Football Analysis with YOLO, OpenCV and Python

Hamim Saad Al Raji, Kashshaf Labib, Navid Kamal, Mainul Hasan, Md Mehedi Al Mahmud
BSc in Software Engineering
Department of CSE, Islamic University of Technology
Gazipur, Bangladesh
hamimsaad@iut-dhaka.edu, kashshaflabib@iut-dhaka.edu, navidkamal@iut-dhaka.edu,
mainulhasan@iut-dhaka.edu, mehedialmahmud@iut-dhaka.edu

***Abstract*—In recent years, computer vision has transformed the way sports are analyzed, enabling automated performance tracking and tactical evaluation. This paper presents a comprehensive system for *automated football match analysis* using deep learning-based object detection and tracking. Leveraging the YOLOv5 model [1] for real-time detection of players, the ball, and referees, the system processes match footage to extract key tactical insights. The model was trained on a publicly available dataset [2] comprising labeled frames from nine Bundesliga matches, generated on December 6, 2022, through video segmentation and annotation. For temporal consistency, ByteTrack [3] is used to maintain object identities across frames. To account for camera motion, the system applies Lucas-Kanade-based optical flow estimation [4] for positional correction. Player teams are automatically classified using KMeans clustering based on dominant jersey colors [5], and ball possession is inferred by measuring proximity between the ball and player foot positions. A perspective transformation maps pixel positions to real-world field coordinates, enabling precise speed and distance estimation. This tool aims to assist coaches, analysts, and broadcasters in understanding player dynamics, team formations, and ball control with minimal manual intervention.**

I. Introduction

Football, often referred to as the world's most popular sport, has increasingly embraced technological advancements to enhance game understanding, player performance, and tactical insights. Traditionally, football analysis relied heavily on manual annotation and expert observation, which were not only time-consuming but also prone to subjectivity and inconsistency [6]. Coaches and analysts would watch hours of footage, annotate player movements by hand, and attempt to infer strategies—a process that lacked the precision and scalability required in modern competitive environments.

In contrast, the evolution of computer vision and deep learning has enabled automated, data-driven approaches that can process large volumes of match footage and extract meaningful patterns with high precision. These technologies are now reshaping the way matches are studied, bringing objectivity and real-time analysis into the decision-making process.

This paper presents a complete pipeline for automated football match analysis, focusing on object detection, tracking, motion estimation, spatial transformation, and tactical visualization. The detection backbone used in this project is the YOLOv5 object detection model [1], which provides real-time performance and high detection accuracy for identifying players, referees, and the football. Temporal consistency of these detections is maintained using ByteTrack [3], a high-performance multi-object tracker that ensures stable ID assignment across frames.

To account for dynamic camera motion common in broadcast footage, Lucas-Kanade optical flow [4] is used to estimate inter-frame movement and correct object positions accordingly. Player team classification is performed using KMeans clustering [5], based on color feature extraction from jersey regions. Ball possession is determined by computing spatial proximity between the ball and players' feet. Perspective transformation is applied to convert screen coordinates into real-world pitch dimensions, which allows for accurate estimation of player speed and distance traveled.

All techniques were tested using the *Football Players Detection* dataset hosted on Roboflow Universe [2]. This dataset contains 372 annotated images extracted from nine Bundesliga football matches. While the dataset is relatively small, it provides a meaningful kick-start for evaluating the complete football analysis pipeline. The images are labeled with bounding boxes for players, the ball, referees, and goalkeepers, making it suitable for training object detection models with fine-tuned class-level discrimination.

The proposed system demonstrates how deep learning and video analysis can be seamlessly integrated into a tool that assists coaches, analysts, and broadcasters in making informed, objective, and rapid decisions.

II. Related Work

In recent years, sports analytics—particularly in football—has gained momentum due to advancements in computer vision and machine learning. Several research initiatives and commercial tools have been proposed to automate the understanding of team strategies, player movements, and match outcomes. This section explores related work in core areas that underpin our system, including object detection, multi-object tracking, team identification, and tactical behavior analysis.

A. Object Detection in Sports Footage

Object detection is the cornerstone of automated football analysis. Traditional approaches such as HOG-SVM or Haar cascades were largely limited by their handcrafted feature design and poor generalizability [7]. With the advent of deep learning, Convolutional Neural Networks (CNNs) enabled

end-to-end learning for object detection tasks. Models such as Faster R-CNN [8], SSD [9], and YOLO [10] drastically improved performance in terms of both speed and accuracy.

In the context of sports, YOLO-based models have shown great potential due to their real-time performance. Jocher et al.'s YOLOv5 framework [1] has been widely adopted for detecting players, referees, and the ball in football broadcasts. The dataset used in this work is adapted from Roboflow Universe [2], which contains labeled frames from nine Bundesliga matches. Although the dataset contains only 372 images, it provides a representative variety of player appearances and camera angles, making it suitable for small-scale model training and testing.

B. Multi-Object Tracking and ID Consistency

Accurate tracking of players and the ball is essential for analyzing tactical behavior and possession. Earlier tracking systems relied on background subtraction and optical flow, which failed under occlusion or camera motion. Modern tracking systems such as SORT [11] and DeepSORT [12] leverage motion prediction and appearance features.

ByteTrack [3], used in this project, improves over previous approaches by associating low-confidence detections and retaining more candidates through a linear assignment strategy. This increases tracking robustness in crowded or low-resolution scenes, which are common in broadcast football videos.

C. Team Classification via Appearance-Based Clustering

Identifying team affiliations is critical in football analytics for ball possession, passing networks, and team heatmaps. Most commercial tools use jersey metadata and pre-registered rosters, which are unavailable in raw video-based setups.

This project adopts an unsupervised appearance-based method using KMeans clustering [5] on dominant jersey colors extracted from the upper body region. Similar color clustering approaches have been explored in basketball and hockey [13], though they often require controlled lighting conditions and high-resolution footage to be reliable.

D. Ball Possession and Tactical Behavior Analysis

Ball possession analysis is often approached through either spatial heuristics or learned behavior models. Proximity-based rules, like the one used in this project, define possession by calculating the distance between a player's foot and the ball [14]. More advanced models leverage pose estimation or sequential modeling (e.g., RNNs) to infer possession patterns, but these require dense annotations and larger datasets [15].

Tactical analysis systems like Tracab and Second Spectrum are widely used by professional teams, but these are commercial and rely on multi-camera setups and hardware tracking. In contrast, the system presented in this paper offers a software-only solution using single-camera input.

E. Challenges and Limitations in Current Research

Despite the progress, several challenges remain in developing generalizable football analytics systems:

- **Occlusion and Overlap:** Players frequently occlude each other, especially during set pieces, causing detection and tracking errors.
- **Camera Motion:** Broadcast footage often involves pan, tilt, and zoom, which makes motion modeling and object consistency difficult without stabilization.
- **Class Imbalance:** The ball is often underrepresented in datasets, leading to high false negatives unless augmented carefully.
- **Dataset Scarcity:** Publicly available football datasets are limited in scope, resolution, and diversity, making model generalization difficult.
- **Team Color Similarity:** Visual similarity between opposing team jerseys can confuse color-based clustering, especially under varying lighting.

These issues highlight the need for robust pipelines that integrate detection, tracking, motion modeling, and appearance-based reasoning, as demonstrated in this project.

III. Dataset Preparation and Preprocessing

To develop our football analysis system, we used video footage from the DFL - Bundesliga Data Shootout competition on Kaggle [16]. Unfortunately, the dataset is no longer available, but we obtained one of the competition videos from Google Drive and used it for testing.

Additionally, we used an annotated dataset from Roboflow titled "Football Players Detection" [17], which provides images of football players. This dataset consists of 372 total images for training, validation and testing.

A. Data Analysis and Observations

After exploring the dataset, we observed the following:

- The images were taken from **different football matches**, so the **jersey colors are not always the same** in each match.
- Each image contains **multiple players**, and some also include the **referee** and the **ball**.
- The **size and position** of players and the ball vary depending on the **camera angle and zoom level**. The camera often moves, which changes the view.
- There are **more people than just players** in the images. Some images also show **visitors, coaches, or staff** on the sidelines.

These observations helped us understand the challenges in detecting and tracking all relevant objects. It also highlighted the importance of focusing on the correct subjects while training the model.

B. Annotation

The dataset we used came with pre-annotated labels, so we did not apply any additional preprocessing before using it.

Each image was annotated using bounding boxes to mark the positions of objects of interest.

There are **three labeled classes** in the dataset:

- **Player**
- **Referee**
- **Ball**

The annotations were provided in the YOLO, COCO, Pascal VOC, PaliGemma format which lists the object class and bounding box coordinates normalized to the image size. We used the YOLO format as we are using a pre-trained yolo model.

C. Data Splitting

The dataset was already split into two parts when we downloaded it from Roboflow: a training set and a validation set. 80% of the images were used for training, 13% for validating and 7% for testing.

We used the training set to teach our object detection model to recognize players, referees, and the ball. The validation set was used to check how well the model performs on new, unseen images.

No extra changes were made to the split, and the original distribution was kept to ensure fair testing of the model's performance.

IV. Model Training

To build the object detection system for football analysis, we trained a pre-trained model, **YOLO**, using the labeled dataset. This section explains our training setup, model choices, dataset loading method, and training configuration.

A. Training Environment

We used Google Colab as our training environment. It offers free access to GPU resources, which significantly reduces training time and is well-suited for running deep learning models. Using Colab also allowed us to quickly experiment and test different model configurations in a convenient cloud-based setting.

B. Model Selection

Initially, we selected **YOLOv8** due to its strong performance in object detection tasks and its simple training pipeline. However, during our experiments, we observed that YOLOv8 had difficulty accurately detecting the ball in several frames. YOLOv8 is being compared to YOLOv5 rather than any other version of YOLO is that YOLOv5's performance and metrics are closer to YOLOv8's.

It is witnessed that YOLOv8 outperforms YOLOv5 for each RF100 category (shown in Figure 1) [18]. RF100 is a 100-sample dataset from the Roboflow universe, which is a repository of 100,000 datasets. However, we also found that in some cases, YOLOv5 outperforms YOLOv8 for object detection [19]. To address this issue, we switched to **YOLOv5**, which provided better accuracy to detect the ball. The final model used for training and evaluation was **YOLOv5x**, a larger variant of YOLOv5 designed for higher accuracy.

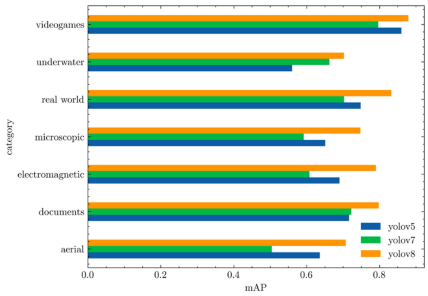

Fig. 1. Performance comparison between YOLOv5, YOLOv7 and YOLOv5 on the RF100 dataset.

C. Dataset Integration

We used the Roboflow platform to access and load the dataset directly into our Colab environment. Roboflow provides a simple interface for downloading datasets in multiple formats. For our use case, we selected the YOLOv5 format. This format includes the image files and their corresponding annotations, along with a configuration file compatible with the YOLO training pipeline.

D. Training Configuration

We trained our YOLOv5 model using the following configuration:

- **Model:** YOLOv5x
- **Image size:** 640 × 640
- **Epochs:** 100

These training parameters were selected on the basis of standard practices and available computational resources. The larger image size helps the model detect smaller objects like the ball, while 100 training epochs allowed enough time for the model to learn from the dataset.

V. Methodology

This section describes the components of the proposed football analysis pipeline. The full process consists of eight major stages that are executed sequentially to transform raw match footage into a tactically annotated output video. The overview of this pipeline is presented in Fig. 2.

A. Pipeline Overview

The entire video analysis system is designed as a modular pipeline with the following key stages:

1) **Video Frame Extraction:** The input football match video is read and split into individual frames.
2) **Object Detection:** A pre-trained YOLOv5 model is used to detect players, referees, and the ball in each frame.
3) **Object Tracking:** ByteTrack is applied to associate detected objects across frames with consistent IDs.
4) **Camera Motion Estimation:** Lucas-Kanade optical flow is used to estimate inter-frame camera motion, which is used to correct object trajectories.
5) **Team Classification:** Players are clustered into two teams using KMeans clustering on jersey colors.

6) **Ball Possession Assignment:** Possession is assigned based on the minimum distance between the ball and player foot positions.
7) **Video Annotation:** The annotated information (IDs, team colors, speed, possession) is drawn on the frames.
8) **Output Generation:** The frames are compiled into a final annotated video for visualization.

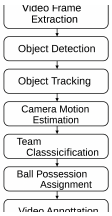

Fig. 2. Overview of the complete video analysis pipeline used for football match processing.

B. Object Detection and Initial Annotation

The object detection step plays a foundational role in the proposed football analysis system, enabling identification of players, referees, and the ball in each video frame. This process is performed using a trained YOLOv5 model [1], which was fine-tuned on a football-specific dataset consisting of 372 annotated images from nine Bundesliga matches [2]. Despite its limited size, the dataset includes all relevant object classes and provides an effective baseline for model training.

Detection Pipeline: The video is first divided into frames using OpenCV. These frames are then processed in batches to reduce memory overhead and increase computational efficiency. A confidence threshold of 0.1 is used during inference to ensure high recall.

The core detection function is summarized as follows:

- Frames are passed in batches to the YOLOv5 model.
- The model returns bounding boxes and class labels (player, referee, goalkeeper, ball).
- Detected objects are converted to the `Supervision` library's format using `from_ultralytics()` for downstream compatibility.
- A custom rule is applied to treat the goalkeeper class as a player to simplify later team analysis.

Tracking Integration: Detected objects are tracked across frames using the ByteTrack algorithm [3], which performs frame-to-frame data association via a linear assignment algorithm. ByteTrack efficiently handles occlusion and appearance changes by considering even low-confidence detections during tracking.

Class-wise Segmentation: For each frame, the objects are stored in a structured dictionary:

- `tracks["players"]`: Contains tracked player bounding boxes indexed by track ID.
- `tracks["referees"]`: Stores referee positions.
- `tracks["ball"]`: Contains ball detections with fixed index (since typically only one ball exists).

Output Annotation: At this point, the system holds positional data (bounding boxes) for all core entities in every frame. These are visualized using annotated bounding boxes during the final output rendering stage.

Fig. 3. Raw input frame extracted from the match video.

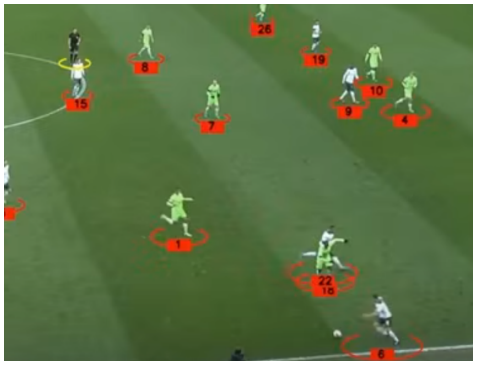

Fig. 4. Output frame after applying YOLOv5 detection and ByteTrack tracking. Players, referees, and the ball are annotated with bounding boxes and IDs.

C. Team Assignment Using KMeans Clustering

Identifying player teams from match footage is a crucial component for tactical analysis, possession tracking, and visualization. Unlike systems that rely on jersey metadata or player registration data, the proposed method uses an **unsupervised machine learning approach** based on KMeans clustering to classify players into two teams using visual features alone.

Color-Based Clustering: Each player's jersey is a strong visual indicator of their team identity. The method extracts the upper half of each player's bounding box (typically corresponding to the torso region), which is expected to contain the most consistent team-specific color information. This region is then reshaped into a 2D array of RGB pixel values to serve as input for clustering.

KMeans Algorithm: KMeans is an unsupervised clustering algorithm that partitions n observations into k clusters by minimizing the *within-cluster sum of squares (WCSS)*. In this project, $k = 2$ is used to separate players into two opposing teams. The algorithm follows these steps:

1) Randomly initialize $k = 2$ cluster centroids in RGB color space.

2) Assign each RGB pixel to the nearest centroid using Euclidean distance.
3) Update centroids by computing the mean of all pixels assigned to each cluster.
4) Repeat assignment and update steps until convergence.

The color cluster that dominates the corners of the cropped player patch is assumed to represent the background or non-player pixels (e.g., grass, crowd), while the other cluster is considered the actual player jersey color. This assumption helps isolate the dominant jersey color for each player.

Clustering Process: The extracted dominant colors of all players from the first frame are passed into a second instance of KMeans to perform team-level clustering. The output cluster centroids are stored as representative colors for Team 1 and Team 2.

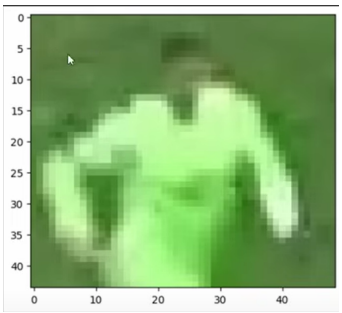

Fig. 5. Example of cropped player torso region before clustering.

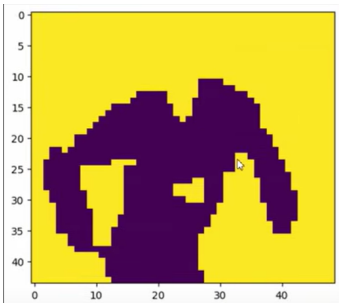

Fig. 6. Second example of a player's jersey region used as KMeans input.

Team Prediction: After team-level clusters are formed, the color centroid of each new player is compared to the cluster centroids using the trained KMeans model. The nearest cluster is assigned as the player's team ID. For computational efficiency and ID consistency, already classified players are cached using a dictionary keyed by player ID.

Model Integration: This color-based clustering approach avoids the need for manual annotation or jersey metadata, making it flexible across different match videos. It is also computationally lightweight, requiring only unsupervised clustering on a few RGB features, which makes it highly suitable for real-time or offline batch processing.

Fig. 7. Clustering output showing players color-coded by assigned team (frame 1).

D. Ball Position Interpolation

In football analysis, the ball is the most critical object for deriving tactical information such as possession, pass trajectories, and goal events. However, due to its small size, fast movement, and frequent occlusion by players, even advanced object detectors like YOLOv5 tend to miss the ball in certain frames. To ensure the continuity and temporal consistency of ball tracking, we employ a post-processing step using **interpolation** to estimate missing ball positions across frames.

Problem Statement: The YOLOv5 detector occasionally fails to detect the ball in frames where:

- The ball is partially or fully occluded.
- The motion blur makes detection uncertain.
- The ball blends into the background (e.g., white ball on a bright line).

Such missing detections result in discontinuous tracking data, which disrupts subsequent tasks like possession assignment, trajectory analysis, and speed estimation. To overcome this, we use a data interpolation technique to reconstruct ball positions in frames where detections are missing.

Interpolation Algorithm: The bounding boxes of the ball (in x_1, y_1, x_2, y_2 format) are extracted from the tracking dictionary and stored as rows in a Pandas DataFrame. This time-series data is then passed through two key steps:

1) **Linear Interpolation:** Missing values (NaNs) in each column are estimated using linear interpolation, assuming that ball movement follows a roughly linear path between two known positions.
2) **Backward Fill:** For leading NaNs (i.e., frames at the beginning where no data exists), the nearest available value is propagated backward using backward fill (bfill).

This process effectively "fills in" missing ball positions with smoothed values, producing a continuous motion path even when the detector fails momentarily.

ML Perspective: Although interpolation is not a learning algorithm per se, it is a **data imputation technique**, widely used in time-series ML pipelines [20]. It helps maintain the temporal structure of the data and avoids introducing noise

due to missing detections. In computer vision applications, particularly in sports analytics, it is common to apply such techniques for object recovery during occlusions or failed detections [21]. Interpolation plays a crucial role in stabilizing downstream logic such as possession detection or velocity estimation [22]. It ensures that brief failures in the object detection stage do not lead to downstream inconsistency, thereby improving the robustness of the overall system.

Fig. 8. Visualization of interpolation techniques. Box1 and Box4 detects the ball and the middle 2 boxes are filled in by interpolation

E. Player–Ball Assignment and Possession Estimation

Determining which player is in control of the ball at any given time is crucial for possession statistics, tactical insights, and event detection in football analytics. In this work, a geometric proximity-based approach is used to assign ball possession frame-by-frame.

Proximity-Based Possession Logic: For each frame, the center of the ball bounding box is computed. Each player's foot position is approximated by taking either the bottom-left or bottom-right corner of their bounding box. The distance from both corners to the ball's center is calculated using Euclidean distance, and the smaller of the two values is selected as the player's proximity measure.

The player with the smallest distance to the ball—within a predefined threshold—is assigned as the ball possessor for that frame. If no player is close enough (i.e., all distances exceed the threshold), possession is marked as undefined, and the system retains the previously known possessing team for continuity.

Temporal Possession Tracking: For each frame, the ID of the possessing player and their team is recorded in a time-series list. This list is later used to generate frame-level ball control statistics for both teams.

Parameters and Thresholds: The maximum allowed distance between the player's foot and the ball for possession assignment is empirically set to 70 pixels. This value balances sensitivity and robustness in the context of standard broadcast video resolutions.

F. F. Camera Motion Estimation Using Optical Flow

Football videos captured from broadcast feeds often involve camera movement such as panning, zooming, or shaking. These movements introduce artificial shifts in object positions, which can mislead tracking algorithms and distort subsequent computations like distance traveled, speed, and team heatmaps. To address this issue, we introduce a camera motion compensation step that estimates the camera's movement per frame and adjusts all object positions accordingly.

Lucas-Kanade Optical Flow: To estimate the motion of the camera, we use the Lucas-Kanade optical flow algorithm [4], a classical computer vision method that assumes consistent motion across small patches of pixels. This method works by tracking the movement of "good" feature points—such as corners or high-texture regions—between consecutive frames.

The algorithm computes the optical flow $\vec{v} = (u, v)$ at each pixel by solving:

$$\nabla I \cdot \vec{v} + I_t = 0$$

where ∇I is the spatial image gradient and I_t is the temporal derivative. By assuming constant motion in a local neighborhood, the method solves this equation in a least-squares sense over small windows.

Implementation Details: We use OpenCV's implementation of Lucas-Kanade with pyramidal refinement and the following parameters:

- Window size: 15×15
- Pyramid levels: 2
- Termination criteria: max 10 iterations or $\epsilon < 0.03$

To ensure that the selected feature points are stable across frames, we apply a binary mask that focuses on the edges and central strip of the image. These regions are less likely to be occluded by players and provide more consistent camera reference points.

Motion Vector Computation: For each frame, the movement of all tracked features is computed using the Pyramidal Lucas-Kanade algorithm. The feature pair (old point and new point) with the largest motion is used to estimate the global shift in X and Y directions. If this motion exceeds a predefined threshold (5 pixels), it is treated as true camera movement.

Track Position Adjustment: The estimated movement vector is subtracted from all tracked object positions, resulting in motion-compensated coordinates:

$$\text{adjusted\_position} = \text{original\_position} - \text{camera\_motion}$$

This step ensures that only real-world movements (e.g., players running) are considered in downstream analytics like distance traveled or possession zones.

Benefits: By removing the influence of camera dynamics, this step improves the accuracy of all trajectory-based analytics in the system. It also enhances visual stability in annotated videos and improves interpretability of player motion heatmaps.

G. Perspective Transformation to Real-World Coordinates

In standard broadcast football videos, the camera captures the field from an oblique angle, resulting in distorted pixel coordinates that do not correspond directly to real-world positions. For an accurate measurement of movement, speed

and spatial relationships between players, it is essential to convert the pixel coordinates to a bird's eye (top-down) view of the pitch. This is achieved through perspective transformation.

Conceptual Basis: A perspective transformation, or homography, is a projective mapping that transforms points from one plane to another using a 3×3 matrix. It is commonly used in computer vision to remove perspective distortion and map points from camera space to a world reference frame [23].

The transformation is computed from four known point correspondences between:

- **Pixel coordinates:** Specific points on the field in the video frame (e.g., corners or intersection of lines).
- **Target field coordinates:** The corresponding locations in the real-world pitch layout (e.g., in meters).

OpenCV's 'cv2.getPerspectiveTransform()' is used to estimate the transformation matrix H that satisfies:

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = H \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \Rightarrow (x_{new}, y_{new}) = \left(\frac{x'}{w'}, \frac{y'}{w'} \right)$$

Implementation Details:

The transformation matrix is computed in the constructor of the 'ViewTransformer' class. The 'pixel vertices' are manually annotated points in the video frame corresponding to known field markers (center of the midfield). The 'target vertices' are real-world positions in meters assuming a pitch width of 68 meters and a length of 23.32 meters (camera view portion only).

The transformation is applied as follows:

1) Each object's adjusted position is checked to ensure it lies within the transformable polygon (using 'cv2.pointPolygonTest').
2) Valid positions are reshaped and passed to 'cv2.perspectiveTransform()'.
3) The resulting (x, y) positions are recorded as real-world coordinates.

Integration into Tracking: The transformed coordinates are stored under 'position transformed' for each player, referee, and ball object. These are later used for:

- Speed and distance computation in meters per second.
- Zone-based tactical analysis.
- Bird's-eye view rendering of the match.

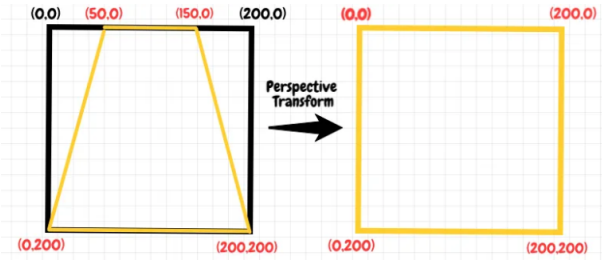

Fig. 9. Top: pixel-space input with key field points marked. Bottom: transformed field view showing object positions in real-world coordinates.

Advantages: Perspective transformation allows the use of real-world metrics (e.g., meters) rather than abstract pixel units. This is particularly useful for fair comparisons between players, tactical formations, and cross-match analytics.

H. Speed and Distance Estimation

One of the key benefits of transforming object positions into real-world coordinates is the ability to quantify performance metrics such as player speed and distance covered. This is particularly valuable for analyzing player stamina, press intensity, and overall match dynamics. The proposed system estimates these metrics using motion vectors derived from transformed positional data.

Distance Calculation: The system samples each tracked player's position over a sliding window of $N = 5$ frames. For each player track, the transformed position at the beginning and end of the window is retrieved, and the Euclidean distance is computed using the real-world (x, y) coordinates (in meters). This distance is accumulated over the match to compute total distance covered.

Speed Calculation: To compute instantaneous speed, the distance covered over a time window is divided by the time elapsed:

$$\text{Speed} = \frac{\text{Distance}}{\text{Time}} \quad \text{(in m/s)} \quad \Rightarrow \quad \text{Speed (km/h)} = \text{Speed (m/s)} \times 3.6$$

The frame rate is assumed to be 24 frames per second, as defined in the class constructor.

Data Storage and Annotation: The calculated speed and cumulative distance are stored in the track dictionary for each player. These metrics are drawn directly onto the annotated output video, placed slightly below the player's bounding box. This allows visual assessment of player workload in real-time.

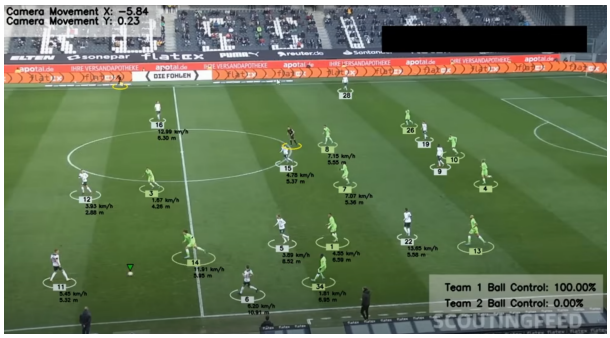

Fig. 10. Final annotated output showing player bounding boxes, team assignments, speed (km/h), and distance (m) covered.

Benefits and Applications: By providing per-frame speed and cumulative distance, the system can highlight high-intensity players, identify fatigue patterns, and inform tactical substitutions. These metrics are also valuable for training analysis and player scouting.

VI. Conclusion

The suggested football analysis pipeline effectively converts raw match video into a tactically enriched annotated video

based on a well-integrated chain of computer vision and machine learning modules. The pipeline runs in a modular way—beginning with video frame extraction and object detection with YOLOv5 and then with strong object tracking with ByteTrack. It proceeds to estimate camera motion to rectify tracking paths and utilize an unsupervised KMeans clustering algorithm to effectively classify teams by jersey color.

The system's capacity to assign ball ownership based on distance calculations and interpolation to manage missing detections provides continuity and consistency in analysis. In all, the approach provides an effective and affordable way to automate football video analysis while providing in-depth tactical knowledge independent of metadata or manual annotation. The low overhead and scalability of the system allow it to be compatible with both real-time use and offline batch processing.

VII. Future Scope

The current football analysis pipeline lays a strong foundation, but there's significant potential for growth and expansion. Here are some key directions for future development:

1) **Action Detection Using Deep Learning**
   While the system currently detects players and the ball, we can take it further by identifying specific in-game actions like:
   - Passing
   - Shooting
   - Tackling

   These actions can be recognized using advanced deep learning models such as I3D (Inflated 3D ConvNet), SlowFast Networks, or LSTM-based models that work on sequences of frames. Frameworks like PyTorch, TensorFlow, or OpenCV with deep learning support would be ideal for this task. This would unlock a deeper understanding of game strategy and flow.
2) **Better Player Identification**
   To pinpoint which player is performing a specific action, jersey number recognition is essential. This can be achieved through:
   - Tesseract OCR
   - YOLOv8 integrated with OCR plugins
   - EasyOCR (a deep learning-based OCR library)

   By pairing object detection with OCR, we can accurately track player identities and analyze individual performance over time.
3) **Ball Trajectory Tracking in 3D**
   Currently, the ball is tracked in 2D. Upgrading to 3D tracking will enable more precise insights. This could be achieved using:
   - 3D pose estimation models like MediaPipe, OpenPose 3D, or DeepLabCut
   - Multi-camera setups to estimate spatial depth

   With this, we can analyze things like whether the ball crossed the goal line, how it curved in mid-air, or its exact speed during a play.
4) **Real-Time Feedback System**
   Right now, the system processes video offline. The next step is to move toward real-time processing using:
   - Optimized models (e.g., TensorRT, ONNX)
   - Edge computing devices like NVIDIA Jetson or Coral TPU
   - Fast pipelines built with OpenCV and Python multiprocessing

   Real-time performance would offer practical benefits like:
   - Giving coaches instant feedback to guide decisions
   - Providing fans with live analytics such as speed, heatmaps, and passing stats
   - Enabling broadcasters to enhance the viewing experience
5) **Expansion to Other Sports**
   This pipeline isn't limited to football. With some tweaks to the dataset and retraining, it can be adapted for:
   - Basketball
   - Hockey
   - Handball

   Whether it's detecting a basketball shot or analyzing a hockey pass, tools like Detectron2, YOLOv8, and MMAction2 provide the flexibility needed for cross-sport adaptation.

REFERENCES

[1] G. Jocher, A. Chaurasia, J. Borovec, NanoCode *et al.*, "Yolov5 by ultralytics," *GitHub repository*, 2020, available at: https://github.com/ultralytics/yolov5.

[2] R. Universe, "Football players detection dataset," 2022, accessed: 2023-12-06. [Online]. Available: https://universe.roboflow.com/roboflow-jvuqo/football-players-detection-3zvbc/dataset/1

[3] Y. Zhang *et al.*, "Bytetrack: Multi-object tracking by associating every detection box," *arXiv preprint arXiv:2110.06864*, 2021.

[4] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proc. Imaging Understanding Workshop*, 1981.

[5] A. Vidhya, "Comprehensive guide to k-means clustering," 2019, accessed: 2023-12-06. [Online]. Available: https://www.analyticsvidhya.com/blog/2019/08/comprehensive-guide-k-means-clustering/

[6] P. Carling and A. Williams, "Performance analysis in football: A critical review and implications for future research," 2005.

[7] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," *Proc. CVPR*, 2005.

[8] R. G. Shaoqing Ren, Kaiming He and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[9] W. L. et al., "Ssd: Single shot multibox detector," *Proc. ECCV*, 2016.

[10] J. R. et al., "You only look once: Unified, real-time object detection," *Proc. CVPR*, 2016.

[11] A. B. et al., "Simple online and realtime tracking," *Proc. ICIP*, 2016.

[12] N. W. et al., "Deepsort: Deep learning for multiple object tracking," *arXiv preprint arXiv:1703.07402*, 2017.

[13] L. Brébisson and P. Vincent, "Appearance-based team recognition in sports videos," 2016, available: https://arxiv.org/abs/1602.07784.

[14] A. Fernandez, "Possession analysis in football using distance heuristics," 2020, blog Article: https://towardsdatascience.com/football-possession-analysis.

[15] C. Li and M. Sha, "Modeling ball possession with lstms for soccer match analysis," *Proc. NeurIPS Workshops*, 2019.

[16] J. Michalczyk, Maggie, M. Janetzke, M. M. Mücke, R. Holbrook, and U. Dick, "Dfl - bundesliga data shootout," https://kaggle.com/competitions/dfl-bundesliga-data-shootout, 2022, kaggle.

[17] Roboflow, "football-players-detection dataset," https://universe.roboflow.com/roboflow-jvuqo/football-players-detection-3zvbc , mar 2025, visited on 2025-04-05. [Online]. Available: https://universe.roboflow.com/roboflow-jvuqo/football-players-detection-3zvbc

[18] D. Reis, J. Kupec, J. Hong, and A. Daoudi, "Real-time flying object detection with yolov8," *arXiv*, vol. 2305.09972v2, 2023, accessed: 2025-04-05. [Online]. Available: https://doi.org/10.48550/arXiv.2305.09972

[19] A. Sharma and R. Kumar, "Comparative study of yolov5 and yolov8 for object detection in hazardous environments," *International Journal of Research and Scientific Engineering and Technology*, 2024, preprint, arXiv:2407.20872. [Online]. Available: https://arxiv.org/abs/2407.20872

[20] J. Brownlee, "How to handle missing data in machine learning: Time series imputation methods," 2020, available: https://machinelearningmastery.com/missing-data-in-time-series/.

[21] Y. Huang and J. Shi, "Interpolation-based occlusion handling in object tracking," *Proc. CVPR Workshops*, 2018.

[22] "Automatic player and ball tracking in broadcast soccer videos."

[23] R. Bansal, "Opencv perspective transformation," Online Article, 2020, available: https://medium.com/analytics-vidhya/opencv-perspective-transformation-9edffefb2143.