Utilization of Temporal Detection Consistency for Improving the Multi-Object Tracking

Abhyudaya Singh Tak[0000−0002−2504−2253] and Soon Ki Jung[0000−0003−0239−6785]

Kyungpook National University, Daegu, South Korea
{abhyudaya,skjung}@knu.ac.kr

Abstract. Various different improvements have been made in the field of multi-object tracking for separate tracker and detector algorithms. It is often shown that the joint trackers are more robust than the separate trackers but this research is conducted to show how much the separate tracker models have been improved and how much more they can still be improved. Various different research has been done to identify the current problems in the separate trackers. An algorithm to improve the detection using the past frame data for the separate tracker and detector scenario, and an algorithm to evaluate the consistency of the detector using previous and current frame tracker data have been developed. The results show that the algorithms seem to have improved the performance of not only the detection but also the entire real-time tracking model. In this paper, the results are visualized and performance is evaluated using the benchmark MOT datasets.

Keywords: Computer vision · Multi-object tracking · Bounding box.

1 Introduction

Multi-Object Tracking (MOT) plays an essential part in video understanding. It aims to discover and track all specific classes of objects frame by frame. The tracking-by-detection paradigm [4], [17], [5] has dominated the multi-object tracking task a number of times. It performs by spotting various objects per frame and formulates the MOT problem as a data association task. Benefiting from high-performing object detection models, these tracking-by-detection styles have gained favor due to their excellent performance. Still, these approaches generally bear multiple calculations having performance-costly building blocks, similar to a detector and an embedding model. To crack this problem, several recent methodologies integrate the detector and embedding model into a unified architecture. Also, training these joint detection and tracking models appears to produce better results compared to the separate one [14]. Therefore, these styles (joint trackers) achieve similar or indeed better preciseness compared to tracking-by-detection ones (separate trackers). The success of joint trackers has motivated experimenters to design unified tracking fabrics for colorful factors e.g., discovery, stir, embedding, and association models. But these have their

own problems such as competition among their own components and less open source material and data to train these models. That is the sole reason to choose the StrongSORT [9], which is the new improved version of the old DeepSORT [27] and OC-SORT [8] methods for use in this paper.

Extensive research was conducted to identify the current problems in the tracking by-detection paradigm and multi-object tracking field where some valid points were discovered. A problem in the separate tracker models was encountered where the detection model is entirely independent of the tracking task and is also not aware of the detected objects in the previous frames which can be a waste of the potential information that a detector can use to improve detection as well as the tracking performance, so an idea to build an algorithm for improving the results of the detection using the information that the detector has already come across from detection in the previous frame was identified. It can be a vast area to research but it has been narrowed down to two tasks, improving the detection in subsequent frames using the information of the previous frame as well as fixing multi-class bounding boxes for the same object using the same algorithm.

The evaluation of the multi-object trackers has also come a long way from where they were a few years ago. There are separate evaluation techniques for detectors as well as separate evaluation techniques for trackers but for this paper, we implemented an idea to find the consistency mistakes of the detector using the tracker information of the previous frame, and the explanation and results can be found in the later sections of this paper.

The main contributions of this paper are as follows:

- A method is presented to improve the detection provided by the detector by utilizing the previous-frame information as well as solving the issue of multi-class detection for the same object in the multi-object tracking for real-time applications.
- A new algorithm is developed to evaluate the detector with the help of a tracker that checks the bounding box size mistakes made by the detector in real-time.
- The method is evaluated by conducting many experiments on the MOT16 and MOT17 benchmark datasets to show how the method has improved the detection from the detector as well as the tracker's performance when used in real-time.

2 Related Work

There have been various developments in multi-object tracking and visual object tracking fields over the past couple of years. This paper explores problems in the tracking-by-detection paradigm, a sub-field in 2D multi-object tracking. There have also been developments in evaluating these models, be it developing new and improved metrics or the creation of new algorithms to test these models in a different manner. With this related works section, some sections and works

related to or close to the field this paper is based on are covered before we dive deep into our methodology and experimental results sections later in this paper.

2.1 Visual-Object Tracking and Datasets

To understand this tracking field more research was conducted in the field of visual-object tracking. A research work [24] covers up mostly every basic thing there is to know about the said field, from the introduction of the field to various available benchmark datasets for training and evaluating the visual object tracking models.

Visual Object Tracking is a task where the objective of the tracker is to locate or track a particular entity or object regardless of the challenges in all of the frames of a video, where the tracker is only given the location of that entity or object in the first frame. It is important to note that the tracker may have to overcome various challenges due to different factors affecting the video during the training and testing phases. Also, it is worth noting that the tracker might not perform the same on all the benchmark datasets currently available cause that hugely depends on how the tracker has been implemented, what training datasets were used, and what benchmark datasets were used to test its performance. Some trackers might perform well in a particular benchmark dataset but might not perform well in others. Some of the flagship benchmark datasets keep updating every year for a good performance evaluation, forcing developers to take a more generalized approach. It is a prime computer vision area and has been proven to be a fundamental concept for a lot of applications like sports [16], autonomous robots [21], self-driving vehicles [12], surgery [7], AR [1], etc.

The research work [2] revisited the challenges in visual object tracking and compared two of the most significant benchmark datasets, OTB and VOT, respectively. The aim was to get a better understanding of which protocols are preferred for what tracking objective and to ultimately compare the two datasets mentioned above. To check for the robustness of a particular tracker they introduced a new concept of mirror tracking which could help in identifying the overfitting scenarios.

2.2 3D Multi-Object Tracking

This area of tracking is also catching a lot of wind as could be seen in the recent development of this field. Authors are focusing on 3D bounding box tracking for better visualization and increased accuracy in tracking.

The research work [26] discusses a new accurate, simple, and real-time 3D MOT baseline based on SORT [4] (Simple Online and Real-Time Tracking) and, a new evaluation metric based on CLEAR [3] MOTA (Multi-Object Tracking Accuracy) and MOTP (Multi-Object Tracking Precision). They did the evaluation on the KITTI [13] dataset and achieve state-of-the-art performance on 3D MOT by creating a 3D extension to the official KITTI 2D MOT evaluation.

One research work for 3D MOT in the tracking-by-detection paradigm, SimpleTrack [18] takes a very basic but effective approach of breaking down all the components of the tracking-by-detection model and analyzing each module to find out their failure cases or shortcomings and then proposing corresponding solutions for them followed by combining them into a simpler baseline model which achieves state-of-the-art results.

2.3 MOT Evaluation Metrics

We have worked with the CLEAR [3] MOT metric in this paper to the sub-metric CLEAR MOTA (Multi-Object Tracking Accuracy) and MOTA+ [23] for evaluation of the proposed algorithm.

$$MOTA+ = 1 - ((|FN| + |FP| + |IDSWs| + |IDTRs|)/|gtDet|). \quad (1)$$

The MOTA+ in Equation (1) introduces IDTRs (identity transfers) error ratio, which is the transpose of the IDSWs (identity switches). In short, it finds the error cases where the id assigned in the previous frame is transferred to a different object in the next frame.

3 Methodology

This section discusses the models that were chosen and used for the research, the problem statements explaining the problem, and solutions to those problems mentioned along with their flowcharts for better visual explanations. This section is further divided into several subsections for an organized explainable approach.

3.1 Detector

The detector model used in this paper is YOLO [19], [20], [6] specifically YOLOv5. The reason for choosing this version of YOLO over the more recently released ones is the fact that the main focus of this research is on real-time usage and version 5 of YOLO assists that goal because of its good accuracy with good real-time speeds to work with. YOLO, when released was a major breakthrough in the field of object detectors. Framing of the object detection by the model was done as a regression problem instead of re-purposing classifiers to perform the detection task. Although, it was noted that it needs a GPU to work under real-time conditions. Figure 1 shows how the data is stored in YOLOv5 after the prediction which will be used and explained later in this paper.

3.2 Tracker

Both DeepSORT [27] and StrongSORT [9] were researched while working on this paper. StrongSORT was chosen after thorough research because of the increased accuracy over the previously proposed DeepSORT method. Below are

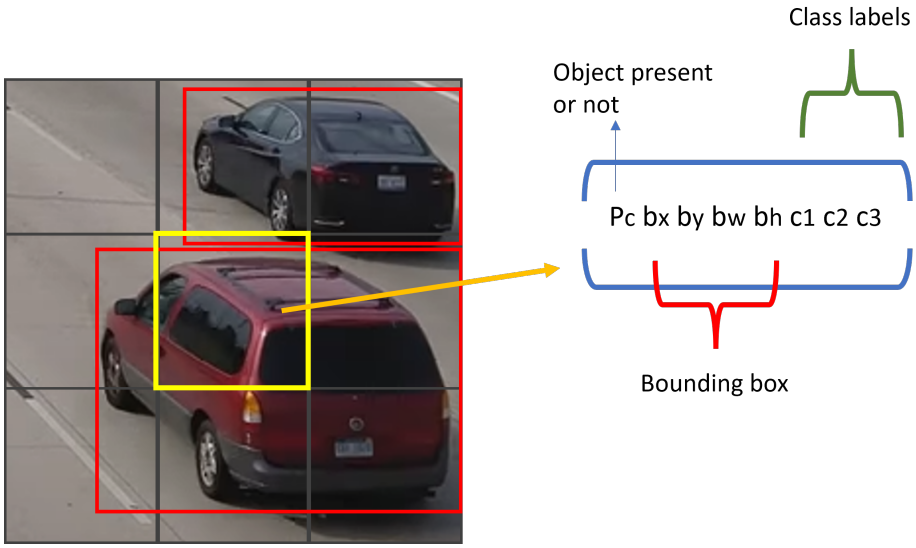

Fig. 1. Data after the YOLO model's prediction.

some points mentioned stating the differences between the two methods and the slightly altered version of the StrongSORT that is used in this paper instead of the original one. The authors of StrongSORT claimed that DeepSORT underperforms because of its outdated techniques, rather than its tracking paradigm. They simply equip DeepSORT with advanced components, achieving SOTA on MOT17 and MOT20 benchmarks. Also, for our work, we changed the BoT feature extractor [15] used in the original work with a lightweight feature extractor OSNet [29] achieving almost similar mAP (mean average precision) on the person re-id benchmark dataset while having a significantly lower number of parameters as shown in Table 1. Some differences have been stated below between Deep-

Table 1. Feature extractor model comparison.

No.	Method	mAP	Parameters
1	BoT (Resnet backbone) [15]	85.9	25.6M (98MB)
2	OSNet [29]	84.9	2.2M
3	ShuffleNet [28]	65.0	2.2M
4	MobileNet V2 [22]	69.5	3.5M

SORT and the newer StrongSORT to show why choosing the latter over the former was a better choice:

– A stronger appearance feature extractor is used instead of a simple CNN.

- They replace the feature bank in DeepSORT with a feature updating strategy [25].
- Adopt ECC [11] for camera motion compensation.
- Change the vanilla Kalman filter to the NSA Kalman algorithm [10].
- Changed the cascading algorithm of DeepSORT with the vanilla global linear assignment.

The other tracker that was used to carry out the experiments is OC-SORT [8] which is the Observation-Centric SORT and as the name suggests is also based on the famous baseline SORT [4]. OC-SORT still remains simple, online, and real-time but significantly improves upon the SORT method. Their main contribution to developing OC-SORT is making it robust for tracking under occlusions and non-linear motion. Even while keeping the method simple as the baseline they achieve state-of-the-art results in some metrics at the time their paper was published. As per their research results, their method also falls short like it did in SORT when the video has low frame rates and when the object motion is fast but they still achieved great results without over-complicating the baseline and suggest that some authors have tried solving the problems mentioned above by using the center distance [30] or by adding appearance similarity [27] into their respective tracking models.

3.3 Improving Detection in Subsequent Frames Using the Previous Frame

An object detector while making its prediction on an image frame is generally unaware of the predictions it had made previously even if the previously predicted frames were part of a collection of frames from the same scene, which could be perceived as the loss of useful information.

Problem Statement.

- It was noted that since in the tracking-by-detection paradigm the tracker is dependent on the detector for all the detections in a particular frame, sometimes the detected bounding box could be missing from the final output even though that particular object was detected in the previous frame as shown in Figure 3. The reason for this particular issue was that the detector could not correctly identify the object and gave it a confidence score below the threshold, hence these detections were removed from the final output. A part of this paper explains the solution to this problem and has been explained further.
- The other issue was that more than one bounding box of different classes was sent by the detector for the same object in some frames as shown in Figure 2.

Since the detection sent by the detector are independent of the past frames sent by the same detector, it was a great opportunity to utilize the past frame information. These issues have been researched and solved using the same algorithm described in Figure 5 along with some visualizations to understand the working of the same.

Fig. 2. Same objects having two different detected classes. (Two boxes of different color shades depict two different classes).

Post Processing in YOLOv5. After the YOLO algorithm makes its predictions as shown in Figure 1, these detections go to a post-processing function for the final processing cause in the current state they are unusable. That post-processing function is responsible for filtering out multiple things:

- Unwanted detections.
- Detections that have confidence scores below the threshold.
- Overlapping/redundant detections.

Unwanted detections and detections that have confidence scores below the threshold are filtered out using a confidence threshold. Overlapping or redundant detections are filtered out using the Non-Max Suppression function. In the end, the final best detections are returned back from this function for passing to the tracker for tracking. To solve the issue of missing detections in the subsequent frames, this function had to be changed/updated using new techniques to make it aware of the various detections it is disregarding or filtering out to regain the missing detections in the frame T if the particular object was detected in the previous frame, frame T-1.

First, the candidates are selected in the function based on their object score using the confidence threshold as shown in Figure 4. Here it is made sure that most of the currently detected objects are not disregarded by setting a small object confidence threshold (1st) and removing the original object confidence check. Later in the function, object scores and class confidence scores are multiplied to get a final confidence score and are again checked against the confidence threshold (2nd), which is still required to take care of unwanted detections. Also, it is still needed to find the missing detections from these rejected values, which is done with our developed algorithm visualized in Figure 5.

The overlap between the final previous frame (T-1) detection tensor and the current frame (T) detection tensor is checked. With this overlap check, it is noted if any of the previous frame detections do not overlap (0% IoU) with the current

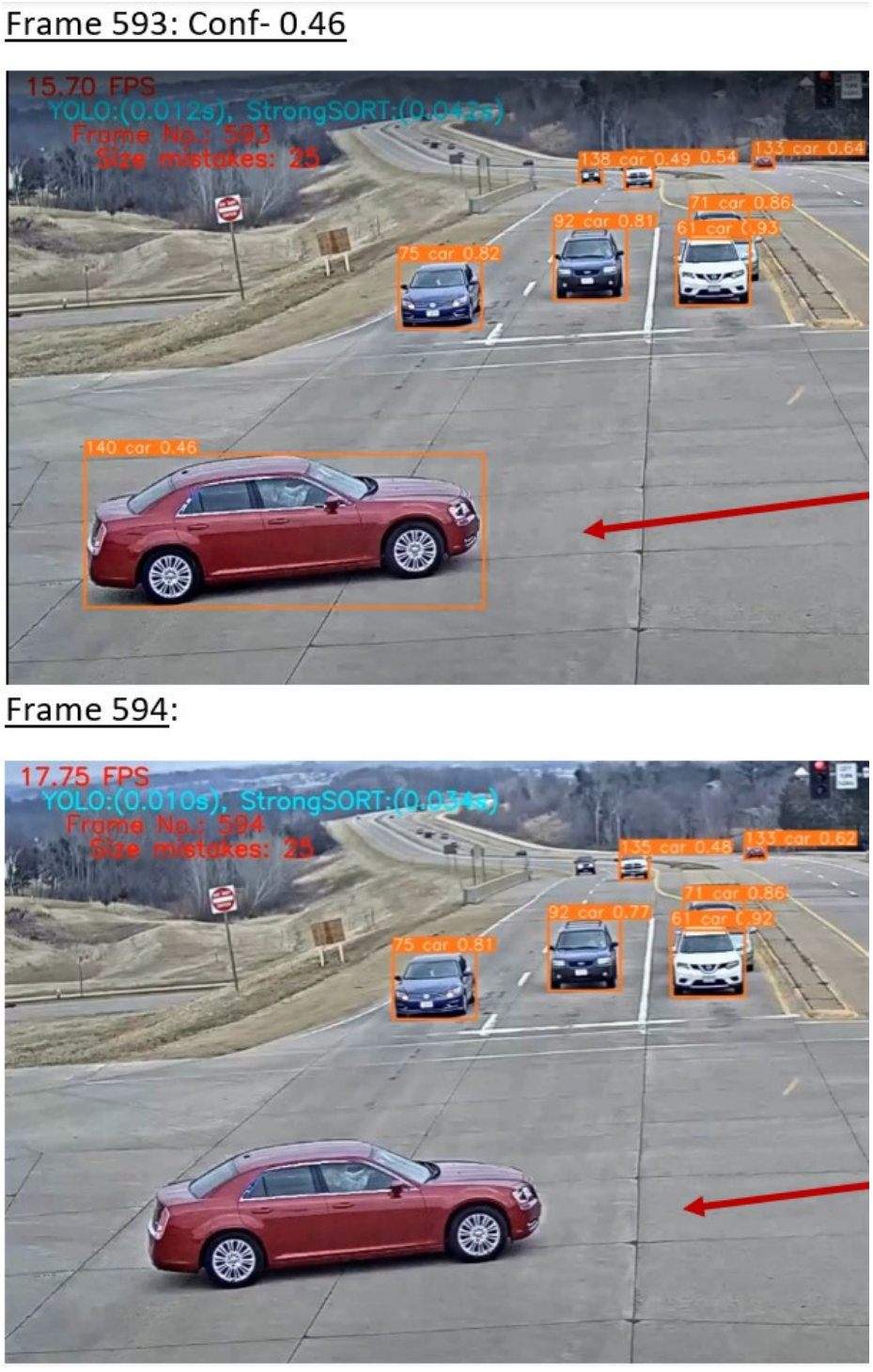

Fig. 3. Missing detection in the subsequent frame even though the object was detected in the previous frame.

Fig. 4. Candidates Selection

frame unfiltered detections and then these detections are saved in a new tensor 'N1'. Now all the rejected detections by the (2nd) threshold check in a new tensor 'LCX' are saved. After that, the overlap between tensors LCX and N1 is checked and the detections having some amount of overlap from the LCX tensor are saved in a new tensor 'N2'. Then the N2 tensor is concatenated with the unfiltered detection tensor of the current frame. In the end, this unfiltered tensor is passed through the Non-Max Suppression function which removes redundant detection values. These values are passed to the tracker and also saved as previous frame final detections for the next iteration.

By using the first overlap check information between the final previous frame detections and current frame detections the detector's mistake of passing two bounding boxes for the same object in the current frame due to them being of different classes is also corrected by using the previous frame detected class result of that object. It is also made sure not to correct this mistake while the object emerges from any side of the frame. The reason is if the object has not properly emerged into the frame the detector might make a mistake in incorrectly identifying it as something else and due to our algorithm that incorrect result might carry forward in future frames, for example, a small part of the front of the car might be identified as a motorcycle or a person which is undesired to be carry forwarded to the future frames.

One concern regarding this algorithm was the speed compromise that could arise from the use of this algorithm during the tracking task. Hence, the speed and performance are kept well preserved by the use of computation time-preserving techniques used throughout the algorithm. One such technique was the use of list comprehension instead of the use of for-loops since many of the tasks in the algorithm involved the need of creating lists/tensors, which made a huge difference in the speed performance. Also, only a single frame (previous frame) information was used for our algorithm because of the fact that as the number of objects and frame comparisons increase, the overall computation overhead is also significantly increased which ultimately decreases the speed of the tracking task which is unwanted for real-time applications. The results of the developed algorithm can be found later in this paper in the results section.

3.4 Algorithm to Evaluate Detector's Consistency in Bounding Box Size Assignment

Problem Statement. It was noted that the detector could make a mistake in keeping the bounding box size consistent in the subsequent frames. As far as it is known, no other evaluation metric captures this inconsistency of size difference in the bounding boxes. The size difference is mainly noticed when the work is done with the real-time applications of these models.

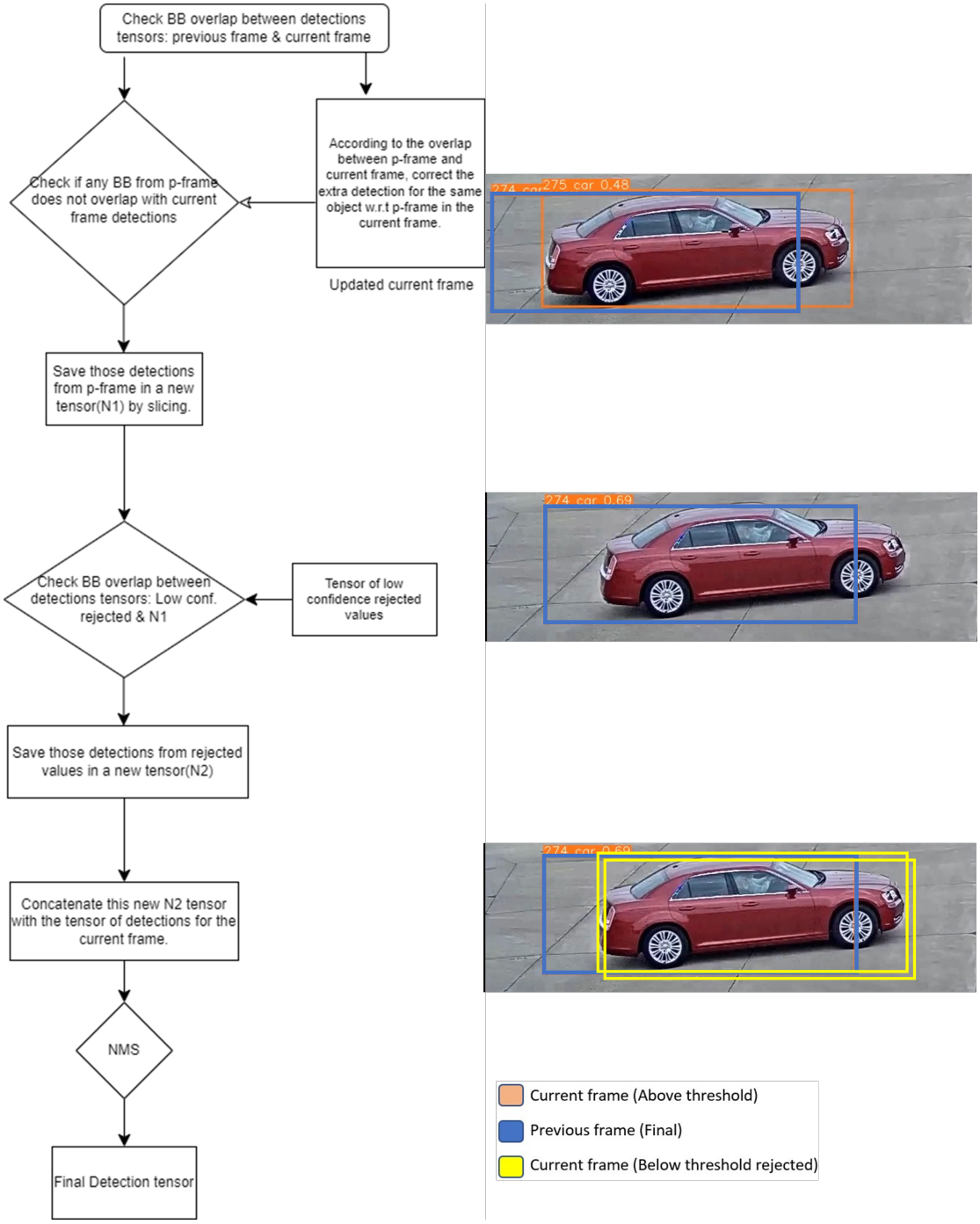

Fig. 5. Algorithm based on previous-frame to reduce the missing detection problem and to solve the multi-class detection for the same object problem.

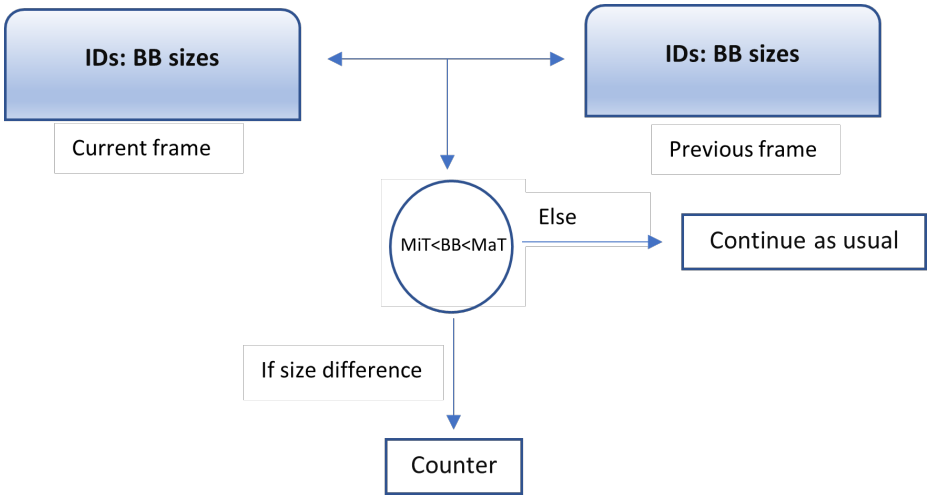

Fig. 6. Depicts the bounding box size difference identifier algorithm's working. (Note - MiT: Minimum Threshold, BB: Bounding Box, MaT: Maximum Threshold).

As already stated in the problem statement above that the reason for developing this algorithm is that it has not come to the observation if any research work has been conducted on checking how many times the detector made an inconsistency mistake while maintaining the bounding box sizes in the subsequent frames, especially in real-time situations with the help of a tracker. Currently, in this algorithm, visualized in Figure 6, the information in two frames, current frame T and previous frame T-1 are utilized. The ID information assigned and provided by the tracker for each and every detection is utilized and then the IDs and the corresponding width and height of each bounding box are stored for the comparison of frames T and T-1. A minimum and maximum threshold value are used which can be adjusted according to the scene by the user to measure the amount of size difference that occurs in subsequent frames. In the end, if the value exceeds or is beneath the set threshold then a counter counts how many times the mistake was made until the scene/video runs out in real-time. By this with the help of a tracker, a user can easily identify the performance and consistency of the detector in a real-time scenario without carrying out the evaluation of the detector separately. The result of the same can be seen later in the results section of this paper.

4 Experiments and Results

This section conveys the results from all of our developed algorithms with resulting real-time frames of a scene, tables, and various different generated data. To run and test our model, we utilize python with the deep learning framework

PyTorch and all computations are performed on a desktop with an Intel(R) Core(TM) i7-9700 CPU @ 3.00GHz and Nvidia Geforce RTX 2080 Super GPU.

4.1 Improved Tracking

Figure 7 shows the improved detection results on the same subsequent frames shown in Figure 3 after applying the algorithm proposed in this paper. In Figure 3 the detection box for the pointed object in frame 594 cannot be seen because the confidence of that object was below the threshold which was set as **0.45** and was not sent by the detector to the tracker, hence it was not displayed initially. But in Figure 7 it could be clearly seen that after applying the algorithm the detection box is visible even though the confidence of the pointed object is still below the threshold. Also, Figure 8 shows examples of other low-confidence bounding boxes during the inference after applying our developed algorithm. Figure 9 shows the improvement made using the algorithm into resolving the multi-class detections for the same object across subsequent frames as it was depicted in Figure 2. Table 2 and 3 show the improvement made by the algorithm in both MOTA/MOTA+ [23] score and True Positive matches using two different trackers and benchmark datasets. It also mentions Mostly Tracked (MT), Partially Tracked (PT), and Mostly Lost (ML) trajectory details before and after applying the algorithm. The improvements have been represented in the Tables using the green font for easy visualization of the performance boost.

Table 2. Evaluation results on MOT16 dataset.

Tracker	**MOTA**	**MOTA+**	**True Positive**	**MT**	**PT**	**ML**
StrongSORT (before)	62.374	61.271	3,515	10	12	3
StrongSORT (after)	64.040	62.936	3,586	11	12	2
OC-SORT (before)	61.080	60.263	3,437	8	14	3
OC-SORT (after)	62.431	61.271	3,510	8	14	2

MT: Mostly Tracked, PT: Partially Tracked, ML: Mostly Lost.

Table 3. Evaluation results on MOT17 dataset.

Tracker	**MOTA**	**MOTA+**	**True Positive**	**MT**	**PT**	**ML**
StrongSORT(before)	60.601	59.587	3,514	8	14	4
StrongSORT(after)	63.061	61.671	3,669	11	12	3
OC-SORT(before)	61.728	60.714	3,463	8	14	4
OC-SORT(after)	62.423	61.239	3,534	8	14	4

Since the main algorithm proposed in this paper is accommodated in the post-processing function of the YOLO model, a speed analysis of the overall tracking model was made in Table 4 with two different scenes having comparisons

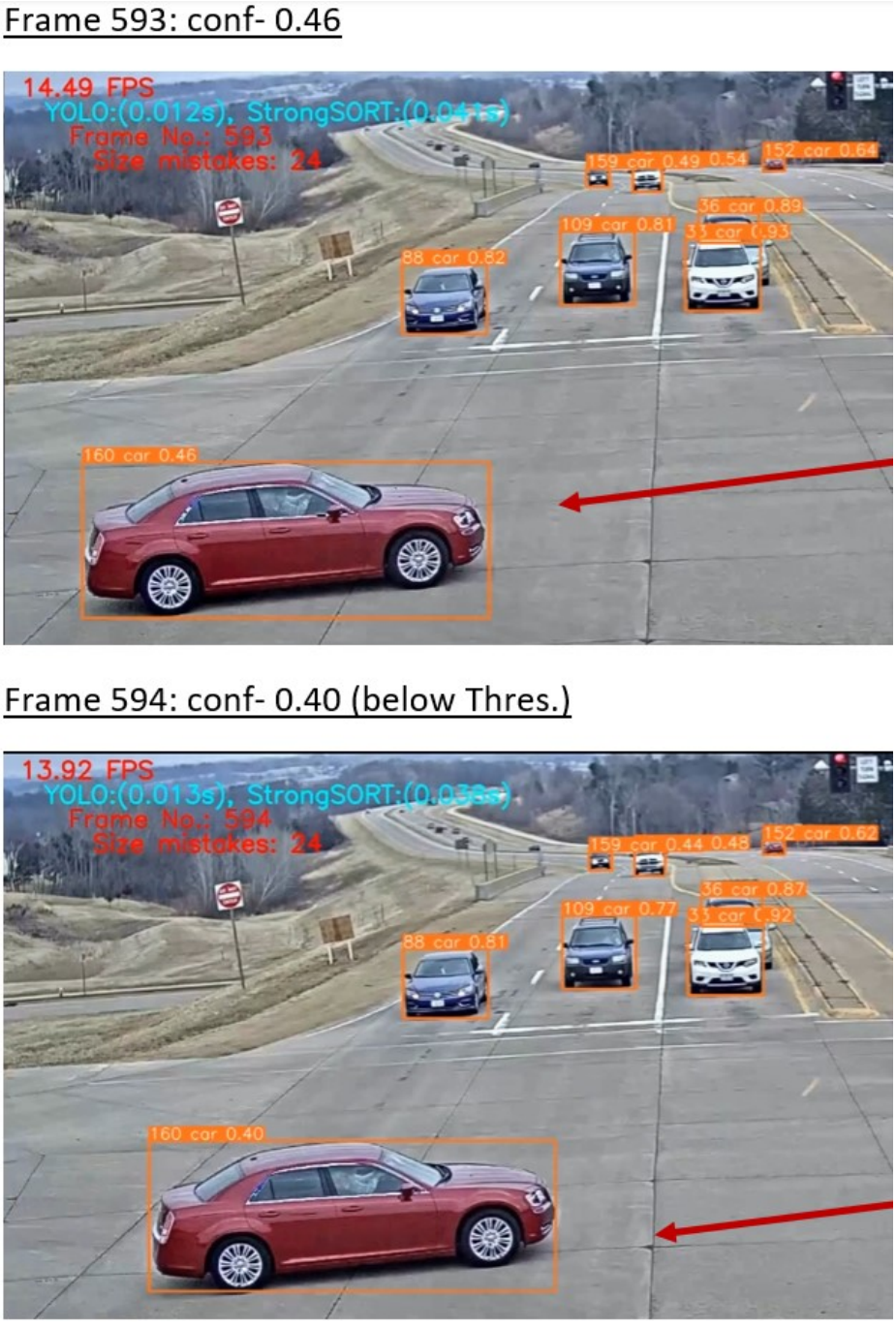

Fig. 7. Bounding boxes in subsequent frames after applying the algorithm.

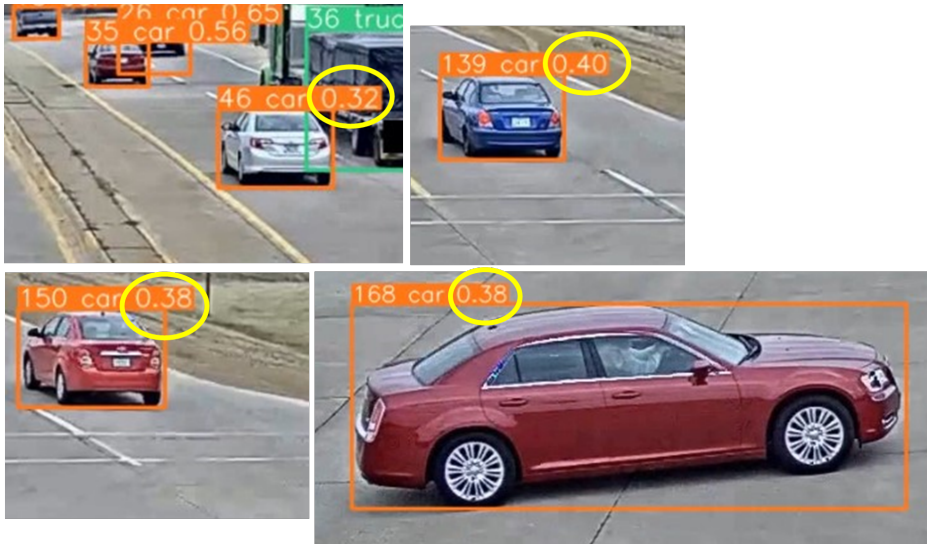

Fig. 8. Low confidence valid bounding boxes in the current frame after applying the algorithm.

Fig. 9. Multi-class detection for the same object issue improvement.

made using unoptimized and optimized versions of our algorithm showing the optimized version achieving almost the similar frames per seconds results as the base version even with added computations because of the implemented algorithm.

Table 4. Model speed analysis

Scene Type	Avg. FPS	Post Processing
Traffic (base)	10.01	2.00ms
Traffic (unoptimized)	6.19	18.07ms
Traffic (optimized)	9.13	10.10ms
Pedestrian (base)	9.10	2.00ms
Pedestrian (unoptimized)	6.20	14.21ms
Pedestrian (optimized)	8.50	9.90ms

Figure 10 shows some failed cases after using the algorithm using the same set of frames. Yellow circles represent good cases and red circles represent bad cases. In the first scene (left), two cars that were detected before applying the algorithm are not detected afterward but instead, we are able to fix other detection mistakes. In the second scene (right), the algorithm fails to detect the marked object in the figure but it can be seen that there is some bounding box size consistency improvement after applying the algorithm.

Fig. 10. Some failed cases.

4.2 Results for Bounding Box Size Difference Error Calculation

Figure 11 clearly shows that when the detector made a mistake in keeping the size of the detected object consistent in the subsequent frames our proposed algorithm worked and the mistake was recorded in the top left. Notice how the bounding box size is visibly changed in those two frames due to the misidentification.

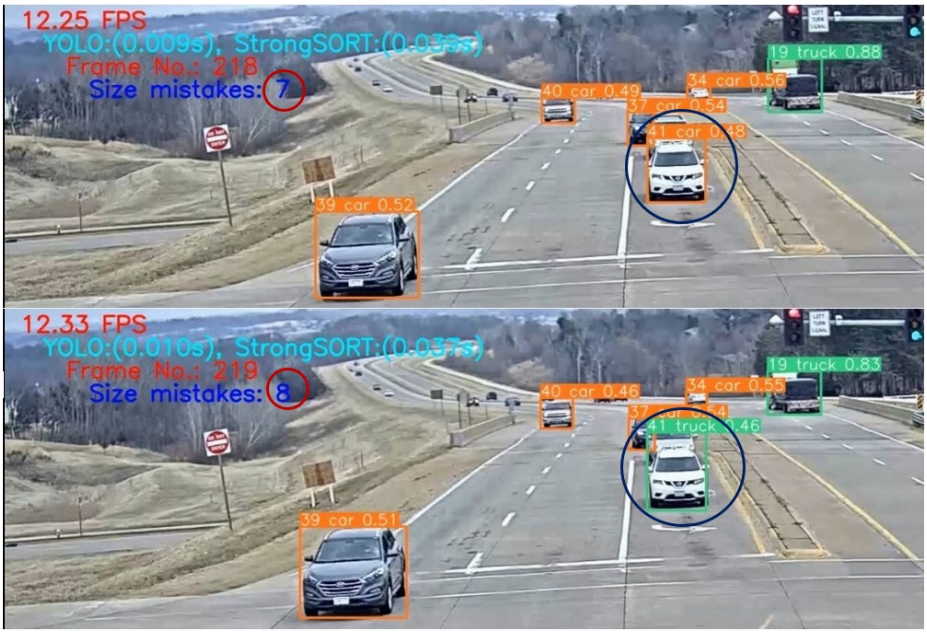

Fig. 11. Bounding box size difference inconsistency in subsequent frames is registered after applying the algorithm.

5 Conclusion

It is concluded that all the experiments were conducted sincerely while keeping the end goals in mind all the time.

- The algorithm to improve the detections using previous frame information was developed and tested to show an improvement over the base evaluation value.
- The detector was evaluated for the bounding box consistency check and the working has been shown using the subsequent frames where the detector had made a mistake and was noted down by the algorithm according to the said threshold.
- Various experiments have been conducted to show various results and findings of all the methods proposed in this paper.

Future Work. Since in this paper we have exclusively focused on YOLO detectors for improving the tracking model, in the future, we would like to work towards evolving the algorithm to incorporate various different detector models and present a qualitative and quantitative comparison with other techniques that are developed to improve the MOT.

Acknowledgements This study was supported by the BK21 FOUR project (AI-driven Convergence Software Education Research Program) funded by the Ministry of Education, School of Computer Science and Engineering, Kyungpook National University, Korea (4199990214394).

References

1. Ababsa, F., Maidi, M., Didier, J.Y., Mallem, M.: Vision-based tracking for mobile augmented reality pp. 297–326 (2008)
2. Bei, S., Zhen, Z., Wusheng, L., Liebo, D., Qin, L.: Visual object tracking challenges revisited: Vot vs. otb. Plos one **13**(9), e0203188 (2018)
3. Bernardin, K., Stiefelhagen, R.: Evaluating multiple object tracking performance: the clear mot metrics. EURASIP Journal on Image and Video Processing **2008**, 1–10 (2008)
4. Bewley, A., Ge, Z., Ott, L., Ramos, F., Upcroft, B.: Simple online and realtime tracking pp. 3464–3468 (2016)
5. Bochinski, E., Eiselein, V., Sikora, T.: High-speed tracking-by-detection without using image information pp. 1–6 (2017)
6. Bochkovskiy, A., Wang, C.Y., Liao, H.Y.M.: Yolov4: Optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934 (2020)
7. Bouget, D., Allan, M., Stoyanov, D., Jannin, P.: Vision-based and marker-less surgical tool detection and tracking: a review of the literature. Medical image analysis **35**, 633–654 (2017)
8. Cao, J., Weng, X., Khirodkar, R., Pang, J., Kitani, K.: Observation-centric sort: Rethinking sort for robust multi-object tracking. arXiv preprint arXiv:2203.14360 (2022)
9. Du, Y., Song, Y., Yang, B., Zhao, Y.: Strongsort: Make deepsort great again. arXiv preprint arXiv:2202.13514 (2022)
10. Du, Y., Wan, J., Zhao, Y., Zhang, B., Tong, Z., Dong, J.: Giaotracker: A comprehensive framework for mcmot with global information and optimizing strategies in visdrone 2021 pp. 2809–2819 (2021)
11. Evangelidis, G.D., Psarakis, E.Z.: Parametric image alignment using enhanced correlation coefficient maximization. IEEE transactions on pattern analysis and machine intelligence **30**(10), 1858–1865 (2008)
12. Gao, M., Jin, L., Jiang, Y., Guo, B.: Manifold siamese network: A novel visual tracking convnet for autonomous vehicles. IEEE Transactions on Intelligent Transportation Systems **21**(4), 1612–1623 (2019)
13. Geiger, A., Lenz, P., Stiller, C., Urtasun, R.: Vision meets robotics: The kitti dataset. The International Journal of Robotics Research **32**(11), 1231–1237 (2013)
14. Gong, X., Zhou, Y., Zhang, Y.: Siamot: An improved siamese network with online training for visual tracking. Sensors **22**(17), 6597 (2022)

15. Luo, H., Gu, Y., Liao, X., Lai, S., Jiang, W.: Bag of tricks and a strong baseline for deep person re-identification pp. 1487–1495 (2019)
16. Manafifard, M., Ebadi, H., Moghaddam, H.A.: A survey on player tracking in soccer videos. Computer Vision and Image Understanding **159**, 19–46 (2017)
17. Naiel, M.A., Ahmad, M.O., Swamy, M., Lim, J., Yang, M.H.: Online multi-object tracking via robust collaborative model and sample selection. Computer Vision and Image Understanding **154**, 94–107 (2017)
18. Pang, Z., Li, Z., Wang, N.: Simpletrack: Understanding and rethinking 3d multi-object tracking. arXiv preprint arXiv:2111.09621 (2021)
19. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection pp. 779–788 (2016)
20. Redmon, J., Farhadi, A.: Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767 (2018)
21. Robin, C., Lacroix, S.: Multi-robot target detection and tracking: taxonomy and survey. Autonomous Robots **40**(4), 729–760 (2016)
22. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: Mobilenetv2: Inverted residuals and linear bottlenecks pp. 4510–4520 (2018)
23. Tak, A.S., Kim, H., Lee, S., Jung, S.K.: Towards improving the multi-object tracking evaluation metric pp. 176–176 (2022)
24. Tak, A.S., Sultana, M., Rahman, M.M., Kim, H., Lee, S., Jung, S.K.: Visual object tracking: Datasets and related information pp. 241–244 (2022)
25. Wang, Z., Zheng, L., Liu, Y., Li, Y., Wang, S.: Towards real-time multi-object tracking pp. 107–122 (2020)
26. Weng, X., Wang, J., Held, D., Kitani, K.: 3d multi-object tracking: A baseline and new evaluation metrics pp. 10359–10366 (2020)
27. Wojke, N., Bewley, A., Paulus, D.: Simple online and realtime tracking with a deep association metric pp. 3645–3649 (2017)
28. Zhang, X., Zhou, X., Lin, M., Sun, J.: Shufflenet: An extremely efficient convolutional neural network for mobile devices pp. 6848–6856 (2018)
29. Zhou, K., Yang, Y., Cavallaro, A., Xiang, T.: Omni-scale feature learning for person re-identification pp. 3702–3712 (2019)
30. Zhou, X., Koltun, V., Krähenbühl, P.: Tracking objects as points pp. 474–490 (2020)