FieldMOT: A Field-Registered Multi-Object Tracking for Sports Videos

Supplementary Material

A. More Information about Datasets

A.1. GRF and CARLA Platforms

In this research, we utilized two advanced simulation environments, Google Research Football (GRF) [9] and CARLA [6], to generate comprehensive data for multi-object tracking (MOT) on broadcast video research. The source codes for both GRF and Carla are available on GitHub [1, 2], respectively. GRF provides an environment specifically designed for simulating football matches involving a fixed number of participants. Participants' movements are constrained within a defined field. A key feature of GRF is its flexibility in configuring camera positions and angles, enabling it to simulate tracking from various perspectives and conditions. Its gaming strategy is based on reinforcement learning. It is highly beneficial for conducting synchronized recordings from multiple angles to generate broadcast videos.

In contrast, CARLA offers a more open and enriched environment for simulating pedestrians' movements across urban settings. Quite different from GRF, CARLA allows for far greater flexibility in terms of the number of participants and the scale of the environment, making it more suitable for MOT studies in more complex, less structured scenarios. The open environment and unpredictable number of pedestrians introduce additional challenges in tracking multiple objects compared to GRF.

(a) GRF snapshot (original) (b) GRF snapshot (modified) (c) CARLA snapshot

Figure 1. Snapshots of football match and street scene datasets. In the GRF snapshot, the minimap was removed and the score bar was preserved (our code will be made available on GitHub upon publication).

A.2. Environment Setups

Frame Modification

With the GRF environment, we aim to simulate the presentation style of football game broadcast videos. While with CARLA, we want to simulate a pedestrian monitoring system. Fig. 1 (a) shows a frame captured in the original GRF environment during a match. Note that in the bottom, there is a virtual screen (radar minimap) showing all players' distribution and their names. This plug-in sub-screen has to be removed to reflect actual recordings. We have modified the corresponding code (refer to our GitHub, where the code will be made available upon publication). However, we preserved the upper-left panel displaying the match time and scores. The resulting frame is shown in Fig. 1(b), which is close to real TV broadcasting. A CARLA snapshot is shown in Fig. 1(c), which we believe to be sufficiently similar to the appearance of a pedestrian monitoring system. So no additional adjustment is made.

Experiment Fields and Keypoints Definitions

In football player tracking, it is intuitive to consider the entire field as the observation area. For keypoints detection, since the football field was set to the standard size of $110m \times 72m$, we mark a keypoint every $11m$ along the length and every $12m$ along the width, which results in a grid of 77 points, as Fig. 2 (a) shows.

The map that we used in CARLA was the default map "TOWN 10" from version 0.9.15. However, this map was too large for few cameras to cover. Besides, since our target was to simulate pedestrians' movements, we wanted to create a scenario where individuals could enter and exit the observation area. Therefore, we chose a portion of the map, specifically the upper-left corner, as the observation area, with dimensions of $108m \times 40m$ as described in the main text. The map is illustrated in Fig. 2 (b) and (c). we mark a keypoint every $12m$ along the length and every $4m$ along the width, resulting in 77 points like in the football field. This number, 77, is not a strict requirement but was chosen for consistency in the output of the keypoint detection model across experiments. Fig. 2 (b) shows the keypoints distribution in the selected field, while Fig. 2 (c) shows the whole map. Note that whenever a person exited the observation area and later re-entered the area, it was considered as a new person.

(a) Football Field in GRF (b) Observation field in Town 10 (c) Town 10 in CARLA

Figure 2. Setups of keypoints in the football field and the TOWN 10 map.

Object Detection

Here we describe how we obtained object bounding boxes as our detections. We took MOT20 [5] training set to train YOLOX [7], and it performed well on our football data. Therefore we decided to employ this MOT20-trained YOLOX as the object detector for each tracker in the football scenario.

Unfortunately, the data collected from the CARLA simulator exhibited severe noise around pedestrians. As a result, the detection performance of YOLOX was unsatisfactory. However, CARLA offers an API that outputs ground truth 3D bounding boxes. To fairly compare different trackers and eliminate the influence of poor detector performance, we decided to utilize this API to retrieve all pedestrians as the detection results. This offers a fair baseline for all trackers. Also, without the errors of detections, one may observe the potential loss of a tracker. The steps to retrieve 3D object bounding boxes in CARLA are as below:

(1) Deploy a depth camera at the position of each RGB camera.
- The depth camera's position and direction are always identical to the RGB camera.

(2) For every RGB image captured:
  (a) Retrieve the following from the API:
  - 3D coordinates and 3D bounding boxes of all pedestrians.
  - Camera projection matrix.

  (b) Simultaneously, capture a depth map from the depth camera.

(3) Filter out objects not in FOV:
  (a) Use the projection matrix to project all ground truth 3D bounding boxes:
  - Check if the maximal 2D bounding box is outside the frame (target out of range).
  - Check if the maximal 2D bounding box area is less than 10 pixels (target too small or too far).

  (b) Compare the actual distance between the camera and pedestrians using the depth map:
  - Check if the pedestrian is occluded.

(4) Use the remaining 2D bounding boxes as object detection results.
- Input these results into the trackers.

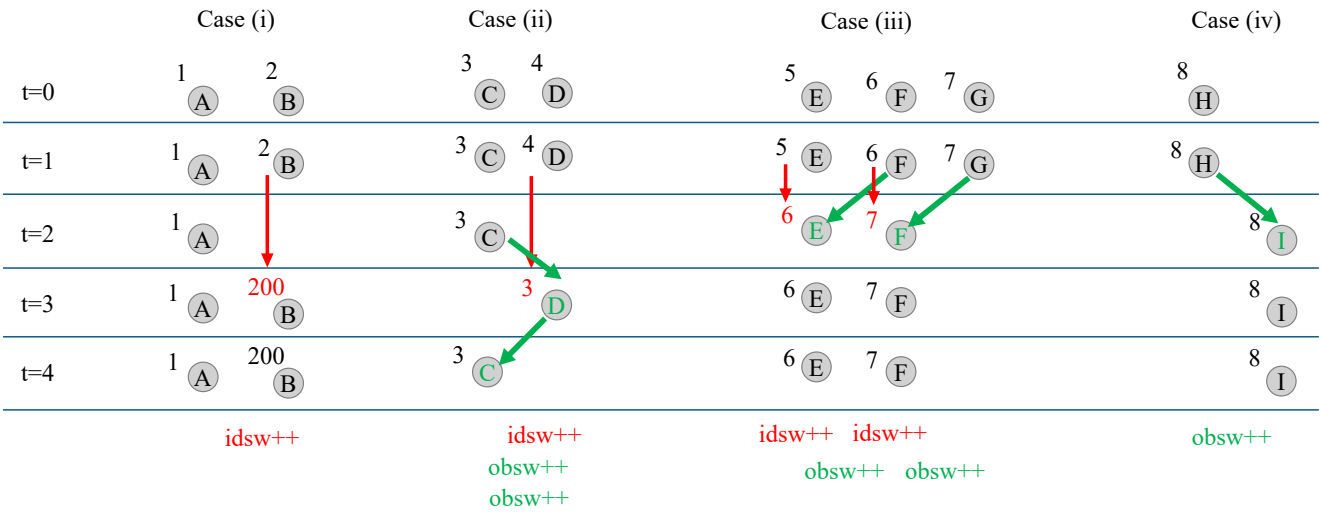

Figure 3. Examples of IDSW and OBSW.

B. More Discussions of ID Errors

B.1. Observations

In our knowledge, the most commonly used metrics for evaluating a MOT tracker's performance is *object-centric*. The track of an object is always continuous. By object-centric, the main concern focuses on ID consistency of each object, which is commonly reflected by the IDSW (identity switching) metric. However, our MOT scenarios are quite different. There are camera switching events occurring in a specific monitoring field. In this case, a problem that was overlooked previously is the incorrect use of IDs. To address this issue, we highlight the key difference between the following two error scenarios:

- IDSW: When the same object switches between two *IDs*, it is called an ID switch.
- OBSW: When the same ID switches between two *objects*, it is called an object switch.

While IDSW quantifies the number of times that an object changes its ID, OBSW measures the number of times that an ID changes its associated object.

Fig. 3 illustrates several erroneous cases, where letters represent ground truth object IDs and numbers represent their predicted IDs. In case (i), there is an IDSW for object B. In case (ii), the predicted ID for object D switches from 4 to 3, resulting in an IDSW. However, for ID 3, it is assigned first to object C, then to object D, and then back to C later, resulting in two OBSWs. In case (iii), there are two IDSWs and two OBSWs. In case (iv), an old ID 8 for object H is switched to a newborn object I, resulting in an OBSW. The commonly used metrics, like MOTA, HOTA, and IDF1, do not consider the OBSW erroneous scenarios. In situations where camera-switching events happen frequently, OBSW deserves more attention.

Table 1. Comparisons on OBSW and IDSW.

	OC-SORT [4]	Deep OC-SORT [10]	BoT-SORT [3]	Deep-EIoU [8]	FieldMOT
Football Match Dataset					
OBSW ↓	16246	7103	3419	17104	**3177**
IDSW ↓	38523	23612	17879	24063	**10540**
IDF1 ↑	0.1841	0.1889	0.1754	0.2193	**0.4823**
Street Scene Dataset					
OBSW ↓	14766	2777	3452	14448	**2367**
IDSW ↓	29496	7910	20972	17541	**5356**
IDF1 ↑	0.3980	0.5278	0.4957	0.4126	**0.5986**

B.2. IDSW and OBSW Evaluations

It is clear that considering both OBSW and IDSW can provide a more comprehensive evaluation on trackers. We show our evaluation results on the related metrics in Tab. 1. We observe that although the OBSW counts are relatively less than the IDSW counts, they should not be ignored because the portion $\frac{OBSW}{IDSW+OBSW} = 14.1 \sim 45.2\%$ is not negligible. Our method outperformed the other methods not only in IDSW, showing its ability in maintaining ID consistency, but also in OBSW, showing its stability in object consistency. For Deep-EIoU, although it received a relatively high IDF1 score on the football match dataset, its OBSW and IDSW were high as well, reflecting that its high IDF1 came with frequent ID switches and, moreover, many of these switches were between old IDs instead of switching to new ones. On the other hand, Deep OC-SORT and BoT-SORT had less OBSW, showing that most of the ID errors came with creating new IDs.

For the street scene dataset, we observed that the IDF1 score of Deep-EIoU was relatively low. Considering the openness of the environment, excessively retaining old IDs naturally increased the overall ID error rate. In contrast, BoT-SORT exhibited higher IDSW score, but its IDF1 score significantly surpassed that of Deep-EIoU. By comparing the OBSW scores of them, we observed that although BoT-SORT tended to establish new IDs more frequently, it achieved better results in an openness environment.

From the above analysis, we noticed that Deep-EIoU tended to keep old IDs, resulting in high OBSW. Instead, BoT-SORT tended to create new IDs. Although both approaches were reasonable under their specific situations, measuring their OBSW scores gave more insights into a tracker's tracking ability.

References

[1] CARLA Simulator. Available at https://github.com/carla-simulator/carla. 1

[2] GRF Data Generator. Available at https://github.com/skimslozo/3dv. 1

[3] Nir Aharon, Roy Orfaig, and Ben-Zion Bobrovsky. BoT-SORT: Robust Associations Multi-Pedestrian Tracking. *arXiv preprint arXiv:2206.14651*, 2022. 3

[4] Jinkun Cao, Jiangmiao Pang, Xinshuo Weng, Rawal Khirodkar, and Kris Kitani. Observation-Centric SORT: Rethinking SORT for Robust Multi-Object Tracking. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9686–9696, 2023. 3

[5] Patrick Dendorfer, Hamid Rezatofighi, Anton Milan, Javen Shi, Daniel Cremers, Ian Reid, Stefan Roth, Konrad Schindler, and Laura Leal-Taixé. MOT20: A benchmark for multi object tracking in crowded scenes. *arXiv preprint arXiv:2003.09003*, 2020. 2

[6] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An Open Urban Driving Simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017. 1

[7] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. YOLOX: Exceeding YOLO Series in 2021. *arXiv preprint arXiv:2107.08430*, 2021. 2

[8] Hsiang-Wei Huang, Cheng-Yen Yang, Jiacheng Sun, Pyong-Kun Kim, Kwang-Ju Kim, Kyoungoh Lee, Chung-I Huang, and Jenq-Neng Hwang. Iterative Scale-Up ExpansionIoU and Deep Features Association for Multi-Object Tracking in Sports. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 163–172, 2024. 3

[9] Karol Kurach, Anton Raichuk, Piotr Stańczyk, Michał Zając, Olivier Bachem, Lasse Espeholt, Carlos Riquelme, Damien Vincent, Marcin Michalski, Olivier Bousquet, et al. Google Research Football: A Novel Reinforcement Learning Environment. In *AAAI Conference on Artificial Intelligence*, pages 4501–4510, 2020. 1

[10] Gerard Maggiolino, Adnan Ahmad, Jinkun Cao, and Kris Kitani. Deep OC-SORT: Multi-Pedestrian Tracking by Adaptive Re-Identification. In *2023 IEEE International Conference on Image Processing (ICIP)*, pages 3025–3029, 2023. 3