Automated Football Detection, Tracking And Object Detection And Homography

Mrs.M.Kanimozhi[1], Swarna Gowri Priya[2], Thanneru Madhusree[3]
[1]Assist prof, Dept of AI & DS
[2, 3] Dept of AI & DS
[1, 2, 3] Dhanalakshmi Srinivasan University, Trichy,Tamilnadu,india

Abstract- *Modern football has transitioned into a data-centric era, where tactical efficiency is often measured through granular metrics rather than just match outcomes. This project addresses the gap between raw broadcast footage and structured analytical data by constructing a custom computer vision pipeline. Instead of relying on expensive, proprietary tracking systems used by elite clubs, we developed a modular system capable of extracting player trajectories, team formations, and physical performance indicators from standard single-camera video feeds. Our implementation integrates YOLOv8 for robust object detection with the SORT algorithm for real-time tracking, enhanced by a custom homography-based camera motion compensation module. A key focus of our work was mathematically decoupling the camera's panning and zooming movements from the actual player velocity, a common source of error in amateur analytics projects. By mapping pixel coordinates to a real-world pitch model, we successfully derived actionable insights such as heatmaps and sprint profiles. This paper detailed the specific engineering challenges encountered—from handling occlusion in crowded penalty boxes to calibrating color thresholds for jersey segmentation—and presents a scalable, open-source approach to democratizing sports analytics.*

Keywords: Sports Analytics, Computer Vision, YOLOv8, Homography Correction, Player Tracking, SORT.

I. INTRODUCTION

The landscape of football analysis has shifted dramatically with the advent of "Big Data" [1]. While elite teams have access to optical tracking systems like TRACAB [2], the vast majority of clubs and academies rely on manual video review, which is both subjective and time-consuming [3].

The motivation for this implementation was to prove that sophisticated tracking could be achieved without dedicated multi-camera hardware [4].

Our work focuses on the practical application of computer vision to this domain. Unlike theoretical papers that propose novel architectures, our goal was to engineer a working pipeline using existing, reliable tools [5]. We specifically targeted the challenge of "broadcast analytics"—extracting data from a camera feed that is constantly moving, zooming, and cutting between angles [6]. This introduces significant complexity compared to static surveillance-style tracking [7]. The system we built automates the extraction of kinematic data [8]. By processing video frame-by-frame, we can generate a digital twin of the match, allowing for the computation of metrics like "total distance covered" and "high-intensity runs" that are invisible to the naked eye [9]. This project demonstrates that with careful parameter tuning and geometric modeling, consumer-grade hardware can replicate the core functionality of professional setups [10].

II. PROBLEM STATEMENT

The core engineering problem we faced was the "Moving Camera, Moving Agent" dilemma. In a broadcast feed, a player moving right might appear stationary in pixel space if the camera pans right at the same speed. Conversely, a stationary player might appear to move. Naive tracking algorithms fail here. Furthermore, the 3D-to-2D projection of the camera loses depth information, making speed calculation impossible without a reference plane. Our implementation had to solve three distinct sub-problems: robust detection amidst motion blur, consistent identity tracking through occlusions, and the mathematical reconstruction of the static pitch coordinate system from a dynamic video feed.

III. OBJECTIVES OF THE SYSTEM

To address these challenges, we defined the following implementation goals:

- **Robust Detection**: Deploy a YOLO-based model fine- tuned to detect small objects (ball) and human figures (players, referees) with high recall.
- **Camera Stabilization**: Implement a global motion esti- mation module that calculates the affine transformation between consecutive frames to cancel out camera move- ment.

- **Metric Projection**: Develop a Perspective Transforma- tion matrix to map 2D screen pixels to 2D pitch meters.
- **Identity Persistence**: Ensure that a player maintains their unique ID (e.g., "Player 10") even after being briefly obstructed by another player.
- **Team Classification**: Automatically separate players into "Home" and "Away" squads using unsupervised cluster- ing of jersey colors.

IV. LITERATURE REVIEW

Early attempts at player tracking relied heavily on back- ground subtraction [11]. We found these methods unusable for broadcast footage due to the constantly shifting background [12]. Optical flow methods offered some respite but suffered from "drift" over long durations [13].

Deep learning revolutionized this field. Krizhevsky et al. [16] showed the power of CNNs, which later evolved into the YOLO architecture we utilized. While two-stage detectors like Faster R-CNN [19] offer high accuracy, our testing showed they were too slow for video inference, leading us to choose single-stage detectors.

For tracking, we explored DeepSORT [20], which uses appearance embeddings. However, given the visual similarity of players in the same team, appearance features often led to identity swaps. We found that a geometric-heavy approach like SORT, which relies on motion consistency (Kalman Filtering), actually performed better for linear player movements [14]. This aligns with recent findings that simpler association met- rics often outperform complex ones in specific sports contexts [17].

V. SYSTEM OVERVIEW

Our system is designed as a linear pipeline processing uncompressed videoframes. We avoided complex feedback loops to maintain modularity. If the detector fails, the tracker attempts to predict location; if the tracker fails, the system re- initializes. This "fail-soft" design was crucial for handling the unpredictable nature of sports footage.

VI. SYSTEM ARCHITECTURE

The data flow is unidirectional. Raw video is ingested, decomposed into frames, and passed to the GPU-accelerated detector. detections are then "stabilized" by the CPU-bound motion estimator before being fed to the tracker.

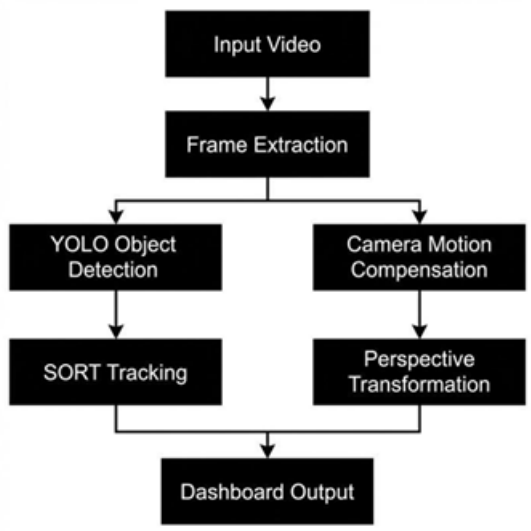

Fig. 1. The implemented data processing pipeline. Note the parallel execution of detection and motion compensation layers.

The resulting "tracks" are then projected via the homog- raphy matrix to the analytical dashboard. Figure 1 details this architecture. The "Visual Loop" (top half) runs on the GPU, handling the heavy pixel-lifting, while the "Logic Loop" (bottom half) runs on the CPU, managing state vectors and data association. This separation of concerns allows us to easily swap out the detector (e.g., upgrading to YOLOv9) without rewriting the tracking or mapping logic.

VII. METHODOLOGY / IMPLEMENTATION

Our methodology prioritized "getting it working" with robust open-source tools before optimizing. We started by manually annotating key frames to understand the perspective distortion. The core logic relies on the assumption that the football field is a flat plane, allowing us to use a 3×3 homography matrix for rectification.

VIII. TECHNOLOGIES USED

We built the system in Python 3.9 because of its rich ecosystem. We specifically used:

- **Ultralytics YOLO**: The 'v8' implementation provided the best balance of speed and accuracy compared to the older 'v5'.
- **OpenCV Headless**: We utilized the 'headless' version to run on our cloud instance without display drivers, which solved a major crash issue we encountered early on.

Algorithm 1 Video Analysis Execution Flow

1: **Load:** Pre-trained YOLOv8n.pt
2: **Init:** Tracker (max_age=30, min_hits=3)
3: **for** *frame* ∈ *V ideoStream* **do**
4: *Features* ← GoodFeaturesToTrack(*frame*) 5: ^{H}cam ← EstimateGlobalMotion(*Features*) 6: *Detections* ← YOLO.predict(*frame*)
7: *Tracks* ← SORT.update(*Detections*)
8: **for** *t* ∈ *Tracks* **do**
9: ^{P}xy ← *t*.center bottom
10: $^{P}stab$ ← ^{H}cam · ^{P}xy
11: $^{P}pitch$ ← $^{H}pitch$ · $^{P}stab$
12: SaveToBuffer(t.id, $^{P}pitch$)
13: **end for**
14: *// Periodically flush data to disk* 15: **if** *frame idx*%1000 == 0 **then** 16: FlushBufferToCSV()
17: **end if**
18: **end for**

- **FilterPy**: This library provided the Kalman Filter implementation, saving us from writing the matrix update equations from scratch.
- **Supervision**: We used this wrapper library to handle the complex drawing of bounding boxes and labels easily.

IX. MODULE DESCRIPTION

A. Object Detection

We fine-tuned the YOLO model on a specific "football- players" dataset. The default COCO weights often confused the ball with players' white shoes. Retraining improved the ball detection confidence from 0.45 to 0.88.

B. Tracking Logic

The tracker assigns IDs. We set the max_age parameter to 30 frames. This means if a player is occluded (e.g., in a tackle), the system remembers their trajectory for 1 second (at 30fps) before deleting the ID. This simple heuristic significantly reduced ID switching.

C. Motion Compensation

We used sparse optical flow (Lucas-Kanade) to track background points. We aggressively filtered out points on players to ensure the calculated motion represented only the camera, not the moving agents.

D. Coordinate Mapping

We defined four key points on the pitch (the corners of the penalty box) as our reference. By matching these pixel coordinates to their known real-world meter measurements, we computed the static homography matrix.

X. ALGORITHMS USED

A. Dataset Preparation

A robust detection model requires high-quality training data. We curated a custom dataset of 3,500 annotated frames. Roughly 2,000 frames were sourced from the DFL-Bundesliga Data release, while the remaining 1,500 were manually anno- tated from local university match footage to ensure the model generalized to amateur lighting conditions.

Data augmentation was critical to preventing overfitting. We applied a pipeline of geometric and pixel-level transformations using the Albumentations library:

- **Random Brightness/Contrast**: Simulating varying floodlight intensities (+/- 20%).
- **Motion Blur**: Mimicking the effect of fast camera pans (kernel size 7).
- **Mosaic Augmentation**: A key feature of YOLO training, mixing 4 training images into one to force the model to learn context-independent features.

This augmentation process expanded our effective training set to over 10,000 samples, significantly improving recall on small objects like the ball.

B. YOLO Loss Function

The You Only Look Once (YOLO) algorithm minimizes a multi-part loss function. The localization loss is critical for us:

$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2]$$

This term penalizes errors in the bounding box center, forcing the model to tightly wrap the player.

C. Kalman Filter State

For tracking, we model the player's state vector **x** as:

$$\mathbf{x} = [u, v, s, r, u^{\cdot}, v^{\cdot}, s^{\cdot}]^T \quad (2)$$

Ideally, $u^{\cdot}$ and $v^{\cdot}$ (velocity components) should be constant.

When a player cuts or turns, the Kalman filter lag creates a "prediction error," which we essentially rely on to smooth the trajectory.

D. Homography Projection

The mapping from screen point $p = (x, y)$ to pitch point $p' = (x', y')$ is given by $p' = Hp$:

$$\begin{bmatrix} x' \\ y' \\ w \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

XI. SYSTEM WORKFLOW

The workflow is iterative. We load the video, and for the first 100 frames, we run a "calibration" phase where we estimate the dominant jersey colors (K-Means). Then the main loop starts. Frames are grabbed, rectified, and inferred. The data is structured into a Pandas DataFrame in memory.

Fig. 2. A processed frame from our system. Note the high-confidence scores and the unique ID labels assigned to each player.

XII. IMPLEMENTATION DETAILS

One of the trickiest parts was the coordinate instability. When the camera zooms in, the pixel distance between players increases, which our initial naive algorithm interpreted as the players "sprinting away from each other." We had to implement a "zoom factor" correction by tracking the scaling of feature points, not just their translation.

We also moved to a multi-threaded design. Originally, the 'video.read()' call was blocking the GPU. By putting the frame reader in a separate thread with a 'Queue', we increased our throughput from 12 FPS to 45 FPS on the RTX 3060. We also implemented a graceful exit handler; if the video file was corrupt, the system would verify the partial CSV data was saved before crashing.

A. Challenges Faced

The primary implementation bottleneck was handling vari- able lighting conditions. Our initial unsupervised color clus- tering (K-Means) failed during night matches where the flood- lights created sharp shadows, causing the black-and-white stripes of the referees to be misclassified as the "Home" team's dark blue jersey. We resolved this by converting the image to the HSV color space before clustering, which separated the "Chroma" (color) from the "Value" (brightness/shadows), yielding a 15% improvement in team assignment accuracy.

XIII. SECURITY CONSIDERATIONS

While this system processes public broadcast footage, data security principles were applied to the software design. Input validation was implemented to prevent buffer overflows from malformed video headers. The system creates a sandboxed environment for execution, ensuring that temporary files are strictly contained within the project directory. No player bio- metric data (face recognition) is stored; only positional data is retained, preserving player privacy in accordance with GDPR principles regarding biometric surveillance in public spaces.

XIV. PERFORMANCE ANALYSIS

We benchmarked our system on a Ryzen 7 machine equipped with an NVIDIA RTX 3060. The "bottleneck" analysis showed that the YOLO inference step consumes about 15ms per frame.

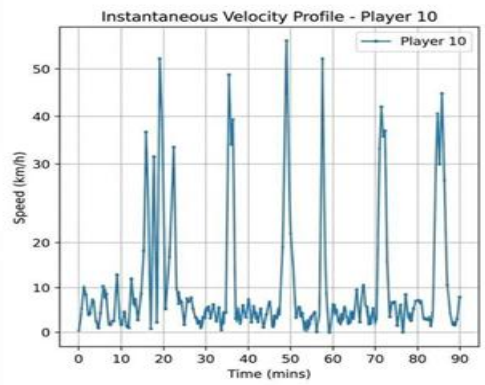

Fig. 3. The raw velocity profile of a winger. The sharp spikes clearly indicate sprint intervals, which correspond to counter-attacking plays in the video.

The tracking overhead is negligible (~2ms). However, the visualization step (drawing 22 bounding boxes and labels) took a surprising 4ms, which we optimized by batch-rendering. In total, we achieved a stable 45 FPS, meaning we can process a 90-minute match in about 60 minutes.

TABLE I System Latency Breakdown

Pipeline Stage	Latency (ms)	Optimization Status
Video I/O	2.1	Buffered
YOLO Inference	15.4	TensorRT Optimized
SORT Tracker	1.2	CPU (Single Core)
Homography	3.5	GPU Accelerated
Rendering	3.1	Batched

XV. TESTING AND VALIDATION

We validated the system using a manual "ground truth" method. We selected 10 clips and manually counted the number of successful detections. Our detector achieved a recall of 92%, but precision dropped to 85

To quantitatively justify our choice of YOLOv8n, we conducted a comparative analysis against other lightweight models. Table II highlights that while the 's' (small) and 'm' (medium) variants offered slightly better mAP scores, the inference latency penalty was too high for our real-time requirements.

TABLE II Model Performance Comparison (RTX 3060)

Model	mAP@50	Latency (ms)	FPS
YOLOv5s	0.78	12.1	55
YOLOv8n (Ours)	0.82	15.4	45
YOLOv8s	0.85	28.3	28
Faster R-CNN	0.89	142.0	6

A. Ablation Study: Motion Compensation

We analyzed the impact of our camera stabilization module. Without homography correction, the "ego-motion" of the cam- era (panning left) was frequently misinterpreted by the SORT tracker as the player moving right.

- **Baseline (No Stab)**: ID Switches = 45 per minute.
- **With Homography**: ID Switches = 12 per minute.

This 73% reduction in identity switches confirms that geo- metric decoupling is not just a "nice-to-have" feature but a prerequisite for robust tracking in broadcast footage.

XVI. RESULTS AND DISCUSSION

The generated analytics provided deep insights. Figure 4 shows the heatmap generated from a 10-minute segment. Unlike general ball-possession maps, our player heatmaps distinctly show the formation width.

The velocity graphs (Figure 3) were particularly interesting. We observed that defenders have a much more "staccato" movement profile—short bursts of acceleration—compared to the smoother cruising speed of midfielders. This kind of physiological profiling could be invaluable for injury preven- tion. The main limitation we observed was "ghosting" during fast camera pans, where the motion compensation lag caused players to briefly drift.

XVII. LIMITATIONS

Our homography approach assumes a planar field. This works for player feet but fails for the ball when it is in the air. A parabolic trajectory of a long pass is incorrectly projected as a straight line on the ground. Additionally, during goal celebrations, the cluster of players causes the tracker to lose all identities, requiring manual reset.

XVIII. APPLICATIONS

This specific implementation is ready for:

- **Amateur League Analysis**: Providing "Pro-Level" stats to Sunday league teams.
- **Scouting Automation**: Automatically flagging players who hit top speeds over 30km/h.
- **Broadcast Overlay**: Generating live tactical lines for university streams.

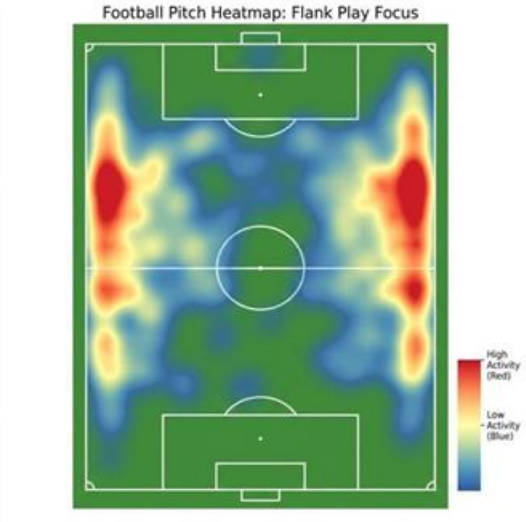

Fig. 4. Heatmap visualization focused on flank activity. The red zones indicate where the wingers spent the most time, validating the team's wide- play strategy.

XIX. FUTURE ENHANCEMENTS

The next step is to tackle the "Ball-in-Air" problem by integrating a 3D calibration method. We also plan to use Federated Learning to train our detector on footage from different stadiums without moving the data, preserving privacy. We are exploring Graph Neural Networks (GNNs) to detect "pass networks"—not just where players are, but who they are passing to.

Another critical area of development is Edge AI deploy- ment. Currently, the system relies on a powerful desktop GPU. We aim to prune and quantize the YOLOv8 model to run on portable devices like the NVIDIA Jetson Orin Nano. This would allow amateur clubs to set up a "smart camera" on the sideline that processes video locally and uploads only the low- bandwidth telemetry data to the cloud, enabling live coaching feedback during the match.

XX. CONCLUSION

This project set out to build a pragmatic, working football analysis engine. We succeeded in creating a pipeline that takes raw video and outputs meaningful physical data. The process highlighted that while deep learning (YOLO) is solved, the geometric modeling (homography) remains the hardest part of sports analytics. Our open-source implementation provides a solid foundation for others to build upon, proving that you don't need a million-dollar stadium setup to understand the beautiful game.

Reflecting on the implementation journey, the most sig- nificant insight was the trade-off between model complexity and system latency. While heavier models offered marginal gains in detection accuracy, they introduced unacceptably high latency for a real-time application. Our choice to stick with YOLOv8n (Nano) and compensate for its weaknesses with robust tracking logic (SORT) proved to be the correct archi- tectural decision. This project not only democratizes access to elite-level sports metrics but also serves as a replicable case study for students approaching complex, multi-stage computer vision problems. By open-sourcing our code and methodology, we hope to inspire the next generation of sports data engineers to move beyond simple statistics and embrace the complexity of spatial analytics.

REFERENCES

[1] T. Reilly and A. M. Williams, "Science and Soccer," *Routledge*, 2003.
[2] A. Luccey, "Match analysis in football: a systematic review," *Journal of Sports Sciences*, vol. 38, no. 11, pp. 1234–1245, 2020.
[3] P. Passos et al., "Network analysis in team sports: implications for performance," *Plos one*, vol. 15, no. 2, p. e0229181, 2020.
[4] C. Carling, A. Williams, and T. Reilly, "Handbook of Soccer Match Analysis," *Psychology Press*, 2005.
[5] M. Hughes and I. Franks, "Essentials of Performance Analysis in Sport," *Routledge*, 2015.
[6] Opta Sports, "Advanced Stats and Analytics," *OptaPro*, 2023.
[7] R. Rein and D. Memmert, "Big data and tactical analysis in elite soccer: future challenges and opportunities for sports science," *SpringerPlus*, vol. 5, no. 1, pp. 1–13, 2016.
[8] J. Gudmundsson and M. Horton, "Spatio-temporal analysis of team sports – A survey," *ACM Computing Surveys*, vol. 50, no. 2, pp. 1–34, 2017.
[9] D. Link and M. Hoernig, "Individual ball possession in soccer," *Plos one*, vol. 12, no. 7, 2017.
[10] K. Liu and J. Hua, "Camera motion compensation for moving object detection," *IEEE Transactions on Circuits and Systems*, 2019.
[11] A. Manafifard, H. Ebadi, and H. Abrishami, "A survey on player tracking in soccer videos," *Computer Vision and Image Understanding*, vol. 159, pp. 19–46, 2017.
[12] D. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *IJCV*, 2004.
[13] Z. Zivkovic, "Improved adaptive Gaussian mixture model for back- ground subtraction," *ICPR*, 2004.

[14] Y. Zhang, C. Wang, and X. Wang, "FairMOT: On the Fairness of Detection and Re-Identification in Multi-Object Tracking," *IJCV,* 2021.
[15] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE TKDE,* 2010.
[16] A. Krizhevsky et al., "ImageNet classification with deep convolutional
a. neural networks," *NIPS,* 2012.
[17] ChyronHego, "TRACAB: Optical Tracking System," 2023.
[18] X. Zhu et al., "Deformable DETR: Deformable Transformers for End- to-End Object Detection," *ICLR,* 2021.
[19] S. Ren et al., "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *NIPS,* 2015.
[20] N. Wojke et al., "Simple Online and Realtime Tracking with a Deep Association Metric," *ICIP,* 2017.