Review of the paper: StrongSORT, make DeepSORT great again - Application to 5-a-side football players tracking

Akedjou ADJILEYE
Master MVA, ENS Paris-Saclay, 15 Jan 2024
akedjou-achraff-brad.adjileye@universite-paris-saclay.fr

Abstract

Multi-object tracking (MOT) is one of the computer vision tasks that finds applications in many domains such as automatic surveillance, autonomous driving, sports, etc. In this report, We make a review of a family of tracking-by-detection algorithms: SORT, DeepSORT, and StrongSORT. These algorithms represented breakthroughs in MOT at the time of their release. Finally, we present how we use these algorithms to track 5-a-side football players in order to calculate performance metrics using the generated tracking data.

1. Introduction

Tracking by detection algorithms has become the most common approach for Multi-Object Tracking (MOT), taking advantage of the emergence of highly efficient detection models such as YOLOX [1]. SORT [2], which stands for Simple Online and Realtime Tracking, is one of the earliest algorithms to adopt this paradigm and has served as the foundation for many methods introduced in the literature later on. In this work, we focus on DeepSORT and StrongSORT.

2. Methods description

As the "S" in **SORT**, the idea behind the method is simple: detect objects in each frame and associate them from one frame to another to build short object trajectories called tracklets. To achieve this, SORT uses three modules:

- **Estimation Module:** At frame n, we assume that we predict multiple targets (bounding boxes) denoted as $(t_i)_i$ using detected objects bounding boxes in frame $n-1$. For each target i, the goal is to update the predicted bounding box by associating it with a detected object in the frame using a well-defined association criterion. If, for any target, no object is associated, a new bounding box is predicted using a linear constant velocity model, solved optimally via a Kalman filter framework (see [2], page 3).
- **Association Module:** This module is based on a distance matrix between the predicted targets $(t_i)_i$ and the detected objects. The association distance used is the intersection over union (IoU) with a minimum admissible association threshold.
- **Tracklets IDs Management:** For each new detected object that has not been associated with an existing target, a new ID is assigned. It is then tracked across frames, and the tracking stops if no new object has been associated to it for a number T_{lost} frames. Note that SORT includes a probationary period for any new object, characterized by $T_{\text{lost}} = 0$. The goal is to avoid false alarms by requiring any new object appearing in a frame to have associations over a certain number of subsequent frames, instead of predicting bounding boxes directly in case of non association.

DeepSORT [3] is an enhanced version of SORT that primarily incorporates a new association metric between the predicted Kalman states and the detected bounding boxes. DeepSORT integrates motion and appearance information through the combination of two appropriate metrics.

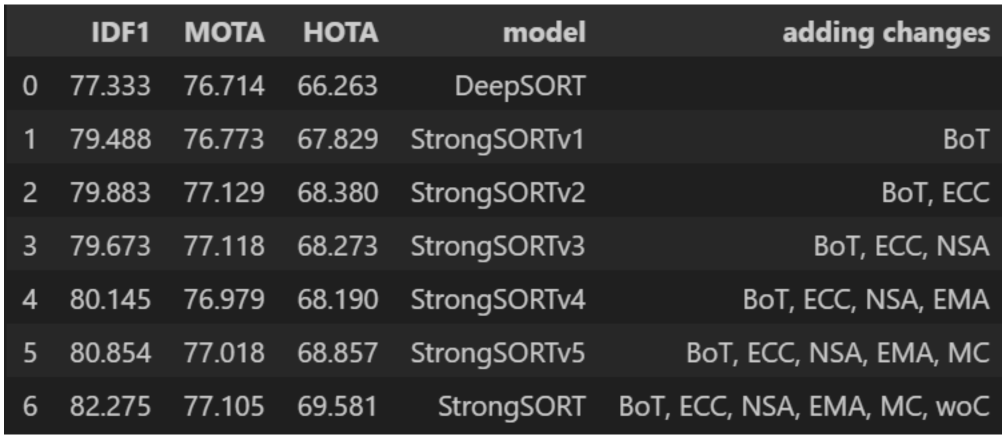
- the (squared) Mahalanobis distance between predicted Kalman states and newly arrived measurements

$$d^{(1)}(i,j) = (d_j - y_i)^T S_i^{-1}(d_j - y_i)$$

where (y_i, S_i) denotes the projection of the i-th track distribution into measurement space and d_j the j-th bounding box detection.
- For each bounding box detection d_j, an appearance descriptor r_j is computed with $||r_j|| = 1$. Keeping a gallery $R_k = \{r_k^{(i)}\}_{i=1}^{L_k}$ of the last $L_k = 100$ associated appearance descriptors for each track k, the second metric measures the smallest cosine distance between the i-th track and j-th detection in appearance space:

$$d^{(2)}(i,j) = \min\{1 - r_j^T r_k^{(i)} \mid r_k^{(i)} \in R_i\}$$

- Note that a binary variable based on thresholding was introduced on each metric to determine if an association



	IDF1	MOTA	HOTA	model	adding changes
0	77.333	76.714	66.263	DeepSORT	
1	79.488	76.773	67.829	StrongSORTv1	BoT
2	79.883	77.129	68.380	StrongSORTv2	BoT, ECC
3	79.673	77.118	68.273	StrongSORTv3	BoT, ECC, NSA
4	80.145	76.979	68.190	StrongSORTv4	BoT, ECC, NSA, EMA
5	80.854	77.018	68.857	StrongSORTv5	BoT, ECC, NSA, EMA, MC
6	82.275	77.105	69.581	StrongSORT	BoT, ECC, NSA, EMA, MC, woC

Figure 1. Reproduced ablations studies for strongSORT on the MOT17 validation set, stronger feature extractor (BoT), camera motion compensation (ECC), NSA Kalman filter (NSA), EMA feature updating mechanism (EMA), matching with motion cost (MC) and abandoning matching cascade (woC); DeepSORT serves as a baseline.

is admissible. To build the association problem, the two metrics are combined in a weighted sum:

$$c_{i,j} = \lambda d^{(1)}(i,j) + (1-\lambda)d^{(2)}(i,j) \qquad (1)$$

where an association is admissible if it is within the gating region of the two decision binary variables.

It's worth to note that DeepSORT also introduce a matching cascade algorithm (see [3], page 3) to replace the vanilla matching used in SORT.

StrongSORT [4] is an enhanced version of DeepSORT that introduces stronger changes and a lot of tricks to improve tracking.

- **Advanced Modules:** The detector used in SORT and DeepSORT is the Faster Region CNN (FRCNN) [5]. It is replaced by YOLOX-X in StrongSORT. Similarly, the appearance feature model, a simple CNN, used in DeepSORT is replaced by a stronger appearance feature, **BoT** [6].
- **EMA:** StrongSORT's authors claims that the feature gallery mechanism in DeepSORT is sensitive to detection noise. They then replace it with the feature updating strategy proposed in [7], which updates the appearance state e_{t_i} for the i-th tracklet at frame t in an exponential moving average (EMA) manner as follows:

$$e_{t_i} = \alpha e_i^{t-1} + (1-\alpha) f_i^t$$

where f_{t_i} is the appearance embedding of the current matched detection, and $\alpha = 0.9$ is a momentum term.
- **ECC:** Camera movements exist in multiple MOT benchmarks, StrongSORT uses the enhanced correlation coefficient maximization (ECC) [8] model for camera motion compensation. This technique used for parametric image alignment can estimate the global rotation and translation between adjacent frames.
- **NSA Kalman:** Instead of the vanilla Kalman filter used in SORT and DeepSORT, StrongSORT uses the NSA Kalman algorithm from [9] which is less vulnerable to low-quality detections and don't ignore the information on scales of detection noise (see [4], page 4).
- **Motion Cost and Vanilla Matching:** DeepSORT only employs the appearance feature distance as a matching cost during the first association stage with $\lambda = 0$ in eq (1), whereas motion information is include in StrongSORT association distance with $\lambda = 0.98$. Also, authors claims that as the tracker becomes stronger, the matching cascade algorithm limits the performance. They then replace it with the vanilla global linear assignment used in SORT.

3. Experiments

The authors of StrongSORT conducted an ablation study to assess the cumulative impact of changes made to DeepSORT in StrongSORT. This experiment was performed on the well-known MOT17 dataset [10], available on the official MOT Challenge website. MOT17 is a popular dataset for MOT, consisting of 7 sequences and 5,316 frames for training, and 7 sequences and 5,919 frames for testing. The first half of each sequence in the MOT17 training set was used for training, and the last half for validation.

We replicated the ablation experiments to verify the reported results in the paper. Note that the data follows the official MOT Challenge format presented in [10], providing all object detection results in .txt files, allowing us to focus solely on the tracking aspect. As StrongSORT uses the YOLOX model, which is different from the detection model

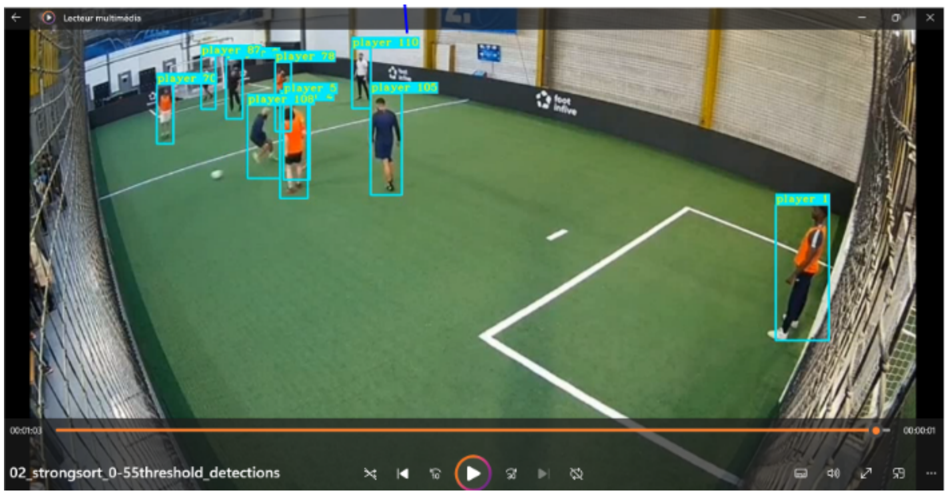
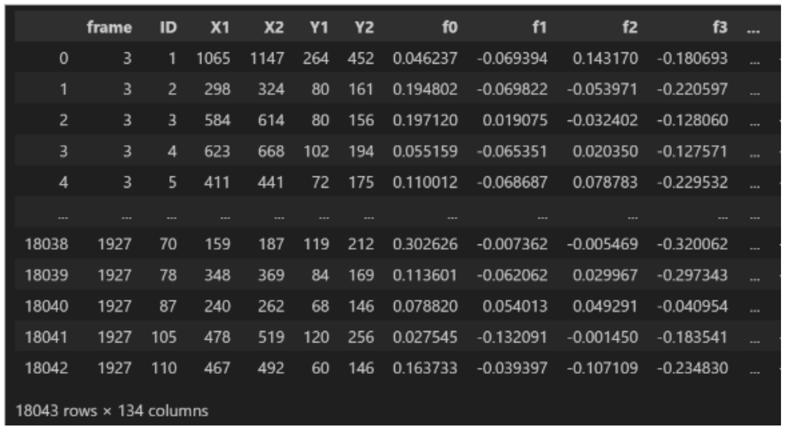


Figure 2. Output video



	frame	ID	X1	X2	Y1	Y2	f0	f1	f2	f3	...
0	3	1	1065	1147	264	452	0.046237	-0.069394	0.143170	-0.180693	...
1	3	2	298	324	80	161	0.194802	-0.069822	-0.053971	-0.220597	...
2	3	3	584	614	80	156	0.197120	0.019075	-0.032402	-0.128060	...
3	3	4	623	668	102	194	0.055159	-0.065351	0.020350	-0.127571	...
4	3	5	411	441	72	175	0.110012	-0.068687	0.078783	-0.229532	...
...	...	...	...	...	...	...	...	...	...	...	...
18038	1927	70	159	187	119	212	0.302626	-0.007362	-0.005469	-0.320062	...
18039	1927	78	348	369	84	169	0.113601	-0.062062	0.029967	-0.297343	...
18040	1927	87	240	262	68	146	0.078820	0.054013	0.049291	-0.040954	...
18041	1927	105	478	519	120	256	0.027545	-0.132091	-0.001450	-0.183541	...
18042	1927	110	467	492	60	146	0.163733	-0.039397	-0.107109	-0.234830	...

18043 rows × 134 columns

Figure 3. output tracking data: each row represent a detected object; for columns, frame contains frames index, ID contains tracklets IDs, X1, X2, Y1, Y2 contains the object bounding box coordinates in pixels and f0 to f128 represents the appearance descriptor of the detected object.

used in the MOT17 Challenge, the authors of StrongSORT provided new detection files in the same format as the MOT Challenge, available on a drive folder. The detection model used was trained on the CrowdHuman dataset [11] on the MOT17 half training set. The appearance feature extractor, BoT, was pretrained on DukeMTMC [12].

For **implementation**, we used the official repository of StrongSORT with the default parameters presented in [4].

For **evaluation**, we considered three well-known metrics for evaluating the multi-object tracking task: the identification F1-score **(IDF1)**, the multi-object tracking accuracy **(MOTA)**, and the higher-order metric for evaluating multi-object tracking **(HOTA)**. We used the popular TrackEval codebase repository for metrics calculation.

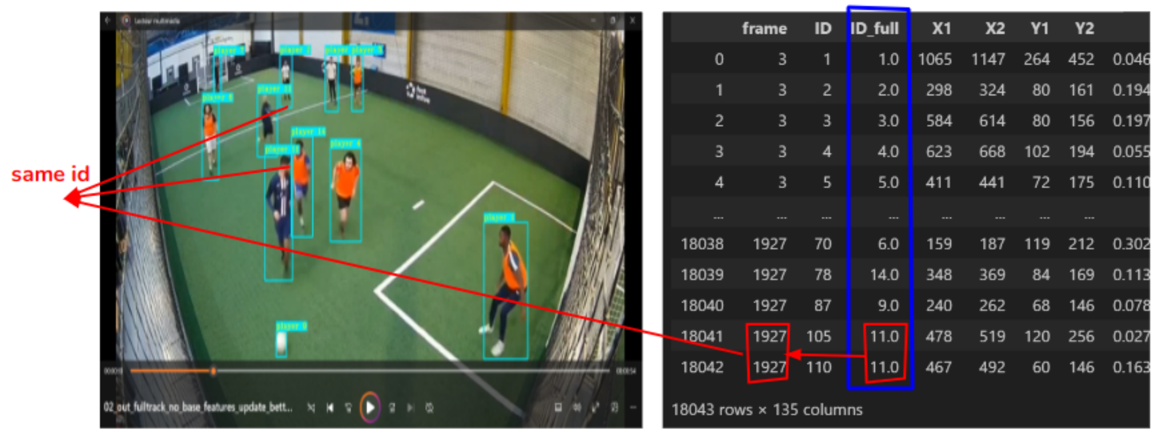
Figure 1 shows the results we obtained while trying to reproduce the reported ablation results in [4] (page 7, TABLE I). We got identical results to the authors. Upon analyzing the results, progressive performance gains are observed compared to the baseline by cumulatively incorporating changes in StrongSORT. Notably, StrongSORTv6 outperforms DeepSORT on all metrics of interest, with +4.9 on IDF1, +0.4 on MOTA, and +3.3 on HOTA.

4. Five-a-side football players tracking

4.1. Experiments

The experiments were conducted on a 64-second video of a 5-a-side football game with friends, captured at a speed of 30 frames per second. Both DeepSORT and StrongSORT trackers were tested on the video with the following specifications:

- For the detector, we utilized YOLOv5n fine-tuned on a small dataset specific to the domain, featuring two classes: player and ball. The fine-tuning process is beyond the scope



	frame	ID	ID_full	X1	X2	Y1	Y2	
0	3	1	1.0	1065	1147	264	452	0.046
1	3	2	2.0	298	324	80	161	0.194
2	3	3	3.0	584	614	80	156	0.197
3	3	4	4.0	623	668	102	194	0.055
4	3	5	5.0	411	441	72	175	0.110
...	...	...	...	...	...	...	...	
18038	1927	70	6.0	159	187	119	212	0.302
18039	1927	78	14.0	348	369	84	169	0.113
18040	1927	87	9.0	240	262	68	146	0.078
18041	1927	105	11.0	478	519	120	256	0.027
18042	1927	110	11.0	467	492	60	146	0.163

18043 rows × 135 columns

Figure 4. Illustration of IDs unicity per frame losing after tracklets merging

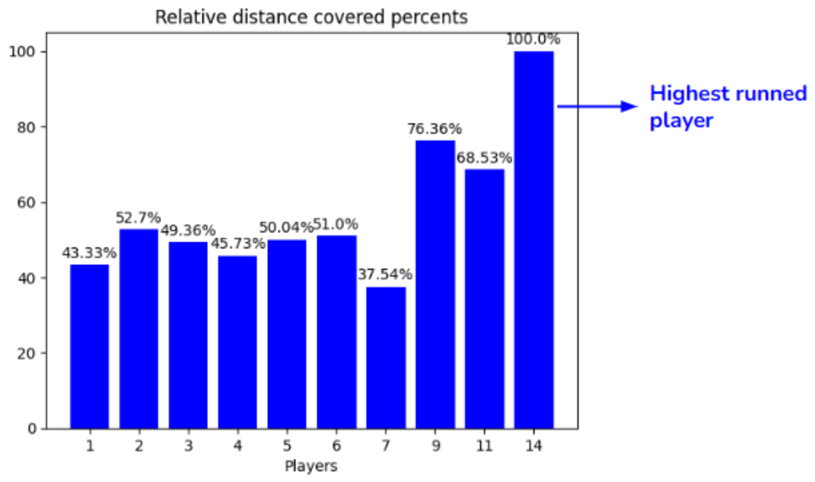


Figure 5. Players relative total distance run

of this project, and our focus was solely on player detection. The appearance descriptor used is the same as presented in [3]. We employed this checkpoint pretrained on MARS [12]. These choices were primarily motivated by simplicity and limited computational resources.
- The code was developed based on this repository, and we made all necessary changes to adapt it to the 5vs5 football context. Additionally, we replaced all the DeepSORT code with StrongSORT, utilizing the original StrongSORT repository.
- For StrongSORT tricks, we considered only EMA, NSA Kalman, Motion Cost, and Vanilla Matching. ECC was not considered since the input video camera is static.

4.2. Results

The tracking process returns two objects: the output video showing the tracked players with their IDs (figure 2) and the tracking data table (figure 3). To be able to compute a simple players performance metric like the **total run distance**, we first need to reduce the number of tracklets (small track with a unique ID) given by StrongSORT. As we know that we have only ten players to track, we want to have exactly 10 tracks with a unique ID associated to each of the ten players across all the frames. The ID column contains 37 unique IDs and we need to merge them to get a new ID column with 10 unique IDs. We use the following algorithm to merge tracklets:
- first, for each tracklet, we compute an appearance descriptor by taking the mean of all appearance descriptors of its bounding boxes;
- We initialize 10 base tracks by the first 10 tracklets of the 37 ordered by frames numbers. We make a visual verification that each of the 10 tracklets represent a unique player on the output video;
- We loop on the 27 remained tracklets: for each of them, we calculate the cosine similarity on appearance descrip-

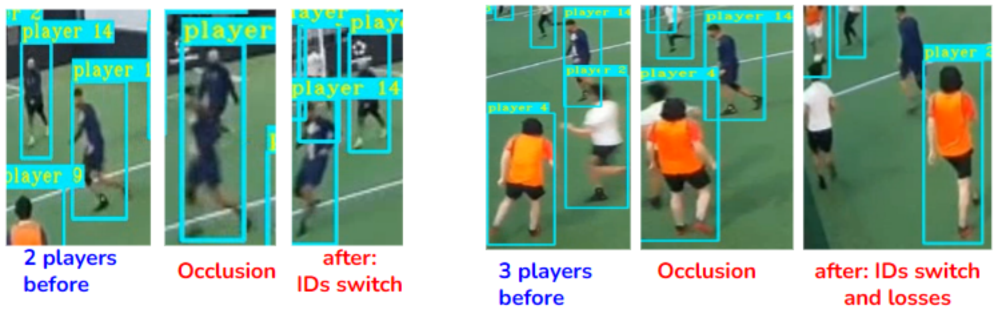


Figure 6. Occlusion illustrations

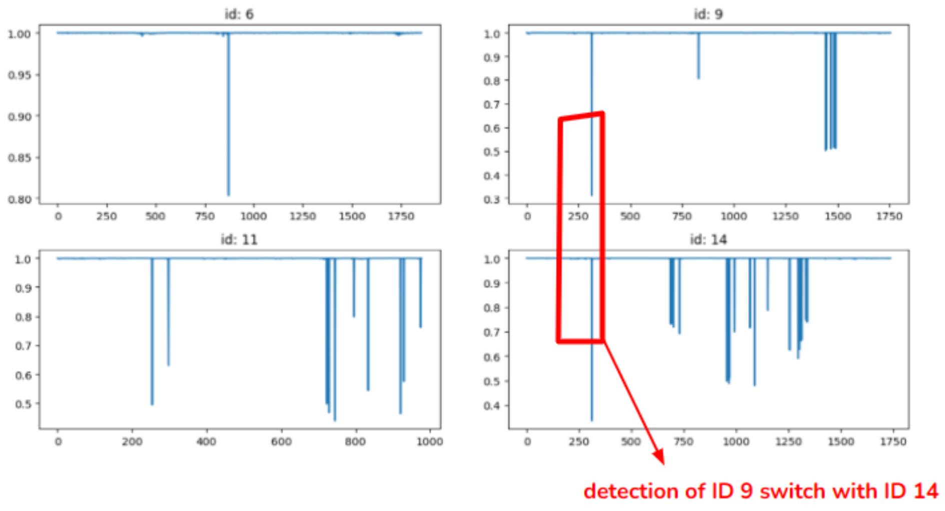


Figure 7. Occlusions detection

tors with the 10 base tracks and we merge the tracklet to only one of the 10 tracks using a maximum cosine similarity criterion and if the new base track length will not exceed the total number of frames in the video; if it will exceed, we don't merge and we remove the track from the 10 base tracks.
- We update the remained base tracks lengths.

After that process, we have a new column ID_full in the tracking data table with 10 unique IDs across all frames, but we need to fix one more problem to be able to compute the total distance run. On a frame, two distinct objects cannot have the same ID. This is an intrinsic condition maintained by a tracking algorithm, but when we merge tracklets, this condition is lost on certains frames (figure 4). To address this, we employed a simple method, which involves selecting the bounding box most suitable for the base track among those with the same ID and removing the other bounding boxes. To achieve this, we compare the feature descriptors of each bounding box with the same ID to the feature descriptor of the tracklet that served as initialization for that ID's track and we use again the maximum cosine similarity criterion.

After this final step, we obtain the desired tracking data with 10 players tracked in all frames, a maximum of 10 players per frame with a unique ID for each player detected per frame. By using the center of a bounding box as the position of the detected player, we can determine the distance run by the player by summing all distances between his positions in two consecutive frames. However, this distance is in pixel, and since we do not perform homography estimation with a 2D pitch template with lengths in meters, we have instead determined the distances relative to the player who has run the farthest by normalizing them based on the maximum distance among the 10 players (see Figure 5).

4.3. Players tracking challenges and limitations

- We have successfully obtained 10 unique IDs to track the 10 players across all frames; however, a remaining issue persists: **occlusions**(figure 6). Even though StrongSORT is a robust tracker, precisely tracking 10 players over long sequences inevitably increases the risk of IDs switch. To detect them, we plot the the cosine similarity of the appearance descriptors of bounding boxes for all 10 tracks against the frames (figure 7). As each track corresponds to the same person, the similarity score should remain maximally constant, with minor fluctuations near epsilon. By observing these curves, we notice certain IDs exhibiting peaks of decrease, which serve as evidence of occlusions. However, automatically determining which player switches with another at any given moment is a challenging task that will require future work. Also the absence of ground truth data for the 5-a-side football player tracking task prevents the quantitative evaluation of tracking results using metrics commonly employed for MOT17. Additionally, several solutions could enhance tracking results, such as fine-tuning the Re-Identification model on the specific domain, introducing a second camera perspective, incorporating player depth estimation to get players positions in 3D.

5. Discussion, Conclusion and Future works

In this project, we reviewed the SORT, DeepSORT, and StrongSORT algorithms for their application in tracking 5-a-side football players. After evaluating StrongSORT and its performance on the well-known MOT17 benchmark, we applied it to a 5-a-side football game video and developed several tricks to finally be able to estimate the total distance run by each player based on the tracking results. This project, which is part of a larger and ambitious initiative to develop a data and performance metrics tool for 5-a-side football centers using broadcast video, allowed us to delve into the core algorithm used in the tool, DeepSORT, and test the introduction of StrongSORT, a more powerful tracker. We will continue to work on improving the tracking results, as well as all other components of the tool, such as the pitch calibration module and the homography estimation module. The goal is to secure funding for the project transition into a startup by the end of 2024.

6. Computational Resources

All MOT17 tracking experiments and evaluation were run on local computer (8 CPU cores): it was quiet fast as detection data were already provided. For the football video, all codes were run on the same computer and all process take around 15 minutes : detection with yolov5n (1.9M parameters) + tracking with DeepSORT/StongSORT + computing tracking data table + creating output video).

7. References

[1] **Ge, Z., Liu, S., Wang, F., Li, Z., Sun, J.**: Yolox: Exceeding yolo series in 2021. arXiv preprint arXiv:2107.08430.

[2] **Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, Ben Upcroft**: SORT, Simple Online and Realtime Tracking. *https://arxiv.org/abs/1602.00763*.

[3] **Nicolai Wojke, Alex Bewley, Dietrich Paulus**: Simple Online and Realtime Tracking with a Deep Association Metric, *https://arxiv.org/abs/1703.07402*.

[4] **Yunhao Du, Zhicheng Zhao, Yang Song, Yanyun Zhao, Fei Su, Tao Gong, Hongying Meng**: StrongSORT: Make DeepSORT Great Again, *https://arxiv.org/abs/2202.13514*.

[5] **S. Ren, K. He, R. Girshick, and J. Sun**, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in *Advances in Neural Information Processing Systems, 2015.*

[6] **Luo, H., Jiang, W., Gu, Y., Liu, F., Liao, X., Lai, S., Gu, J.**: A strong baseline and batch normalization neck for deep person re-identification, *IEEE Transactions on Multimedia 22(10), 2597–2609 (2019).*

[7]] **Wang, Z., Zheng, L., Liu, Y., Li, Y., Wang, S.**: Towards real-time multi-object tracking. *In: European Conference on Computer Vision. pp. 107–122. Springer (2020).*

[8] **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).*

[9] **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. *In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 2809–2819 (2021).*

[10] **Anton Milan, Laura Leal-Taixe, Ian Reid, Stefan Roth, and Konrad Schindler**: MOT16: A Benchmark for Multi-Object Tracking, *https://arxiv.org/abs/1603.00831*.

- **Shao, S., Zhao, Z., Li, B., Xiao, T., Yu, G., Zhang, X., Sun, J.**: Crowdhuman: A benchmark for detecting human in a crowd. *arXiv preprint arXiv:1805.00123 (2018).*

[11] **Ristani, E., Solera, F., Zou, R., Cucchiara, R., Tomasi, C.:** Performance measures and a data set for multitarget, multi-camera tracking. *In: European conference on computer vision. pp. 17–35. Springer (2016).*