Comparison of Multi-Object Tracking in Soccer Video

Rishit Chatterjee*
University of Illinois
Urbana-Champaign, IL, USA
rc33@illinois.edu

Adit Agarwal*
University of Illinois
Urbana-Champaign, IL, USA
adit3@illinois.edu

Evan Chen*
University of Illinois
Urbana-Champaign, IL, USA
echen48@illinois.edu

Figure 1: Single frame of soccer video with detection of players via YOLOv11 + DeepSORT

1 Introduction

Multi-object tracking (MOT) is an important task in computer vision with vast applications in sports, where it enables automatic extraction of player movement, strategy, and team performance. While modern deep learning approaches are known to achieve state-of-the-art results on standard benchmarks, tracking multiple players in team sports videos is known to be challenging given rapid motion, frequent occlusions, and the visual similarity of multiple players. In this project, we present a comparative analysis of classical computer vision techniques and modern deep learning pipelines for tracking soccer players in side-view video footage.

The primary motivation of our project is to evaluate tradeoffs between tracking accuracy, recall, and processing speed. We choose side-view soccer videos given their unique challenges: players are small in pixel height, move unpredictably, and often occlude one another. Even modern tracking systems struggle to maintain consistent identities without failed detections or identity switches. We hope to quantify these struggles for both traditional computer vision methods and state-of-the-art models in identifying players. By comparing a lightweight classical baseline against deep learning methods, we hope to confirm whether the anticipated additional computation and time yields proportional gains in accuracy.

1.1 Background and Related Work

In early MOT systems for fixed-camera footage, movement detection leveraged background subtraction to separate and detect objects. In 2004, Zivkovic introduced the enhanced Gaussian Mixture Model (GMM) approach robust to lighting changes and repetitive background motions [13]. Once objects are detected, tracking requires linking observations over time to preserve identities. To estimate where targets would appear in subsequent frames, state prediction tools like the Kalman filter [5] are traditionally used to model position and velocity. Matching said predictions with measurements becomes an assignment problem often solved using something like the Hungarian algorithm [6], which pairs candidates based on spatial proximity.

The popularity of deep neural networks replaced classical motion-based detectors with learned object representations. In 2016, Redmon et al. introduced the You Only Look Once (YOLO) architecture, a unified model for fast object detection [8]. Building on this, Bewley et al. proposed Simple Online and Realtime Tracking (SORT) in 2016, demonstrating that basic filtering and Intersection-over-Union (IoU) association could achieve high performance when paired with accurate deep learning detections [3]. To combat occlusion, Wojke et al. extended this framework with DeepSORT, integrating a deep appearance descriptor to re-identify objects based on visual features instead of motion [12]. In 2022, Aharon et al. introduced BoT-SORT, combining camera motion with Kalman state vectors to further reduce identity switches in dynamic environments [1]. Alongside the introduction of YOLO, to facilitate deployment of models on resource-constrained hardware, Sandler et al. introduced MobileNetV2, a lightweight feature extractor [10]. In our use case, MobileNet serves as an embedding architecture for DeepSORT.

To rigorously compare these methods, standardized metrics were established. Bernardin and Stiefelhagen established the CLEAR MOT metrics in 2018, proposing Multi-Object Tracking Accuracy (MOTA) and Precision (MOTP), which quantify a tracker's ability to minimize false positives, false negatives, and identity switches [2]. Building on these, in 2016, Ristani et al. introduced the ID F1 Score (IDF1), emphasizing identity preservation over the entire duration of a

*All authors contributed equally to this work.

trajectory and enabling better measure of tracking stability in longer videos [9].

2 Data

We utilize the TeamTrack dataset [11], a publicly available benchmark for sports video analysis. The data we use can be found in subfolder *soccer_side*. It has high-resolution side-view videos of soccer games captured from a fixed camera.

For evaluation, we use 10 videos from the test split of the dataset. Each video is 30 seconds long and 25 frames per second for a total of 750 frames per video. The videos are ultra-wide: 6500 by 1000 pixels. Despite the high resolution, the players are small due to camera distance. Ground truth annotations are provided as *.txt* files in the standard MOTChallenge format [7], where each line represents a single bounding box in a frame:

$$\text{frame, id, bb}_{\text{left}}, \text{bb}_{\text{top}}, \text{bb}_{\text{width}}, \text{bb}_{\text{height}}, \text{conf}, x, y, z$$

For the deep learning approaches, we considered fine-tuning the models on additional videos which we would obtain from the TeamTrack dataset's *soccer_side* training and validation subsets. However, to conserve computational resources, we elected to use models pre-trained on person recognition via Common Objects in Context (COCO) to test out-of-the-box performance without fine-tuning. A copy of all data used for this project can be found in our GitHub repository (see Appendix A.1 for details).

3 Methods

3.1 Classical Pipeline

Our classical pipeline is inspired by SORT [3], but swaps neural network detectors for classic motion separation methods. It works in two phases: finding objects through movement cues, then following them using Kalman filters. A full overview of the pipeline can be seen in Figure 2.

3.1.1 Motion-Based Object Detection. To detect players in the fixed side camera view, we apply background removal using the Gaussian Mixture Model implemented by OpenCV's MOG2 function. Starting from a video sequence, the system builds its initial background using $H = 500$ past frames, which helps reduce noise caused by small scene changes. When processing each new frame I_t, instead of direct comparison, the method estimates how likely every pixel fits into that stable background pattern. If an object's variation goes beyond 16 pixels, it gets marked as a moving object.

The raw foreground mask M_{raw} often includes fine-grained noise along with gaps inside player shapes. However, we improve it through several morphological steps using an elliptical kernel K, sized 5×5.

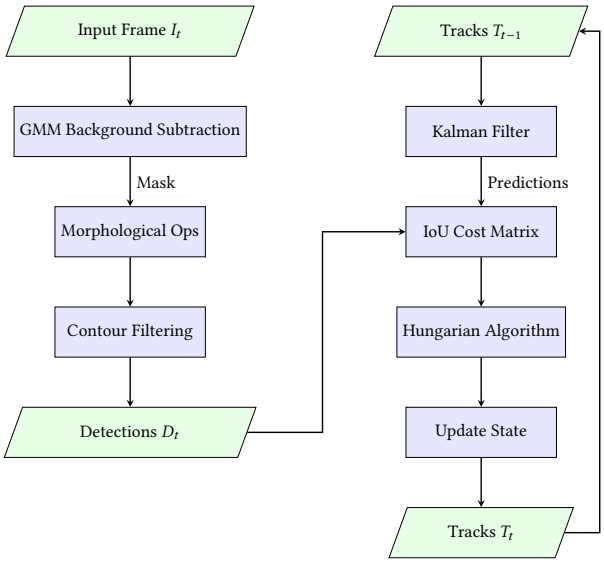

Figure 2: Classical MOT pipeline.

(1) **Opening:** $M_{open} = M_{raw} \circ K$. We perform 1 iteration of erosion followed by dilation to eliminate small noise artifacts (salt-and-pepper noise).
(2) **Closing:** $M_{final} = M_{open} \bullet K$. We perform 2 iterations of dilation followed by erosion to close gaps within the object boundaries.

Finally, contours are extracted from M_{final} and their bounding boxes are calculated. In order to limit incorrect triggers due to encoding noise or changes in brightness, we remove any proposed detection d_i if its outline covers fewer than $\tau_{area} = 300$ pixels. This leaves a group of confirmed findings $D_t = \{d_1, d_2, ..., d_n\}$ for the present image, each described by coordinates $[x, y, w, h]$.

3.1.2 State Estimation and Data Association. We monitor the detections through a simple approach. For every detected item T_k, we use a Kalman Filter that operates on an 8-element state vector $\mathbf{x}$. While this method assumes smooth motion, it adapts well to abrupt changes. Because each dimension corresponds to position or velocity in 2D plane, tracking stays efficient. Although real movements may vary, the model remains effective under typical conditions:

$$\mathbf{x} = [u, v, w, h, \dot{u}, \dot{v}, \dot{w}, \dot{h}]^T$$

Here, (u, v) represents the bounding box center, (w, h) represents the dimensions, and $(\dot{u}, \dot{v}, \dot{w}, \dot{h})$ represent their respective velocities. For each frame, the updated status of ongoing tracks is estimated with the Kalman filter. Following this, links between forecasted paths and fresh observations D_t are

formed through linear assignment. The cost matrix C is built using IoU distance: $C_{ij} = 1 - \text{IoU}(\text{Track}_i, \text{Detection}_j)$. The best match is identified with the Hungarian method [6]. We apply a threshold: any pairing with IoU < 0.3 gets discarded.

3.1.3 Track Lifecycle Management. We manage track lifecycles to combat players occluded out of view or joining mid-scene using transitions based on visibility changes:

- **Creation:** When detections don't match, tentative tracks begin. They become confirmed only if linked correctly over $N_{init} = 3$ consecutive frames.
- **Termination:** Unmatched tracks from occlusion or leaving the frame are kept temporarily. When such a track lasts $N_{age} = 25$ consecutive frames without confirmation (one full second), it gets removed entirely.

3.2 YOLOv11 + BoTSORT

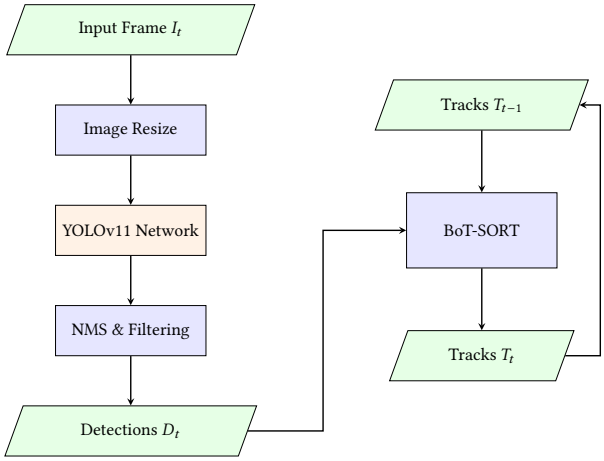

Figure 3: YOLOv11 + BoT-SORT pipeline.

To evaluate deep learning, we first combine the YOLOv11 [8] architecture for object detection with a BoT-SORT [1] tracker. A full overview of the pipeline is shown in Figure 3.

3.2.1 Object Detection with YOLOv11. For objection detection, we utilize the YOLOv11 Nano variant (`yolo11n.pt`), pre-trained on the COCO dataset, as our core detector. While the model is lightweight, the default input resolution of 640×640 pixels initially proved insufficient for our wide-angle footage (6500×1000 pixels). To address this, we set the inference input size parameter to $imgsz = 1920$ pixels. This ensures sufficient resolution for the network to resolve and detect player features. The detection logic is as follows:

(1) **Inference:** The network processes the resized frame to generate candidate bounding boxes.
(2) **Filtering:** We filter detections to retain only the "person" class (Class ID 0). To minimize false positives common in background crowd textures, we increased the confidence threshold to $\tau_{conf} = 0.25$.
(3) **Non-Maximum Suppression (NMS):** We apply NMS with an IoU threshold of $\tau_{iou} = 0.5$ to eliminate redundant overlapping boxes.

3.2.2 BoT-SORT Tracking Logic. For tracking, we employ BoT-SORT (Box Tracking with SORT), which extends the standard Kalman Filter approach. BoT-SORT augments the Kalman filter's prediction step by estimating the global camera motion (using optical flow or feature matching) between frames. The predicted bounding box coordinates are transformed by an estimated homography matrix. This compensation is intended to align track predictions with any new camera views, preventing identity breaks caused by camera panning or zooming.

3.2.3 Data Association. Unlike the classical pipeline which relies solely on spatial overlap, BoT-SORT associates detections using a fused cost matrix. For each track T_i and detection d_j, the cost combines spatial distance (IoU) and appearance similarity (Cosine distance of deep feature embeddings):

$$C_{ij} = 1 - (\lambda \cdot \text{IoU}(\hat{T}_i, d_j) + (1 - \lambda) \cdot \cos(\mathbf{f}_{T_i}, \mathbf{f}_{d_j}))$$

The appearance features $\mathbf{f}$ allow the tracker to re-identify players even after they have been occluded or separated from the track for several frames. We execute the pipeline in streaming mode on a CUDA-enabled GPU to maintain real-time performance, exporting results in the standard MOTChallenge format for evaluation.

3.3 YOLOv11 + DeepSORT

To further evaluate deep learning, we combine YOLOv11 [8] architecture for object detection with a DeepSORT [12] tracker using MobileNetV2 embeddings [10]. A full overview of the pipeline can be seen in Figure 4.

3.3.1 Objection Detection with YOLOv11. To ensure fair comparison, this pipeline utilizes the exact same YOLOv11 detection configuration as the BoT-SORT experiment (Nano model, $imgsz = 1920$, $\tau_{conf} = 0.25$). However, for DeepSORT [12], the validated detections are cropped and passed to a MobileNetV2 [10] embedder to extract appearance feature vectors and perform tracking.

3.3.2 DeepSORT Tracking Logic. The DeepSORT tracker extends the SORT framework by incorporating deep appearance features for robust re-identification. Unlike BoT-SORT which fuses appearance and motion in a single cost matrix, DeepSORT employs a cascaded matching strategy that prioritizes appearance similarity for tracks with recent observations. We implement the pipeline using the

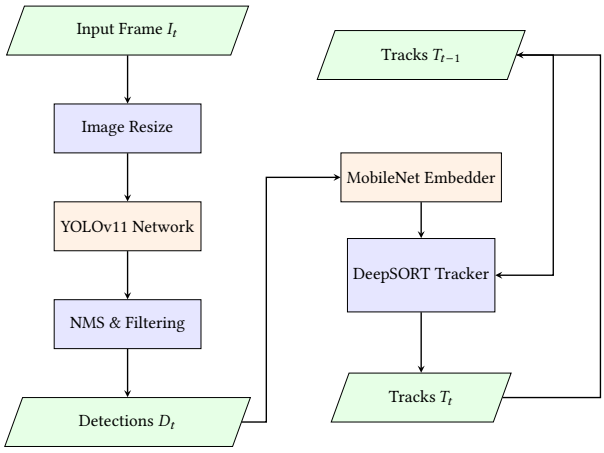

Figure 4: YOLOv11 + DeepSORT pipeline.

`deep-sort-realtime` library, which provides an optimized DeepSORT implementation with MobileNetV2 embeddings.

For each detection d_j, we crop the corresponding image region from the input frame and pass it through a MobileNet backbone pre-trained on re-identification tasks. The network produces a 128-dimensional embedding vector f_{d_j} that encodes the visual appearance of the detected player. These lightweight embeddings enable efficient similarity computation while maintaining discriminative power for distinguishing between visually similar players. DeepSORT performs data association in two stages:

(1) **Appearance Matching:** For confirmed tracks that have been recently updated, we compute the cosine distance between their stored appearance descriptors and the new detection embeddings:

$$d_{cos}(T_i, d_j) = 1 - \frac{f_{T_i}^T \cdot f_{d_j}}{\|f_{T_i}\| \|f_{d_j}\|}$$

Matches with $d_{cos} > \tau_{app} = 0.3$ are rejected. The Hungarian algorithm assigns detections to tracks based on minimum cosine distance.

(2) **IoU Matching:** Unmatched detections and tracks with stale appearance information (e.g., due to occlusion) fall back to standard IoU-based association with $\tau_{iou} = 0.7$, similar to the classical pipeline.

3.3.3 State Estimation. Each track maintains an appearance gallery storing recent feature vectors. When computing appearance distance, we use the minimum distance to any gallery sample, providing robustness against temporary appearance variations. Tracks are initialized as "tentative" and require $n_{init} = 3$ consecutive associations before confirmation. Unmatched tracks are kept for $n_{age} = 25$ frames (one second at 25 FPS) before deletion, allowing recovery from brief occlusions.

3.4 Evaluation

The primary evaluation of our three MOT pipelines is quantitative, using the `motmetrics` library in Python, a standard tool for benchmarking Multiple Object Tracking performance [4]. This library empowers us to compare our generated hypothesis files (tracking results) directly against the ground truth annotations provided by the TeamTrack [11] dataset, both of which adhere to the standard MOTChallenge format [7].

3.4.1 Quantitative Metrics. For each frame, we compare the ground truth bounding boxes and our tracker's predicted bounding boxes extracted from the output *.txt* files. Based on these frame-level comparisons, the`motmetrics` library computes the following key metrics:

- **MOTA (Multiple Object Tracking Accuracy):** combined metric of false positives, false negatives, and identity switches, normalized by the total number of ground truth objects.
- **IDF1 (ID F1 Score):** ratio of correct detections over the average number of ground truth and computed detections, emphasizing consistency of ID assignment over the entire video sequence.
- **ID Switches (IDSW):** total count of times a tracked trajectory changes its matched ground truth identity.
- **Precision and Recall:** percentage of true objects detected (recall) and percentage of detected objects that are relevant (precision).

3.4.2 Runtime Analysis. In addition to measuring performance, we seek to confirm the efficiency of each method. We log the start and end times for processing each video sequence (750 frames) to calculate the total processing duration. These timings are saved to a `metrics.json` file for each run, enabling us to compute the average runtime across the entire test set for each pipeline.

3.4.3 Qualitative Analysis. Beyond metrics, we manually watch the produced output videos from each method. To support this, each pipeline produces annotated video files for all test sequences, overlaying the predicted bounding boxes, IDs, and confidence scores onto the original footage. We visually inspect these videos to identify failures that may align with our measured metrics, including rapid ID shifts, false positives, and drift during heavy occlusion.

Table 1: Comparison of Multi-Object Tracking Method Metrics

Method	MOTA†	IDF1†	ID Switches	Precision†	Recall†	Processing Time (s)†
Classical	0.296	0.376	576	0.875	0.335	**33.79**
YOLOv11 + BoTSORT	0.248	0.301	811	0.826	0.291	112.29
YOLOv11 + DeepSORT	**0.426**	**0.466**	**396**	**0.888**	**0.475**	428.19

†*Indicates value is an average over the ten videos tested. Bold values indicate the best result for each metric.*

4 Results

MOTMetrics [2] results are summarized in Table 1, with detailed analysis below. An output video comparing one test sequence across all three methods visually can be found at https://www.youtube.com/watch?v=mITuZZlb2yQ.

4.1 Classical Pipeline

Table 2: Classical Pipeline Results per Sequence

Sequence	MOTA	IDF1	ID Switches	Precision	Recall
F_20220220_1_1890_1920	0.548	0.537	104	0.979	0.566
F_20220220_1_1920_1950	0.125	0.238	39	0.823	0.162
F_20220220_1_1680_1710	0.287	0.395	48	0.938	0.310
F_20220220_1_1770_1800	0.298	0.412	66	0.866	0.357
F_20220220_1_1950_1980	0.092	0.229	28	0.738	0.146
F_20220220_1_1830_1860	0.579	0.457	95	0.969	0.604
F_20220220_1_1740_1770	0.156	0.290	39	0.835	0.197
F_20220220_1_1860_1890	0.304	0.434	47	0.896	0.348
F_20220220_1_1800_1830	0.451	0.514	69	0.924	0.496
F_20220220_1_1710_1740	0.120	0.250	41	0.785	0.168

The classical pipeline had a modest tracking performance with an average MOTA of 0.296 and IDF1 of 0.376. Although based on no learned features whatsoever, the approach delivers competitive precision (0.875), which suggests that the MOG2 background subtraction effectively separates the moving players from the static field. However, a low recall of 0.335 indicates that motion-based detection misses stationary or slow players who blend into the background model. Further examination of individual sequence performance (Table 2) shows recall scores as low as 0.146. The pipeline produced 576 identity switches in total across all the sequences. These mostly occur when players cluster around the ball and change direction rapidly. The stop-and-go kind of motions in soccer are clearly difficult for the constant-velocity assumptions of the Kalman filter. As anticipated, the classical approach boasted great speed, requiring only 33.79 seconds on average for each 30-second video. It is the only near real-time performance we observe. Qualitative inspection of the output videos revealed some characteristic challenges: players who pause briefly are absorbed into the background model, and closely spaced players often merge into single detections due to connected components in the foreground mask. The smallest background players are often undetected.

4.2 YOLOv11 + BoT-SORT

Table 3: YOLOv11 + BoT-SORT Results per Sequence

Sequence	MOTA	IDF1	ID Switches	Precision	Recall
F_20220220_1_1890_1920	0.534	0.494	145	0.944	0.576
F_20220220_1_1920_1950	0.047	0.150	26	0.682	0.090
F_20220220_1_1680_1710	0.248	0.336	70	0.872	0.296
F_20220220_1_1770_1800	0.307	0.374	86	0.907	0.347
F_20220220_1_1950_1980	0.023	0.112	17	0.622	0.062
F_20220220_1_1830_1860	0.508	0.455	150	0.944	0.550
F_20220220_1_1740_1770	0.053	0.130	52	0.698	0.098
F_20220220_1_1860_1890	0.249	0.348	90	0.873	0.297
F_20220220_1_1800_1830	0.399	0.395	134	0.917	0.447
F_20220220_1_1710_1740	0.107	0.216	41	0.801	0.145

The average MOTA was 0.248 and IDF1 was 0.301, lowest for all methods, with the YOLOv11 + BoT-SORT pipeline. While the YOLOv11 detector provides reasonable detection, the BoT-SORT tracker accounts for camera motion compensation and seemed counterproductive for the fixed-camera footage. We theorize homography estimation adds noise, creating increased ID switches (811 in total, highest across all methods). The pipeline achieved an average precision of 0.826 and recall of 0.291, indicating that valid detections are being produced but poorly associated across frames. Processing time averaged 112.29 seconds per 30-second sequence. Further examination of individual sequence performance (Table 3) shows subpar performance across multiple sequences. Manual inspection of the annotated videos revealed significant ID flickering in players receiving new identities every few frames despite continuous visibility. This suggests that better parameter tuning of the fused IoU-appearance cost matrix for the specific characteristics present in distant, small players in wide-angle soccer footage is needed, although was not possible for our current experimentation due to compute limitations being reached.

4.3 YOLOv11 + DeepSORT

The YOLOv11 + DeepSORT pipeline achieved the best tracking accuracy among all methods, with an average MOTA of 0.426 and IDF1 of 0.466 across the ten test sequences (see

Table 4: YOLOv11 + DeepSORT Results per Sequence

Sequence	MOTA	IDF1	ID Switches	Precision	Recall
F_20220220_1_1890_1920	0.741	0.655	66	0.937	0.799
F_20220220_1_1920_1950	0.181	0.316	21	0.839	0.226
F_20220220_1_1680_1710	0.449	0.490	37	0.919	0.495
F_20220220_1_1770_1800	0.534	0.560	40	0.929	0.581
F_20220220_1_1950_1980	0.090	0.210	15	0.762	0.132
F_20220220_1_1830_1860	0.747	0.674	54	0.942	0.799
F_20220220_1_1740_1770	0.214	0.333	29	0.857	0.259
F_20220220_1_1860_1890	0.446	0.471	40	0.919	0.491
F_20220220_1_1800_1830	0.637	0.599	72	0.926	0.697
F_20220220_1_1710_1740	0.221	0.351	22	0.853	0.268

Table 1). The pipeline produced only 396 identity switches across all test videos. This was the lowest among all methods, demonstrating the effectiveness of the deep appearance embeddings for maintaining consistent player identities through occlusions and temporary disappearances. Performance varied significantly across sequences depending on player density and camera distance. As seen in Table 4, videos with clear player separation (F...1890_1920 and F...1830_1860) achieved MOTA scores above 0.74, while crowded sequences with distant players (F...1950_1980) dropped to 0.09 MOTA. The pipeline achieved the highest average precision (0.888) and recall (0.475), indicating superior detection coverage compared to both classical and BoT-SORT approaches. As expected, the increased accuracy comes at a clear computational cost. Processing required a total time of 4,281.86 seconds (over an hour). The appearance embedding extraction for each of the detections is the primary bottleneck, requiring a forward pass through the MobileNetV2 [10] encoder for every detected bounding box.

5 Discussion

5.0.1 Method Comparison. Across methods, there are clear trade-offs between speed and accuracy. DeepSORT provides a 44% relative improvement in MOTA compared to the classical baseline (0.426 versus 0.296) and a 72% relative improvement compared to BoT-SORT (0.426 versus 0.248). The identity preservation advantage is even more pronounced: DeepSORT achieves 51% fewer ID switches than BoT-SORT and 31% fewer than classical ones. However, the computational runtime increases are significant. The classical pipeline runs 12.7x faster than DeepSORT (33.79s vs. 428.19s per sequence) while producing similar precision. For applications prioritizing speed over identity consistency, it remains competitive. All methods faced similar challenges: congested players, small players far from the camera, and rapid occlusion or re-appearance patterns during fast breaks.

5.0.2 Classical Method Insights. The competitive accuracy of the classical pipeline (0.875) tells us that traditional computer vision methods remain useful for foreground-background separation. The MOG2 model was able to learn the field texture and illumination properly and provided good foreground segmentation for most cases. The major drawback of this approach is that it fails systematically for non-moving players. Football has many interrupt cases (e.g., waiting for a throw-in) where players blend into the background model. The Kalman filter and Hungarian algorithm offer a light approach to tracking, close to real-time speeds. We imagine that for use cases where immediate result delivery is the priority, advantages of classical computer vision could overcome losses in performance.

5.0.3 BoT-SORT Limitations. The surprisingly low performance of BoT-SORT (MOTA=0.248, lowest among all algorithms) calls for investigation. The camera motion compensation component in our tracker, intended for handheld or moving cameras, likely adds noise when applied to our static camera videos. The homography may disrupt the Kalman filter state instead of smoothing predictions. Further, the appearance model in BoT-SORT (a light feature extractor) might not be discriminative enough for soccer player detection, resulting in flickering and additional switches. Future attempts could attempt more parameter turning the specific video format or reconsider the usage of BoT-SORT altogether.

5.0.4 DeepSORT Trade-offs. The YOLOv11 + DeepSORT pipeline emphasizes the importance of appearance-based re-identification for tracking. The reduction in ID switches can be explained by richer appearance features through deep embeddings and the cascaded matching strategy. This approach is particularly effective when players temporarily leave the frame or become occluded behind teammates, as the appearance gallery allows re-identification even after extended absences. However, a significant gap between best (MOTA=0.747) and worst (MOTA=0.089) sequences reveals the pipeline's sensitivity to player density and camera distance. Sequences where players occupy more pixels benefit substantially from the appearance features, while distant, small players in crowded scenes produce low-quality embeddings that degrade matching performance. An upsampled inference resolution (*imgsz* = 1920) helps mitigate this for mid-field players but cannot fully compensate for players near the far sideline given the ultra-wide format. Additionally, the computational overhead of DeepSORT defines a key trade-off for MOT: embeddings bring discriminative appearance features, but the per-detection CNN inference creates a bottleneck scaling with detection count. For the 89,875 total detections across our test set, this amounts to substantial GPU time. Future work could investigate lightweight embedding networks, caching strategies, or batch processing to cut down on this overhead while maintaining improvements to tracking accuracy.

5.0.5 Data-Specific Challenges. All three methods particularly struggled given characteristics specific to the wide-angle soccer footage: ultra-wide aspect ratio (6500×1000 pixels) causes some players to become extremely small and difficult to detect. The frame-rate of 25 FPS could produce significant motion between frames when players sprint. These challenges suggest that fine tuning parameters for all methods could yield further improvements. Given the variety of performance across individual test sequences, optimal parameters likely exist for each individual video.

6 Conclusion

We compared classic computer vision techniques with deep learning for multi-object tracking in soccer footage. Although the best deep learning method boasted higher precision and recall, it came at an expected cost of speed. Across ten 30-second test sequences, YOLOv11 + DeepSORT performed the best on-average in all quantitative metrics, confirming that linking appearances via deep features helps track players. However, this improvement demands heavy computation, running over 12x slower than the classic method. Our classical method reached a modest MOTA of 0.296 with high speed, merely struggling with low recall values. Unexpectedly, YOLOv11 + BoT-SORT did worse than both other pipelines, indicating advanced trackers being heavily context-dependent. In this case, we suspect BoT-SORT's camera motion correction component added noise. Our study confirms trade-offs between precision in tracking performance and speed of computation for MOT. If speed isn't key, deep learning trackers using visual traits perform best. Otherwise, traditional approaches are practical, especially if minor mismatches in ID or misses can be tolerated.

7 Statements of Individual Contribution

Rishit Chatterjee: Developed the YOLOv11 + BoT-SORT deep learning pipeline. Wrote Methods and Results sections for BoT-SORT and contributed to the evaluation framework.
Adit Agarwal: Developed the YOLOv11 + DeepSORT tracking pipeline. Wrote Methods and Results sections for DeepSORT and contributed to the discussion trade-offs.
Evan Chen: Developed the classical pipeline and evaluation framework for all pipelines. Wrote Introduction, Methods and Results sections for the classical pipeline and evaluation.

All authors contributed equally to the ideation, data preparation, experimental design, and qualitative analysis.

References

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

[2] Keni Bernardin and Rainer Stiefelhagen. 2008. Evaluating Multiple Object Tracking Performance: The CLEAR MOT Metrics. *EURASIP Journal on Image and Video Processing* 2008 (2008), 1–10.

[3] Alex Bewley, ZongYuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. 2016. Simple Online and Realtime Tracking. In *2016 IEEE International Conference on Image Processing (ICIP)*. IEEE, 3464–3468.

[4] Christoph Heindl. 2017. py-motmetrics: Metrics for multiple object tracker (MOT) benchmarking. https://github.com/cheind/py-motmetrics.

[5] Rudolf E Kalman. 1960. A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering* 82, 1 (1960), 35–45.

[6] Harold W Kuhn. 1955. The Hungarian method for the assignment problem. *Naval Research Logistics* 2, 1-2 (1955), 83–97.

[7] L. Leal-Taixé, A. Milan, I. Reid, S. Roth, and K. Schindler. 2015. MOTChallenge 2015: Towards a Benchmark for Multi-Target Tracking. *arXiv:1504.01942 [cs]* (April 2015). http://arxiv.org/abs/1504.01942 arXiv: 1504.01942.

[8] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. You Only Look Once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*. 779–788.

[9] Ergys Ristani, Francesco Solera, Roger Zou, Rita Cucchiara, and Carlo Tomasi. 2016. Performance Measures and a Data Set for Multi-Target, Multi-Camera Tracking. In *Computer Vision–ECCV 2016 Workshops: Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part II 14*. Springer, 17–35.

[10] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. 2018. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4510–4520.

[11] Atom Scott, Ikuma Uchida, Ning Ding, Rikuhei Umemoto, Rory Bunker, Ren Kobayashi, Takeshi Koyama, Masaki Onishi, Yoshinari Kameda, and Keisuke Fujii. 2024. TeamTrack: A Dataset for Multi-Sport Multi-Object Tracking in Full-pitch Videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. 2971–2980.

[12] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. 2017. Simple Online and Realtime Tracking with a Deep Association Metric. In *2017 IEEE International Conference on Image Processing (ICIP)*. IEEE, 3645–3649.

[13] Zoran Zivkovic. 2004. Improved Adaptive Gaussian Mixture Model for Background Subtraction. In *Proceedings of the 17th International Conference on Pattern Recognition (ICPR)*, Vol. 2. IEEE, 28–31.

A Appendix

A.1 Code

All data and code used for this project is publicly available at https://github.com/tihsir/sports-data-tracker. Details on the specific files and folders containing our data, executed code, and saved results are included in the `README.md` file.

A.2 Result Video

As mentioned in our Results section, a YouTube video showcasing an example of our results (drawn detections for all three methods on a single test sequence) can be found at https://www.youtube.com/watch?v=mITuZZlb2yQ.