

Automated Football Video Analysis System

Nehal Deepak Nemade

Abstract

This paper presents an automated football video analysis system that detects and tracks the players, referees and the ball across video frames, and at the same time goes a step further by classifying the teams and flagging kick events on its own. The pipeline combines a YOLO-based object detector with ByteTrack multi-object tracking, a Kalman filter for smoothing the ball trajectory, KMeans color clustering for team classification and a rule-based kick event detector, whereas each module handles one specific part of the problem. There are two main contributions to discuss. The first is a speed-aware ball possession assignment approach, which reduces the false positives on high-velocity passes and kicks, and the second is an on-screen annotation pipeline that overlays the output video with player IDs, team colors, ball speed and kick events, so that a coach can review a complete match without re-watching it frame by frame.

1. Introduction

Automated analysis of football (soccer) footage has substantial applications in sports analytics, coaching and broadcast production. On the other hand, manual annotation of player positions, possession events and kick actions is time-consuming and prone to inconsistency, since two people marking up the same clip rarely agree on every touch. Computer vision methods provide a scalable solution to this problem, analyzing whole match recordings frame by frame without the fatigue or subjectivity that comes with doing it by hand.

This system tackles three major challenges at once: (1) robust multi-object tracking of players and the ball under occlusion and fast motion, (2) accurate team classification using only the visual jersey color, and (3) reliable kick event detection that avoids the false positives caused by ball flyovers and crowd scenes near the sideline. In this paper, I will first explain the system architecture, then describe each method in detail, and finally discuss the current limitations and the planned future work.

2. System Architecture

The pipeline consists of five primary modules, each implemented as a Python class, and the table below lists the file and the responsibility of each one.

Table 1: Pipeline modules, their files and responsibilities

Module	File	Responsibility
Tracker	tracker.py	YOLO detection, ByteTrack tracking, speed estimation, annotation
TeamAssigner	team_assigner.py	KMeans jersey-color

Module	File	Responsibility
		clustering, player-team mapping
PlayerBallAssigner	player_ball_assigner.py	Frame-level ball possession assignment with speed gating
Kick Detector	kick_detector.py	Possession-transition kick event detection with cooldown
BallFilter	ball_filter.py	Kalman filter smoothing of ball trajectory

The main.py orchestration script reads the input video, runs each module in sequence, and writes the annotated output. Adding to this, because every module lives in its own file, any one of them can be replaced without touching the rest of the pipeline.

3. Methods

3.1 Object Detection and Tracking

For the object detection, a fine-tuned YOLOv8 model (best.pt) is used to detect the players, goalkeepers, referees and the ball. The goalkeepers are then re-labelled as players, so that they are tracked with a consistent ID assignment, and the detection is performed in batches of 20 frames for efficiency.

The detected objects are then passed to the ByteTrack algorithm (via the supervision library), which maintains consistent track IDs over the frames with the help of appearance and motion cues. The ball detections, however, are treated separately outside of the tracker, where only the highest confidence detection per frame is retained and stored under the fixed key 1, which simplifies the downstream access.

3.2 Ball Trajectory Smoothing

Motion blur and detection uncertainty create a frame-to-frame jitter in the raw ball detections. To handle this, the ball centre coordinate sequence is filtered with a 4-state Kalman filter (position and velocity in x and y), which is initialized at the first valid detection and generates smoothed predictions for the following frames. These smoothed predictions are used only for the speed estimation.

The missing ball positions, meaning the frames where the ball is not detected, are filled by pandas linear interpolation followed by a backward-fill, in order to provide a continuous trajectory for the downstream processing.

3.3 Ball Speed Estimation

The ball speed is estimated per frame using the following formula:

$$\text{speed (km/h)} = \text{pixel_distance} \times \text{meters_per_pixel} \times \text{fps} \times 3.6$$

Here, meters_per_pixel is derived from the known pitch width (68 m) and the frame width in pixels. The Kalman-smoothed ball centre is used for the pixel displacement, since this reduces the noise in the measurement.

3.4 Team Classification

The team assignment is performed using a two-stage KMeans clustering approach. In the first stage, the

top-half bounding box crop of each player is clustered into two color groups using KMeans ($k=2$), and the cluster corresponding to the background (identified by the majority of the corner pixels) is discarded, which leaves the dominant jersey color of that player.

In the second stage, all the player jersey colors from the first video frame are jointly clustered ($k=2$) to produce two global team centroids. All the subsequent player crops are then classified by predicting their cluster membership against these centroids, and the team assignments are cached per player ID to ensure temporal consistency.

3.5 Ball Possession Assignment

Possession is granted to the player whose feet are closest to the ball centre, after the assignment passes through three filters:

- **Speed gate:** if the ball speed exceeds 35 km/h, no possession is assigned (flyover filter).
- **Airborne gate:** if the ball bounding-box height is less than 10 pixels, the ball is considered airborne and no assignment is made.
- **Waist gate:** if the ball centre is above the player's waist (y-coordinate), then that player is excluded.

Adding to this, when the ball is small (height less than 20 px), the effective distance threshold is scaled down proportionally, so that the ball is not assigned to distant players when it is partially out of the frame.

3.6 Kick Event Detection

A kick is detected when all three of the following conditions are true at the same time:

- Prev_has_ball=True (the player had the ball in the previous frame).
- The player has lost possession of the ball in this frame.
- The current ball speed is greater than 10 km/h (ruling out the intentional stops).

A 10-frame cool-down period removes the duplicate detections in crowded scenes, whereas the possession state is still updated normally within the cooldown window, in order to avoid state corruption. The output video is then annotated with the kick event for two seconds after each detected kick.

4. Implementation Details

The table below summarizes the libraries and versions used for each component of the implementation.

Table 2: Libraries and versions used for each component

Component	Library / Version	Notes
Object detection	Ultralytics YOLOv8	Custom fine-tuned weights
Multi-object tracking	supervision (ByteTrack)	Consistent IDs across frames
Kalman filtering	OpenCV KalmanFilter	4-state: x, y, vx, vy
Team clustering	scikit-learn KMeans	$k=2$, k-means++ init
Video I/O	OpenCV (cv2)	AVI output, per-frame annotation

Component	Library / Version	Notes
Data handling	pandas / NumPy	Interpolation and array ops

The tracking stubs (serialised pickle files) can optionally be loaded to skip the re-detection, which enables rapid iteration during development. The annotation pipeline draws ellipses under the players (color-coded by team), downward triangles above the ball, ball speed labels, and kick event banners with a drop-shadow for readability.

5. Limitations and Future Work

The current implementation has several known limitations, which I will list first, followed by the planned improvements.

- Team classification is based on two teams with visually distinct jersey colors, therefore mislabelling may occur with similar colors or third kits.
- The `meters_per_pixel` calibration is valid only for a fixed camera angle and a standard pitch width, whereas pan/tilt/zoom cameras require a dynamic recalibration.
- Kick detection is heuristic, so it can miss the low power passes (below the 10 km/h threshold) and can also generate false positives during sliding tackles.
- The KMeans team assigner is initialised on the first frame only, and for this reason it is sensitive to crowd noise or occlusion at the match start.

The planned improvements are the following:

- Pitch mapping with homography for precise metric distances.
- Appearance embedding (Re-ID) for recovering the lost tracks after occlusion.
- A learned kick classifier to replace the threshold-rule based approach.
- Rolling window dynamic team re-clustering to handle the kit changes.

6. Conclusion

This work demonstrates a modular, end-to-end football video analysis pipeline that combines object detection, multi-object tracking, Kalman filtering, color-based team classification and rule-based event detection, with each stage kept independent enough to be swapped out on its own. It can be concluded that the speed-aware possession allocation (with the 35 km/h gate) and the kick cooldown mechanism reduce the false positives significantly compared to the naive proximity-based approaches. In my opinion, I would personally say this is the part of the project I'm most satisfied with.

The system generates an annotated output video that is easy for coaches and analysts to review without touching the underlying code, and its modular design makes it straightforward to extend later with more sophisticated models, whether that is a learned kick classifier or homography-based pitch mapping. At the end, this means a coach can spend the time on real decisions about the team, instead of watching and marking the same footage again and again.