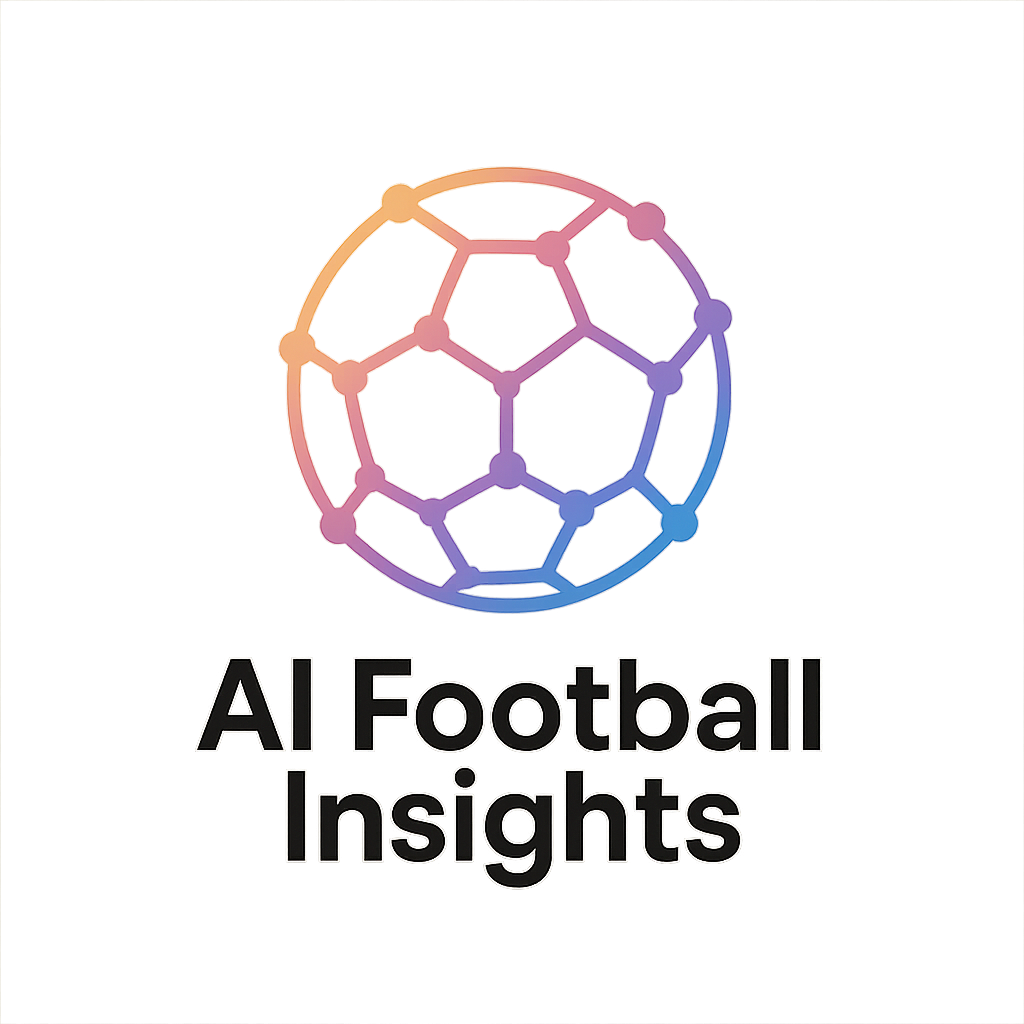


AI FOOTBALL ANALYSIS



**Members:**

Ishma Hafeez – 21k 4688

Bilal Abdul – 21k 4522  
Umer Naeem – 21k – 4927

**Abstract**

Football is more than just a game of goals it’s a complex interplay of player positioning, team tactics, possession dynamics, and split-second decisions. This project takes on the challenge of making sense of football match footage using Artificial Intelligence. The methodology progresses from basic object detection using pre trained YOLO models to sophisticated analysis involving fine tuning models with custom datasets, implementing robust object tracking, assigning players to teams based on jersey colors, determining ball possession, calculating team ball control statistics, and estimating camera movement. The overarching aim is to extract meaningful insights from video footage, such as player movements, team formations, and possession dynamics, presented through custom and clear visualizations.

**Problem Statement**

Using off the shelf AI models for match analysis might seem like an easy win—but it's not. When we tried applying pre trained object detection models directly on football videos, a few problems became immediately clear:

* The ball, being small and fast, often went undetected.
* Players and referees weren’t always differentiated, leading to messy analysis.
* The models couldn’t maintain consistent tracking across frames, which is critical when you want to know who’s where.
* Without special logic, we couldn’t tell who had the ball or even which team a player belonged to.
* Camera movements—like pans and zooms—threw off our understanding of player positions and speed.
* Default visualizations from libraries cluttered the footage, making it hard to actually watch the game.

This project aims to solve those problems by building an AI system that’s tailor made for analyzing football match videos.

**Objectives**

Our focus was to:

* Accurately detect key entities like players, referees, goalkeepers, and the ball.
* Track each of these entities consistently across the video.
* Identify team membership using jersey colors.
* Keep tabs on the ball’s position, even when it's briefly missed.
* Automatically figure out who’s in possession of the ball at any moment.
* Calculate how much time each team controls the ball.
* Analyze camera movement so it doesn’t skew player motion data.
* Add clean, informative visual overlays to the video (without blocking the action).
* Design the entire system to be modular, so it’s easy to improve and build on.

**Methodology**

The project is developed in phases, with each stage building upon the results of the previous ones to create a comprehensive analysis pipeline.

**Step 1: Getting the Data**

We used a video of a football match with an eagle eye view of the field. For training and fine tuning our models, we used annotated images from Roboflow’s Football Player Detection dataset. Everything was formatted in YOLOv8’s style and organized properly before training.

**Step 2: Building the AI Pipeline**

**Object Detection**  
We started with YOLOv8x for quick experimentation and then fine tuned a YOLO12l model using 612 labeled images. This gave us much better detection of players, referees, and the ball.

**Object Tracking**  
For tracking, we used ByteTrack from the Supervision library. Each object detected was given a unique ID, and we maintained these IDs across frames to track players, referees, and the ball. Goalkeepers were relabeled as “players” to simplify the tracking process.

**Team Assignment**  
Using K Means clustering, we extracted shirt colors and grouped players into teams automatically. We applied clustering both at the individual level (per shirt) and overall (first frame) to identify the two dominant team colors. A fix was also added for when goalkeepers were misassigned.

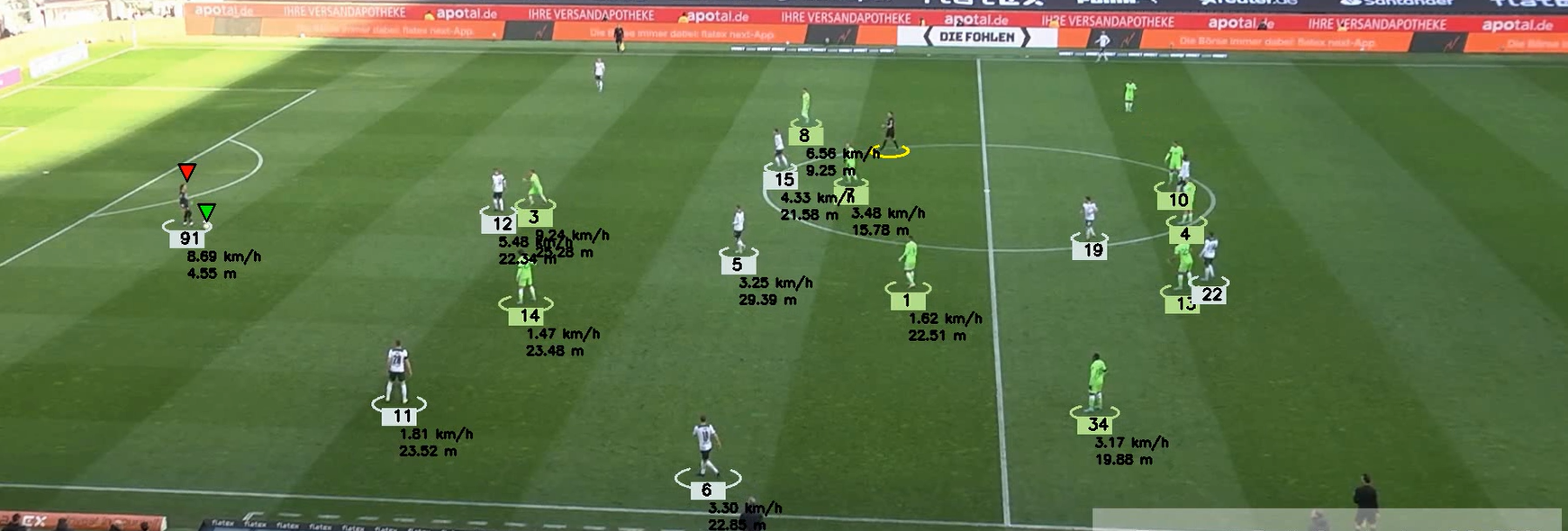
**Ball Interpolation**  
When the ball wasn’t detected in some frames, we filled in the gaps using linear interpolation. This ensured a smooth and continuous ball trajectory throughout the match.

**Possession Detection**  
We determined which player had the ball based on how close the ball was to the player’s inferred foot position. If the ball was within a 70 pixel radius of a player's foot, and no one else was closer, we marked that player as having possession.

**Team Ball Control Calculation**  
As we identified who had the ball frame by frame, we also calculated what percentage of the time each team was in control. If no player had the ball, we carried over the last known possessing team to maintain continuity.

**Camera Movement Estimation**  
To keep track of camera movements like pans and zooms, we used optical flow on static parts of the frame. This helped us separate actual player movement from camera induced shifts.

**4. Results**

****

**Ball Possession Assignment**  
The system reliably assigns ball possession by checking which player’s foot is closest to the ball. A red triangle overlay confirms possession visually and works well in most situations.

**Team Ball Control Stats**  
Ball control percentages are updated in real time and shown on the video. When the ball isn’t clearly possessed, the system assumes the last team still has control, ensuring continuity. Fine tuning team assignments, especially for goalkeepers, improved overall accuracy.

**Camera Movement Estimation**  
Using optical flow, the system estimates camera movement frame by frame by tracking stable features like banners. These estimates are cached and refreshed every frame for consistency, with future visualizations planned to confirm accuracy.

**5. Conclusion**

This project built a robust AI driven system for analyzing football matches—from detecting players and tracking ball possession to estimating camera motion. The output is a more insightful and visually clear match video, laying the groundwork for deeper tactical analysis in the future.

**6. References**

* **Models/Algorithms:** YOLO, ByteTrack, K Means Clustering, Lucas Kanade Optical Flow.
* **Libraries/Frameworks:** Ultralytics, Supervision, OpenCV Python, Scikit learn, Pandas, NumPy, Roboflow SDK.