project is supposed to give a brief review and comparison of the most advanced research in the field of object tracking. It attempts to give a practical evaluation of those existing methods by developing a Pedestrian Tracking application for tracking and analyzing pedestrians on the dataset Segmenting and Tracking Every Pixel (STEP) [1].

INTRUDUCTION

This project utilizes existing object detection and tracking techniques to track pedestrians in an open area, such as a square. The input is captured by a stationary or moving camera, and it can be in video or continuous image format. Pedestrians are identified by bounding boxes and bounding boxes associated to the same pedestrian will be linked together to track the person. It will use some visual techniques to show the bounding boxes and trajectory for each pedestrian on videos or images. All trajectories and bounding boxes are color-unique that aim to distinguish different pedestrians. Based on the detected bounding boxes and identity information, it will do some pedestrian analysis work, such as pedestrian counting, group formation and destruction, pedestrians entering or leaving. For more detail, it will count the total number of pedestrians appearing in the video from beginning to end and will also display the current number of pedestrians detected in the video. Moreover, it will analyze whether pedestrians are grouped or not while walking. The input video or images are provided at the training stage and testing stage to the model in order to obtain a precise prediction model. The frames captured by webcam placed randomly on street under bright illumination condition, the continuous frames indicate the movement of objects since the frames are consecutive and close to each other in time dimension. Apart from pedestrians, the images consist of common static and moving objects, such as trees, lamps, trash bins, animals and vehicles. This indicates during training, the model should not overfitted to pedestrian, it also needs to distinguish other moving objects. The dataset also provides pixel-level labeling images of pedestrian for every relative image which is used for training purpose of model. The pixel-level labeling image consists of red and blue parts. Red color in labeled images aims to shape the edge of pedestrian, while blue color with multiple intensity tries to separate each pedestrian for their total area. The videos are provided to well-trained models at the demonstration stage and in order to visualize the model. The video contains all images mentioned in the previous section as every single frame. The algorithm adopted in this project are YOLO (You Only Look Once) [13] and Deep SORT [15]. YOLO, an object detection algorithm which

developed recently in 2020, aims to deal with detecting instances of semantic objects of a certain class (such as humans, buildings, or cars) in digital images and videos. It's super-fast detection with high frame rate and excellent precision. Deep SORT is a popular object tracking algorithm which takes charge of tracking pedestrians in this project. The details will be explained in the method section.

LITERATURE REVIEW

This section will review some existing research related to Pedestrian Tracking to give us an outline of possible approaches to this project. Visual Tracking can be categorized into traditional methods and deep learning-based methods, deep learning-based trackers are also known as deep feature-based trackers. The survey conducted by [3] Fiza et al. have evaluated 24 handcrafted and deep feature-based methods, which observed that deep feature-based trackers outperform handcrafted trackers on average, but the speed of deep feature-based trackers is slower. Deep learning-based trackers have improved the performance of tracking algorithms and have made significant progress in recent years [3]. An experiment [2] based on the popular OTB-100, TC-128 and VOT2015 benchmarks have compared different deep learning-based trackers, which shows the Convolutional Neural Network (CNN) model can improve the performance of the tracker, for example effectively distinguish the target objects from the background. This review [2] also suggests that traditional methos, such as IVT [21], MIL [22] and TLD [23], are not feasible to be used on realistic applications.

A survey of the deep learning-based trackers [4] points out that the raising deep learning-based trackers give us a new solution to the Multiple Object Tracking (MOT) problem. One standard 2 steps approach that can combine with the MOT algorithms is tracking-by-detection [24], which needs to train 2 neural networks to do the detection and tracking separately. This survey [4] introduces some popular detection networks including Faster R-CNN [12], SSD [16] and YOLO [13]. Especially, variants of R-CNN have been widely used in some specific scenarios. The Simple Online and Realtime Tracking (SORT) algorithm is one of the first CNN based methods for pedestrian tracking. The combination of Faster R-CNN and SORT has the best performance on MOT15 dataset among all open-source methods at the time of that survey [4].

Regions with CNN (R-CNN) [11] has increased the mean average precision by 30% on the dataset VOC 2012 compared to the methods before it. R-CNN extracts around 2000 bottom-up region proposals form the input image and then uses CNN to extract features for each proposal, and finally employs the linear SVM to do classification. The Fast Region-based Convolutional Network (Fast R-CNN) [12] includes a very deep VGG16 network, which is 9 × faster than R-CNN, this deep design also improves the accuracy of the network.

You Only Look Once (YOLO) [13] is another popular CNN based object detection network, which takes object detection task as a regression problem. YOLO divides the input image into different sizes of cells, the cell located at the center of the object is responsible for detecting the object. The neural network output feature maps of those cells, which include the coordinate of the bounding box directly. The number of the bounding boxes can be predicted in each cell is limited by a hyperparameter. The whole pipeline of YOLO is a single network, this end-to-end design is simple and helps to improve accuracy.

SORT [14] is a pragmatic CNN based MOT algorithm which focuses on efficiency to deal with online object tracking tasks. Deep SORT [15] improve the performance of SORT by employing a deep association metric, which lets the network better deal with object occlusion problem and id switch problem. The deep association metric is pretrained on a large person re-identification dataset, this prior information helps achieve higher accuracy while maintaining efficiency.

METHODS

A. The workflow of Tracking-by-detection

The approach tracking-by-detection [24] can be divided into 4 computational steps, Fig. 1 shows the detail. Firstly, given the raw frames of a video and an object detector is run to obtain the bounding boxes of the objects. Then, for every detected object, different features are computed, usually visual and motion ones. After that, an affinity computation step calculates the probability of two objects belonging to the same target, and finally an association step assigns a numerical ID for each object.

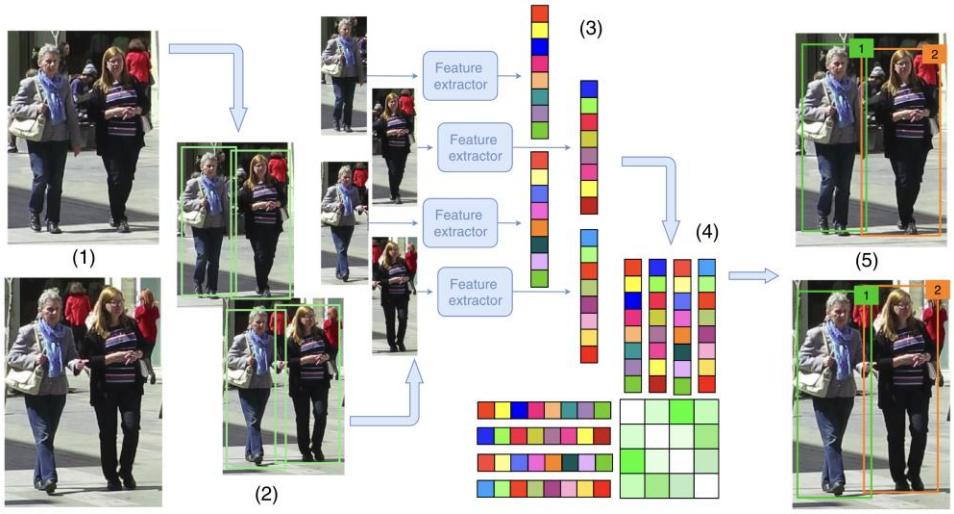

Fig. 1. Usual workflow of a MOT algorithm:

B. Pedestrian Detection and Tracking

We use yolov5m with label “people” to train our own model based on MOT and use yolo to do pedestrian detection which can help us locate a detected pedestrian in a single frame with a bounding box. Then we use deep sort Tracking to find pedestrian that persists across multiple frames with their own ID.

“In MOT system, training a CNN model can be roughly classified as training beforehand using classification datasets, training holistically with tracking data, or pre-training initially and then fine-tuning. CNNs are widely used for image classification and recognition because of its outstanding ability for feature learning. When training the CNN models, different objective functions are utilized and various training datasets are needed according to specific tasks”.[4]

During the frames’ loop process, we need to store the pedestrian information into pedestrian management which will help us draw boundary boxes with different colors around each person in every frame and label everyone with a different id.

During the frames’ loop process, we use a key-value structure in memory to store and manage the information of each pedestrian. The key is the ID of the pedestrian, and the value is the related information, which includes the bounding boxes and the trajectory of the corresponding pedestrian. That information can help us to calculate and draw the bounding box, trajectory and counting information on each frame. Fig. 3 gives us an example of the colored bounding boxes.

Fig.3 Detection Example

C. Detect pedestrians in a group

In shared space, pedestrians are often found walking in groups and behaving differently than individual pedestrians. However, automatically detecting pedestrian groups with high accuracy is not trivial given the dynamic environment and interactions in mixed traffic. During the frames’ loop process, we will check if all people boundary boxes in this frame are in same group or not. First, we need to do it through the box’s width and height to estimate the distance to the camera. If the two people’s positions are about the same and their boundaries box is similar, then they are in a group. pedestrian detection and tracking techniques to discover small groups of people who are traveling together. Determining the group structure of a crowd provides a basis for further mid-level analysis of events involving social interactions of and between groups. Like sitting people usually are group with people sitting next to him/her.[]

Detecting pedestrian groups with high accuracy is not trivial given the dynamic environment and unstable bunding boxes. The simple two steps approach is to check all paired pedestrians if they are in the same

group and assign a group ID to them, and then merge all small groups into large groups. One critical point of this algorithm is how to check if 2 pedestrians are in the same group. Firstly, we use the box's width and height to estimate their distance to the camera. According to the Pinhole camera model, the distance of the object to the camera is inversely proportional to its width or height, therefore the equation used to estimate the distance is:

$f = 1 / (Width + Hight)$

$distance = \gamma \times f$

as we assume the size of all pedestrians are the same, where the param γ can be estimated by experience. After that for the pedestrians have the similar "camera distance", we calculate the distance between the centers of those 2 boxes, note this distance should also inversely proportional to the size of the box, which need to multiply by f. If this "center distance" is smaller than a threshold, the checking algorithm will return True. Fig. 3 shows the output of the group checking algorithm.

Fig.4 Group Example

D. Pedestrian Counting

During the frames' loop process, we will get detection data from yolo and store it in Pedestrian management with frame id and all boxes in this frame. So, we just need to count how many boxes are in this frame to get the total count of pedestrians present in the current video frame. For the total count of all unique pedestrians detected since the start of the video, we can count how many unique ids of box in pedestrian management.

We use two Sets to store current Pedestrian IDs and historical Pedestrian IDs separately. The number of current pedestrians and appeared pedestrians is the length of those two Sets.

Fig.5 Pedestrian Counting Example

E. Manually draw rectangular region

During the frames' loop process, we set cv2.setMouseCallback with event of Mouse Left Down and Mouse Left Up to record the position of rectangular region. After that, when we draw boundary box in the frame, we can only draw and count those boxes which center point are in the rectangular region.

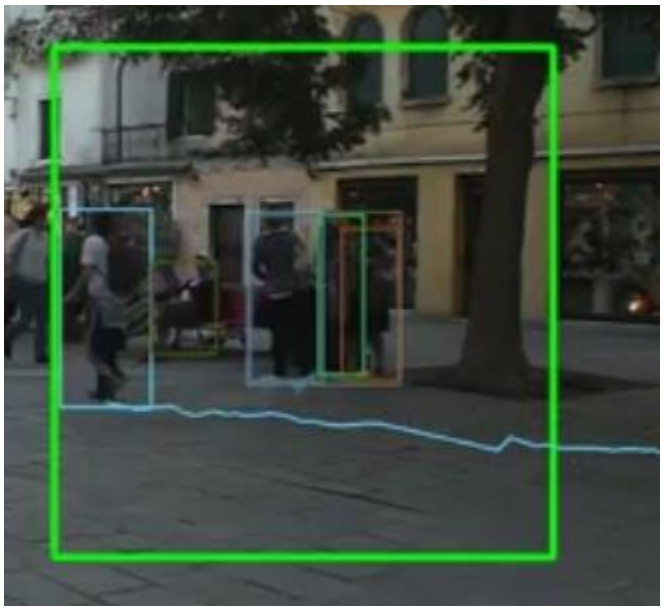

Fig.6 Manually Region Example

F. Pedestrians entering or leaving the scene

The algorithm will find those pedestrians who are just entering the screen and highlight them in each frame. During the frames' loop process, we will check the boundary area of window trying to find box that is very close to the boundary and just appear. After that we will check if the pedestrian is getting closer to the center of the screen. If True, it means this is an entering pedestrian. Inversely, if the pedestrian is moving away from the center, close to the boundary and disappear in the next video frame. It means this is a leaving pedestrian.

G. Kalman Filter

The Kalman Filter has been added to YOLOv5 to help to improve the performance. Visual features were combined with spatial ones, extracted with a Kalman filter, and then an affinity matrix was computed. We used Unscented Kalman filter to keep track of the dynamics of the motion of each detected human. The Kalman Filter produces estimates of hidden variables based on inaccurate and uncertain measurements. Also, the Kalman Filter provides a prediction of the future system state based on past estimations. And position can be easily calculated using Newton's motion equations [25].

$$x=x0+v0\Delta t+½*a\Delta t^2$$

x	is the target's position
x0	is the target's initial position
v0	is the target's initial velocity
a	is the target's acceleration
Δt	is the time interval

EXPERIMENTAL RESULTS

A. Result of Track Pedestrians

This project has basically achieved the detection and tracking of pedestrians, while in the evaluation of the results we have combined qualitative and quantitative analysis methods to evaluate our project. For the quantitative analysis, we mainly used indicators commonly used for binary classification, such as precious and recall.

TP (True Positive) prediction is a positive sample, and the prediction is correct. The closer this metric is to the number of annotations of pedestrians in the validation set, the higher the detection rate of the detector.

FP (False Positive) predicts a positive sample, but the prediction is wrong. This indicator reflects the false detection rate, the lower the false detection rate the better.

FN (False Negative) A negative prediction but an incorrect prediction, i.e., a sample that should have been detected was not detected.

Precious describes the proportion of TP in the test results and is calculated as Precious = TP/(TP+FP), the larger the indicator, the higher the detection accuracy.

Recall describes the detection rate of the marked pedestrians and is calculated as Recall = TP/(TP+FN).

Considering that the test file has 450 frames, the quantitative analysis was carried out only 3 times with random sampling, 10 frames each time for analysis, and the following results were obtained:

TABLE I. INDICATORS OF PEDESTRIANS DETECTION

	TP	FP	FN	precious	recall
1	14.0	0.2	3.0	98.6%	82.4%
2	15.2	0.4	3.6	97.5%	80.9%
3	14.6	0.4	3.4	97.3%	81.1%
mean	14.6	0.3	3.3	97.8%	81.5%

From this it can be concluded that the pedestrian detection carried out in this project has a high degree of accuracy and the results obtained are reliable.

As for the qualitative part of the analysis, we have selected some representative types of error detection, and humanoid models are incorrectly detected as humanoid, in addition, it is often difficult to detect pedestrians whose features are not obvious due to their proximity to the background and their distance.

Besides, there are certain problems with the marking of boxes, including but not limited to the switching of box IDs and the unstable size of boxes, for which no suitable solution can be found in this group project.

Fig. 1. Examples of detection errors or non-detection

Overall, task 1 was completed well, perhaps there is a better algorithm to optimize the detection results that we need to continue to learn and research.

Overall, the performance of pedestrian detection algorithm is acceptable in the simple environment. Better pedestrian detection algorithms, especially for various complex situations, can be explored in further research.

B. Result of Count Pedestrians and Analyze Pedestrians

Task 2 and task 3 was equally well done. Based on the results of task 1, we show the counts and result of analysis at the bottom of the video as follows:

Fig.2. Examples of Pedestrian statistics analysis results

The results are presented in the form of 'numbers of pedestrians: {a/b}. In group: {c}. alone: {d}. (In the region: {e})'.

For this form, a is denoted as the total count of pedestrians present in the current video frame while b represents the total count of all unique pedestrians detected since the start of the video. As for c and d, they represented how many pedestrians walk in groups and how many walk alone. e indicates the total number of pedestrians in the manually drawn rectangular box.

If the group forms, there will be a larger frame that will enclose all the pedestrians within the group, while the destruction of the group will cause the group frame to shrink or disappear. In addition, new pedestrians entering the video area will have a 'NEW' mark to the left of the detection box to indicate the situation.

The results of all the requested statistical analyses are displayed below the video, and if the boxes in task 1 are drawn correctly, the results displayed are also correct.

DISCUSSION

5.1 Raw Training Data and Data Pre-processing

The benchmark datasets used in this project is STEP [1], which contains segmentation labels for most pedestrians in each image. Apart from missing pedestrians, each pedestrian has a unique label ID. This makes it straightforward to transform annotations into YOLO [27] format. However, the incomplete annotation of pedestrians causes negative impact on the results, for instance flickering bounding boxes and occlusion interference, which require more work in tracking and post-processing to compensate.

5.2 Model Training

This project adopts three YOLOv5[27] models with different sizes trained on STEP [1]. The overall best model is 'my_m' because it is hardware friendly and generates the least number of incorrect bounding

boxes in demonstration. We also conduct valuations from YOLO COCO AP [27] on the three models, and the ‘m’ size model performs best.

Fig. 8. Valuations on three YOLOv5[27] models trained on STEP datasets[1]; ‘my_s’ is based on YOLOv5s[27]; ‘my_m’ is based on YOLOv5m[27]; ‘my_l’ is based on YOLOv5l[27] .

5.3 Tracking Methods and Post-processing

This project utilizes Deep SORT [28] to analyze features and assign IDs to bounding boxes, as well as our post-processor to estimate the distance between the pedestrian and the camera and to eliminate occlusion interference. Since groups are not annotated in the model, a criterion is developed and implemented in the post-processor to detect group formation and destruction, which works fine. Additionally, the post-processor solves the problem that part of pedestrians can no longer be detected after being blocked. Furthermore, excess boxes due to model errors are also eliminated. Nonetheless, our tracker is inefficient and has a low frame rate. Fortunately, this project is not pursuing real-time framerate, and the quality of tracking is crucial, thus the tracker based on Deep SORT [28] could be a good choice. Another improved type of Deep SORT is Strong SORT [29], but it is not used in this project because it does not provide macroscopic accuracy improvement and even occasionally causes frame drops.

CONCLUSION

In this project, we review some traditional methods in Pedestrian Tracking and the recent deep learning-based trackers. We develop our model on STEP datasets [1] based on YOLOv5 [27] and implement the counter, trajectory detector and group analyzer based on Deep SORT [28]. In addition, we evaluate our models and find the most suitable one for the datasets specified by this project, which is based on YOLOv5m [27]. Subsequently, we deploy metrics to assess the results of our methods, which satisfy expectations just as viewed with the naked eye. However, in the scene of a dense crowd, this project is not ready for practical application, and one of the reasons is that the original data is not sufficiently labeled. The other reason is the tracking algorithm for the bounding boxes is not completely accurate. In future research, we should continue to modify Deep SORT [28] algorithm as well as Strong SORT [29] and develop and perfect other tracking algorithms.

This project reviews some traditional and deep learning-based approaches in pedestrian tracking and builds an application on the STEP datasets [1]. This project focuses on the combination of YOLOv5 [27] and Deep SORT [28], which is one of the most advanced deep learning-based methods. Based on this combination, we have further developed some application functions, such as pedestrian counting and group analyzing to help to get a more practical evaluation. Both the objective and subjective test results show the model YOLOv5m has the best performance among all tested models. Generally, the final program worked well to detect and track pedestrians in the public area on some simple scene. But for complex scenarios, such the dense crowds, dark backgrounds and objects far away, the output of the model is not stable. Id switching, occlusion and stability of the bounding box are widely known problems in this research field, this project shows those problems can seriously affect the performance of the application. The application layer takes some specific algorithms to alleviate those problems, such as using cache to deal with occlusion problems and average box size to deal with instability problems, seems necessary. For further research, more models can be explored, such as Fast R-CNN [12] and Strong SORT [29], and the combination of the traditional and deep learning base methods may help to find a

more balanced solution regarding speed and accuracy. More diverse data sets can also be employed in both the training and the evaluation phase.

REFERENCES

[1] Weber, M., Xie, J., Collins, M., Zhu, Y., Voigtlaender, P., Adam, H., Green, B., Geiger, A., Leibe, B., Cremers, D. and Osep, A., 2021. Step: Segmenting and tracking every pixel. *arXiv preprint arXiv:2102.11859*.

[2] P. Li, D. Wang, L. Wang, H. Lu. Deep visual tracking: review and experimental comparison. Pattern Recognition 76:323-338, April 2018. https://doi.org/10.1016/j.patcog.2017.11.007

[3] M. Fiaz, A. Mahmood, S. Javed, S. Jung. Handcrafted and deep trackers: recent visual object tracking approaches and trends. ACM Computing Surveys 52(2):43, April 2019. https://doi.org/10.1145/3309665

[4] G. Ciaparrone, F. L. Sánchez, S. Tabik, L. Troiano, R. Tagliaferri, F. Herrera. Deep learning in video multi- object tracking: a survey. Neurocomputing 381:61-88, March 2020. https://doi.org/10.1016/j.neucom.2019.11.023

[5] M. Y. Abbass MY, K.-C. Kwon, N. Kim, S. A. Abdelwahab, F. E. A. El-Samie, A. A. M. Khalaf. A survey on online learning for visual tracking. The Visual Computer 37(5):993-1014, May 2021. https://doi.org/10.1007/s00371-020-01848-y

[6] Y. Zhang, T. Wang, K. Liu, B. Zhang, L. Chen. Recent advances of single-object tracking methods: a brief survey. Neurocomputing 455:1-11, September 2021. https://doi.org/10.1016/j.neucom.2021.05.011

[7] E. Meijering, O. Dzyubachyk, I. Smal, W. A. van Cappellen. Tracking in cell and developmental biology. Seminars in Cell and Developmental Biology 20(8):894-902, October 2009. https://doi.org/10.1016/j.semcdb.2009.07.004

[8] D. Chaudhary, S. Kumar, V. S. Dhaka. Video based human crowd analysis using machine learning: a survey. Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization 10(2):113-131, October 2021. https://doi.org/10.1080/21681163.2021.1986859

[9] S. M. Marvasti-Zadeh, L. Cheng, H. Ghanei-Yakhdan, S. Kasaei. Deep learning for visual tracking: a comprehensive survey. IEEE Transactions on Intelligent Transportation Systems 23(5):3943-3968, May 2022. https://doi.org/10.1109/TITS.2020.3046478

[11] Rich feature hierarchies for accurate object detection and semantic segmentation

[12] Girshick, R., 2015. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision* (pp. 1440-1448).

[13] You Only Look Once: Unified, Real-Time Object Detection 2015

[14] Bewley, A., Ge, Z., Ott, L., Ramos, F. and Upcroft, B., 2016, September. Simple online and realtime tracking. In *2016 IEEE international conference on image processing (ICIP)* (pp. 3464-3468). IEEE.

[15] Simple Online and Realtime Tracking with a Deep Association Metric (Deep SORT) 2017

[16] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y. and Berg, A.C., 2016, October. Ssd: Single shot multibox detector. In *European conference on computer vision* (pp. 21-37). Springer, Cham.

[21] Ross, D.A., Lim, J., Lin, R.S. and Yang, M.H., 2008. Incremental learning for robust visual tracking. *International journal of computer vision*, *77*(1), pp.125-141.

[22] Babenko, B., Yang, M.H. and Belongie, S., 2010. Robust object tracking with online multiple instance learning. IEEE transactions on pattern analysis and machine intelligence, 33(8), pp.1619-1632.

[23] Kalal, Z., Mikolajczyk, K. and Matas, J., 2011. Tracking-learning-detection. *IEEE transactions on pattern analysis and machine intelligence*, *34*(7), pp.1409-1422.

[24] Andriluka, M., Roth, S. and Schiele, B., 2008, June. People-tracking-by-detection and people-detection-by-tracking. In *2008 IEEE Conference on computer vision and pattern recognition* (pp. 1-8). IEEE

[25] A. B. (n.d.). *Online kalman filter tutorial*. Kalman Filter Tutorial. Retrieved July 25, 2022, from https://www.kalmanfilter.net/default.aspx

[26] W. Ge, R. T. Collins and R. B. Ruback, "Vision-Based Analysis of Small Groups in Pedestrian Crowds," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 5, pp. 1003-1016, May 2012, doi: 10.1109/TPAMI.2011.176.

[27] GitHub - ultralytics/yolov5, accessed 11 July 2022, <https://github.com/ultralytics/yolov5>.

[28] GitHub - ZQPei/deep_sort_pytorch, accessed 15 July 2022, <https://github.com/ZQPei/deep_sort_pytorch>.

[29] GitHub - dyhBUPT/StrongSORT, accessed 21 July 2022, <https://github.com/dyhBUPT/StrongSORT>.

[26]