Video Summarization for Soccer Match

Review Report - 4

6th January 2025 - 22nd January 2025

Uchit N M	- PES1UG22CS661
Arya Gupta	- PES1UG22CS111

Faculty Mentors:

Prof. Prasad H B	prasadhb@pes.edu
Dr. Gowri Srinivasa	gsrinivasa@pes.edu
Prof. Preet Kanwal	preetkanwal@pes.edu

Table of Contents

List of Figures

Background :

PESU and the Ittiam team held their first meeting on September 17, 2024, during which the Ittiam team provided a broad overview of the expected work. They later shared a document outlining the scope on October 9, 2024. Based on this discussion, the PESU team initiated the project on October 1, 2024.

Our initial efforts centered on understanding the landscape of available data for analysis. This involved hands-on experience with frame extraction and bounding box analysis to process visual data, alongside an exploration of audio's role in summarization. Specifically, we examined transcription techniques and the use of timestamps to effectively align textual and visual elements. The key insights from this phase were compiled into a report and shared with the Ittiam team.

During the second meeting on October 24, 2024, the Ittiam team recommended a deeper dive into the technical aspects of summarization. In response, we explored two primary approaches: (1) leveraging transformer models for automated summary generation and (2) utilizing Graph Neural Networks (GNNs) to enhance the summarization process. Our findings were documented and submitted for review.

In the third meeting on December 2, 2024, the Ittiam team suggested focusing on soccer match summarization as an initial application. Following this discussion, we shared a report outlining a proposed framework for video summarization, integrating both audio and visual elements. As we did not receive any contrary feedback, we proceeded with our own structured approach to tackling soccer summarization while ensuring alignment with our proposed framework. The PESU team then began analyzing the fundamental components of the game, exploring relevant datasets, and identifying key characteristics essential for generating a meaningful soccer match summary. The details of which are provided next.

This report presents our progress from January 6th to January 22nd, 2024, focusing on understanding soccer match data and developing summarization techniques.

- **Section 1** provides an overview of the data, detailing the workflow, structure of game events, and the datasets used for analysis.
- **Section 2** discusses the implementation, including methods for tracking players, the ball, referees, and pitch. It also covers team classification, view transformation, soccer pitch configuration, and football object detection using the SoccerNet dataset. Additionally, it provides a code walkthrough, explaining the data exploration, preprocessing, model training, inference, and post-processing steps.
- **Section 3** highlights the challenges encountered during implementation and outlines potential future directions for improvement.

The report also includes links to relevant resources such as the GitHub repository, Google Drive folder, and Colab Notebook for further reference.

Our current work focuses on these main areas:

1. **Player Tracking**
    - Following player movements throughout the match
    - Using radar visualizations to show player positions
    - Identifying players and their roles on the field
2. **Ground Mapping**
    - Creating a detailed map of the playing field
    - Understanding different areas of the pitch
3. **Event Detection**
    - Finding key moments like goals and fouls
    - Identifying set pieces and other important plays

4. **Technical Implementation**
    - Using YOLOv8 for detecting players and objects
    - Applying UMAP and KMeans for grouping similar actions
    - Using color-based methods to analyze the objects in video
    - Automatically identifying team affiliations
    - Tracking both players and the ball
    - Marking important video sections for later use

Recap: A list of reports submitted by Team PESU to Team Ittiam

1. PESU_Ittiam_Report_01 (Initial Work)
2. PESU_Ittiam_Report_02 (Architecture based work)
3. PESU_Ittiam_Report_03 (Proposed Framework)

Section 1 : Understanding the Data

1.1) The workflow:

The workflow began with a detailed study of the game's structure, focusing on its patterns, key events, and rules. This was supplemented by the analysis of some full-length soccer matches to gain a deeper understanding of gameplay dynamics and the spatial arrangement of the field.

Key steps in the process included:

- Exploring relevant datasets to identify critical data points and suitable methodologies.
- Evaluating technologies such as player tracking and trajectory marking to enhance detection accuracy.
- Understanding the mechanisms to provide detailed insights into player and ball movements.

This improved detection accuracy and offered comprehensive insights into player and ball movements.

1.2) About the Game Events and the Dataset:

Association football, commonly referred to as football or soccer, is a team sport where two teams of 11 players compete on a rectangular field known as a pitch. Players primarily use their feet to control and propel the ball. A standard game lasts 90 minutes, divided into two 45-minute halves, with additional extra time (two 15-minute periods) if the match is tied at the end of regulation time.

Major Events in Football:

1. Goals: The primary objective, scored when the ball crosses the goal line within the frame of the goalpost.

2. Misconducts:

    - Yellow Card: A caution issued for unsporting behavior or rule violations.
    - Red Card: Issued for severe offenses, resulting in a player's expulsion from the match.

3. Kick-Off: The initial action to start each half or restart play after a goal.
4. Throw-Ins: Awarded when the ball crosses the touchline, allowing a player to throw the ball back into play.
5. Corner Kicks: Taken when the ball crosses the goal line, last touched by the defending team, but not resulting in a goal.
6. Indirect Free Kicks: Awarded for non-penal fouls or technical infringements, requiring the ball to touch another player before a goal can be scored.
7. Direct Free Kicks: Awarded for penal fouls, allowing a direct attempt at goal.
8. Penalty Kicks: Taken from the penalty spot following a foul within the penalty area, providing a one-on-one opportunity against the goalkeeper.
9. Dropped Ball: Used to restart play when the game is stopped for reasons other than fouls or goals.

Video analysis systems can detect and classify these moments using frame-by-frame analysis and trim the clips for highlight generation.

This enables the creation of concise match summaries, capturing key actions and moments, while reducing the need to view the entire 90-minute game.

1.3) The Datasets:

The datasets are well annotated across various classes and are captured from a wide bird's-eye perspective, enabling the tracking of patterns, the ball, and other key details of the pitch.

These are configured for use with a range of pre-trained open-source models, making them well-suited for the intended training and usage.

i. Smart Football: Object Detection Computer Vision Project

Link - [Smart Football: Object Detection Computer Vision Project]

The dataset contains 12,820 all annotated for ball, corners, referee, and players.

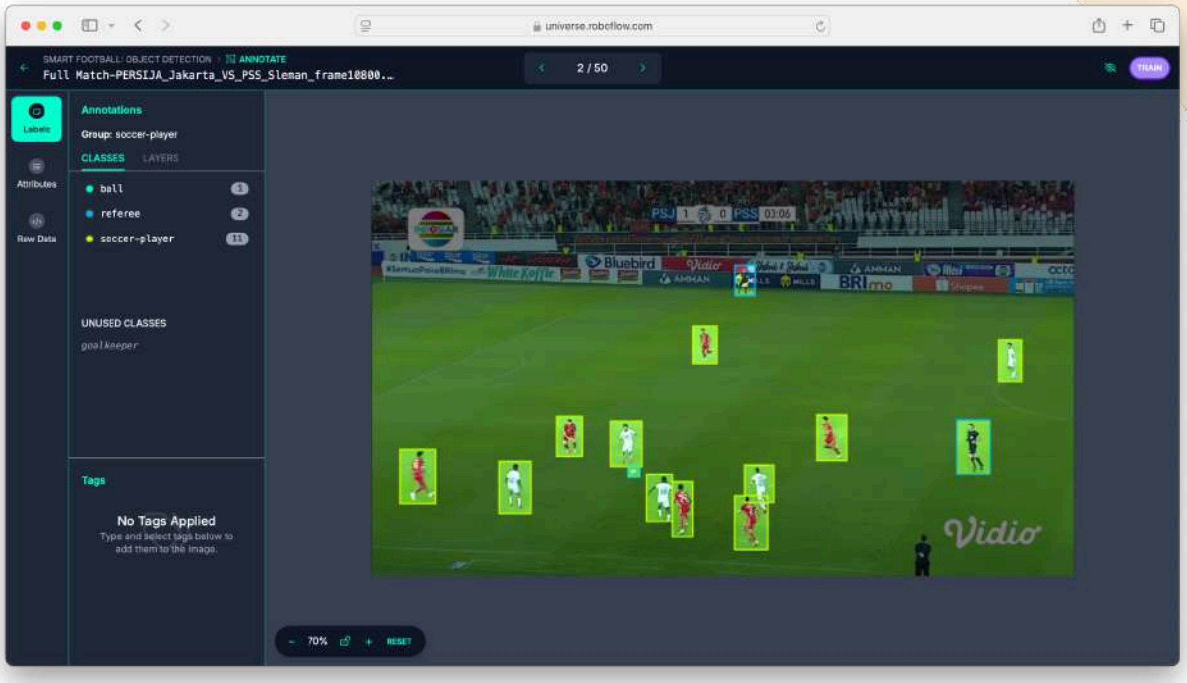
Dataset Split : Train Set - 88% - 11268 Images ; Valid Set - 8% - 1004 Images; Test Set - 4% ; 548 Images

Preprocessing :

- Auto-Orient: Applied
- Resize: Stretch to 640x640
- Auto-Adjust Contrast: Using Contrast Stretching
- Tile: 2 rows x 2 columns

Augmentations:

- Outputs per training example: 3
- Flip: Horizontal, Vertical
- 90° Rotate: Clockwise, Counter-Clockwise, Upside Down
- Rotation: Between -30° and +30°
- Cutout: 4 boxes with 20% size each
- Bounding Box: Flip: Horizontal, Vertical
- Bounding Box: 90° Rotate: Clockwise, Counter-Clockwise, Upside Down
- Bounding Box: Rotation: Between -15° and +15°
- Bounding Box: Brightness: Between -25% and +25%



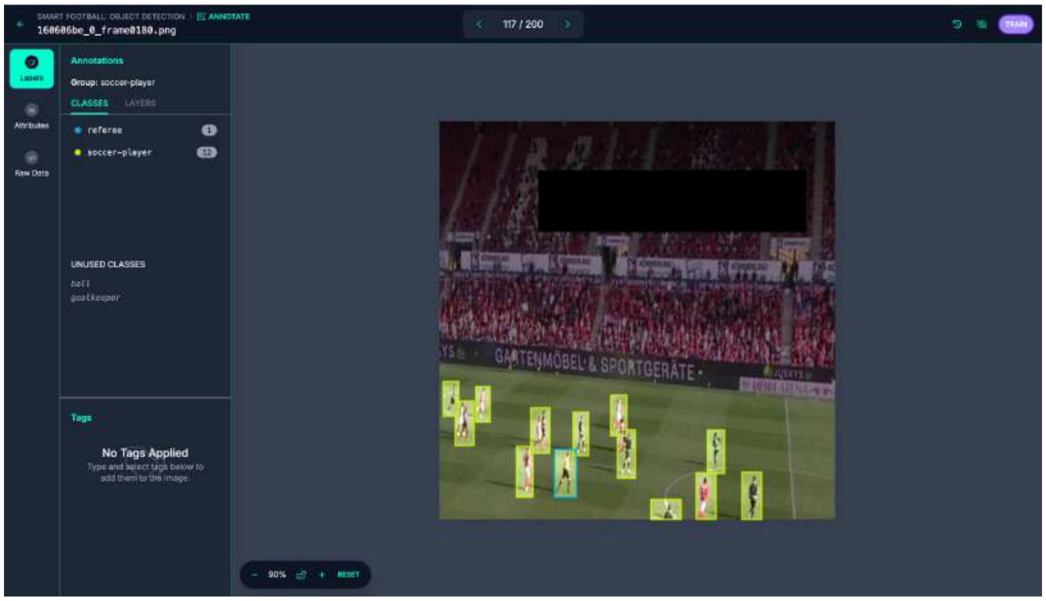


Fig 1.1.1 Smart Football: Object Detection Computer Vision Project Dataset

ii. Football-Player-Segmentation

Link : [Voxel51/Football-Player-Segmentation · Datasets at Hugging Face]()

The dataset is specifically designed for computer vision tasks related to player detection and segmentation in foot goalkeepers, and forwards, captured from various angles and distances.

The Dataset Structure contains two fields, detections and segmentations across 512 different samples

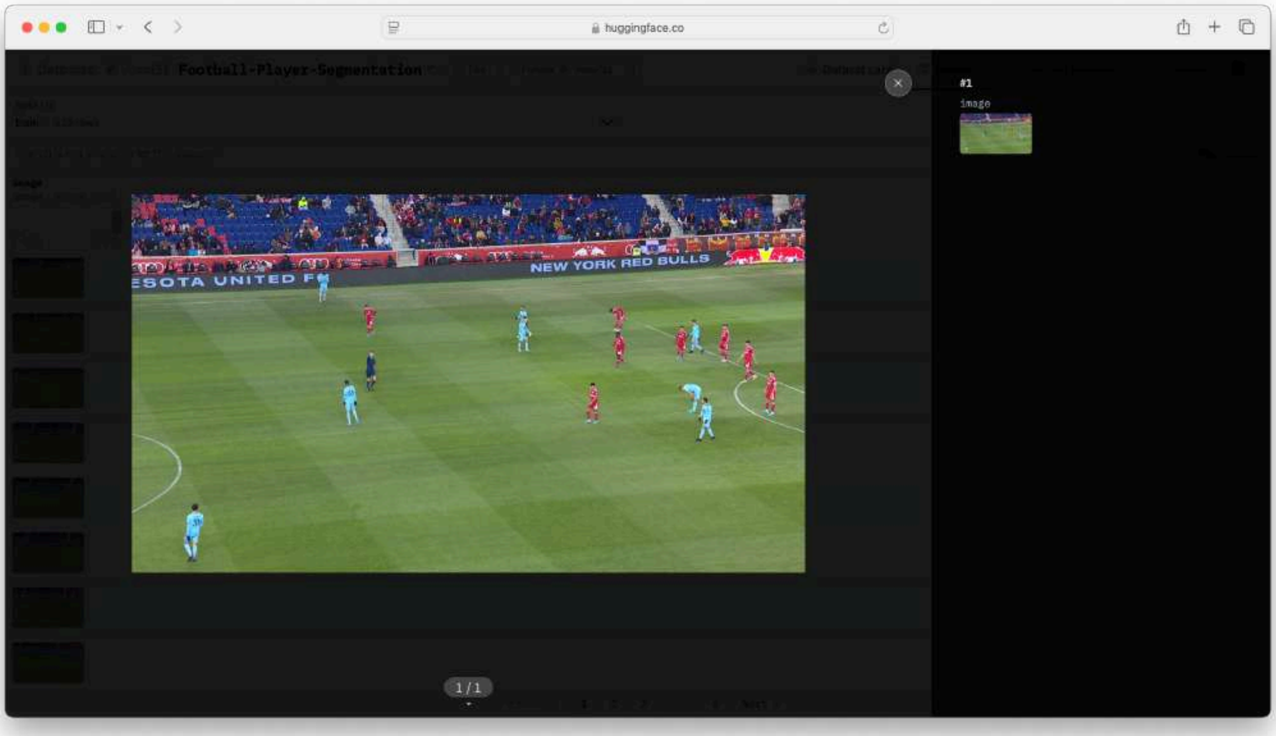


Fig 1.1.2 Football-Player-Segmentation Dataset

iii. football-players-detection Computer Vision ProjectWe (Used in Implementation 2.1)

Link : [football-players-detection Object Detection Dataset and Pre-Trained Model by Roboflow]

The dataset contains 382 all annotated for ball, corners, referee, and players.

Dataset Split : 298 Images - Train ; 59 Images - Validation ; 25 Images - Test

Preprocessing : Resize - Stretch to 1280x1280

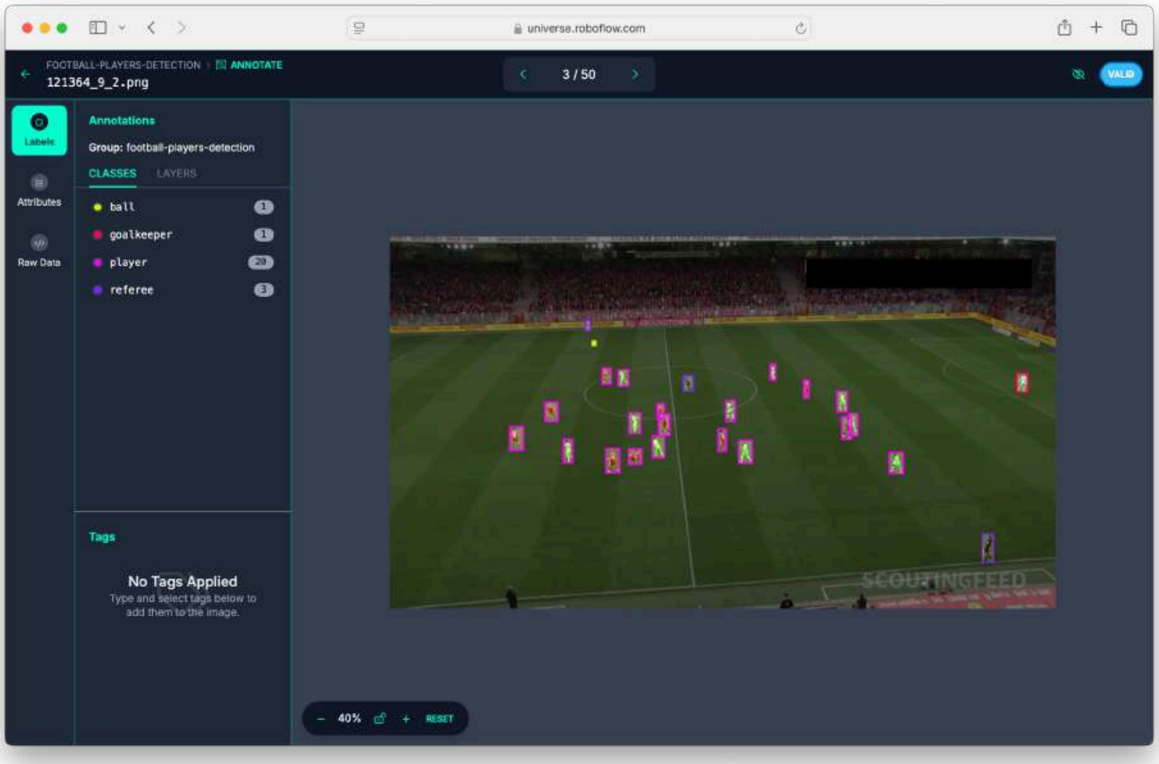


Fig 1.1.3 football-players-detection Computer Vision ProjectWe (Dataset Used in Implementation 2.1)

iv. Football Computer Vision Project

Link : [Football Object Detection Dataset by football detect]()

An annotated images dataset of 629 frame images.

Dataset Split : Train Set - 33% - 209 Images ; Valid Set - 33% - 210 Images; Test Set - 33% ; 210 Images

Preprocessing : Auto-Orient: Applied ; Resize: Stretch to 640x640 ; Modify Classes: 1 remapped, 2 dropped

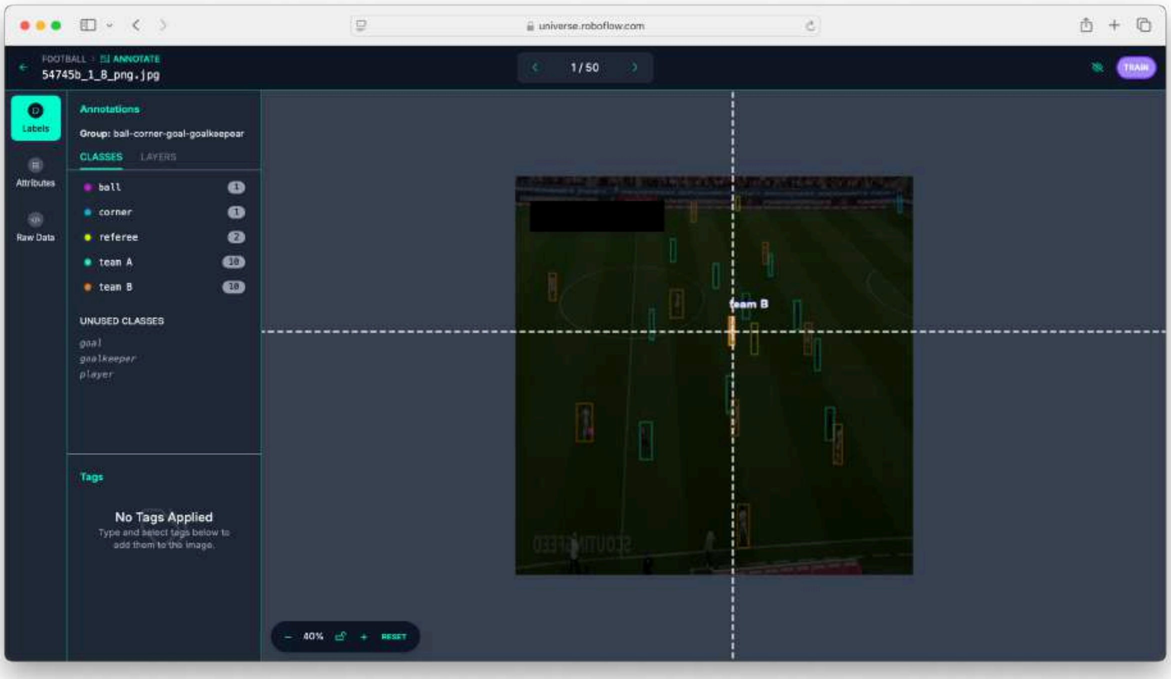


Fig 1.1.4 Football Computer Vision Project Dataset

v. Football Match Computer Vision Project

Link : [Football Match Object Detection Dataset and Pre-Trained Model by Test]() -

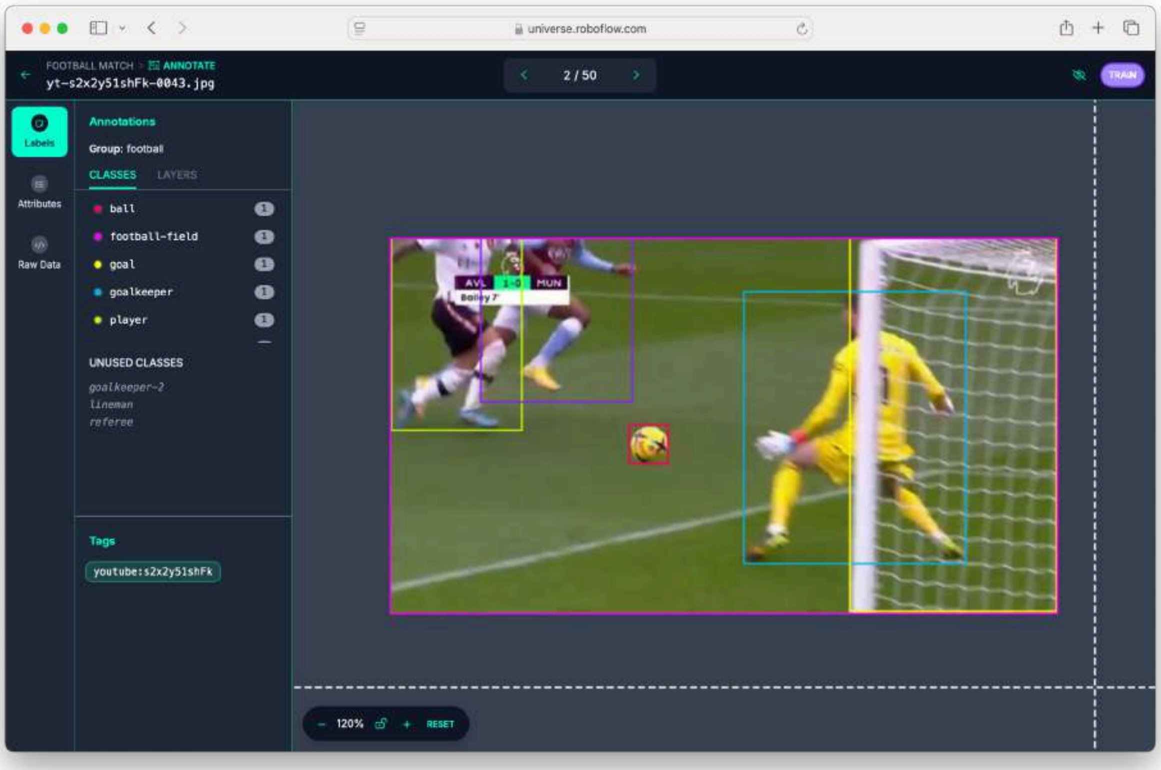
An annotated set of ball, field, goal, goalkeeper, player of 584 frame images.

Dataset Split : Train Set - 85% - 496 Images ; Valid Set - 10% - 59 Images; Test Set - 5% ; 30 Images

Preprocessing : Auto-Orient- Applied; Resize- Stretch to 640x640

Augmentations : Outputs per training example: 3 ; Flip: Horizontal, Vertical ; Grayscale: Apply to 15% of images

Fig 1.1.5 Football Computer Vision Project Dataset



vi. Football-field-detection (Used in Implementation 2.1)

Link : [football-field-detection Keypoint Detection Dataset and Pre-Trained Model by Roboflow]()

This Dataset is for field keypoint detection, has 317 augmented and key pointed images.

Dataset Split : Train Set - 80% - 255 Images ; Valid Set - 11% - 34 Images; Test Set - 9% ; 28 Images

Preprocessing : Auto-Orient- Applied; Resize- Stretch to 640x640

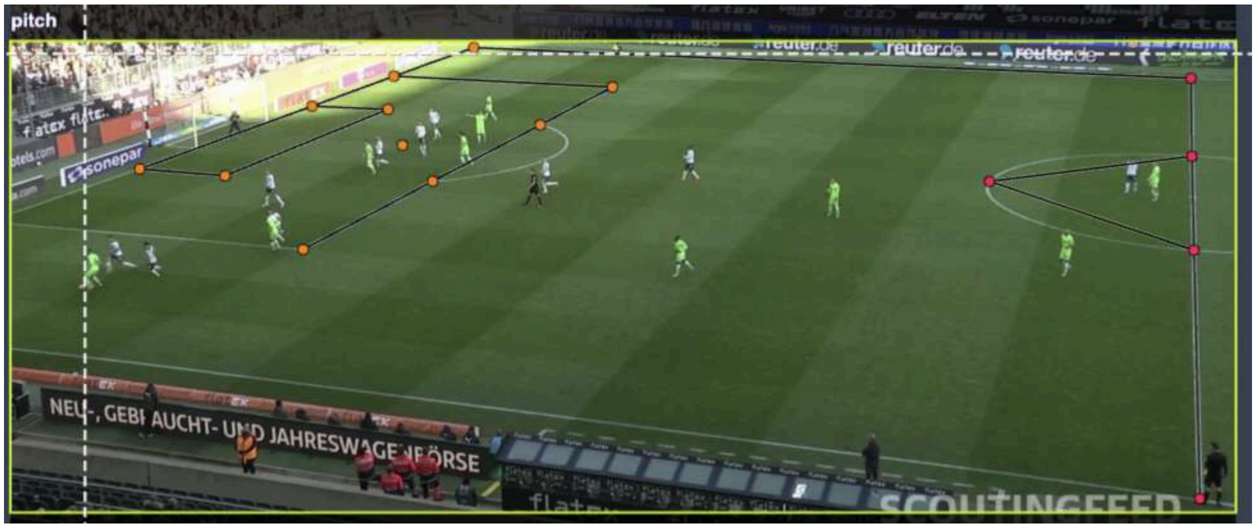


Fig 1.1.6. Football-field-detection (Dataset Used in Implementation 2.1)

Section 2 - The Implementation

2.1) Tracking Players, Ball, Referee, and Pitch

The primary objective of this system is to determine the positions of players, the ball, and referees on the soccer pitch. In addition to this, the system also identifies the team affiliation of each player and traces the movement of the ball across the field.

The system is designed to detect key events such as goals or fouls by processing each frame.

Approach

The approach involves analyzing soccer video clips to detect and track players, identify the ball, classify teams, and visualize the data using radar-like visualizations.

Key tasks include pitch detection, player detection, ball detection, player tracking, and team classification.

The methodology utilizes pre-trained models, such as YOLOv8 for object detection, and integrates advanced techniques like UMAP, KMeans, and SigLIP for feature extraction and clustering.

The system comprises three major components. The first involves fine-tuning models for detecting players, referees, and the ball. The second focuses on pitch detection and mapping. The third component leverages these models to implement the detection framework effectively.

This supports multiple modes, including radar visualization, which combines various detection tasks to provide a comprehensive overview of player positions and movements.

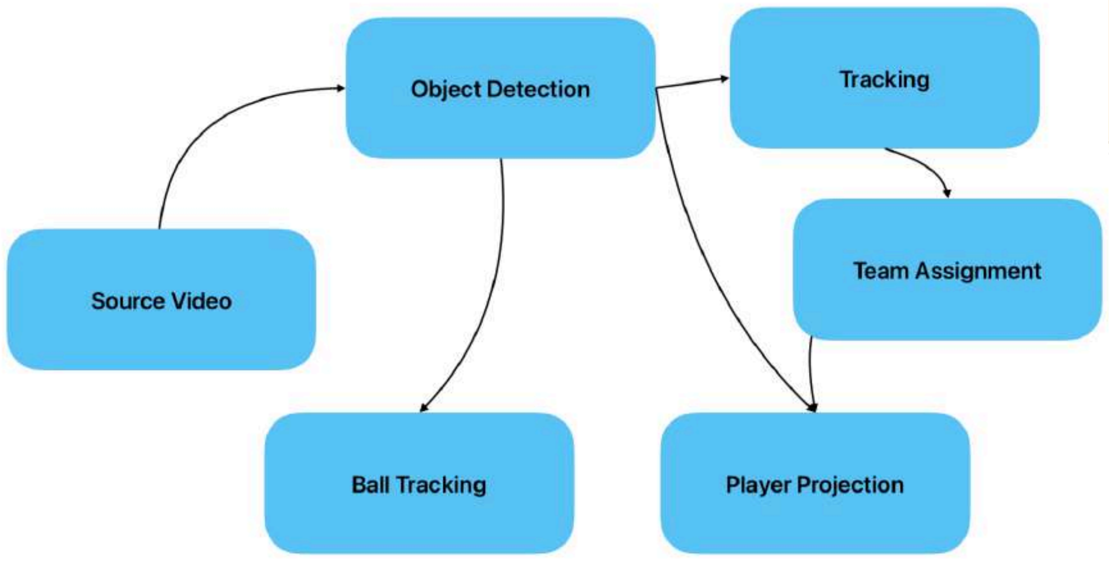


Fig 2.1.1. Model Architecture and Flow

The YOLOv8 model was fine-tuned on a custom-annotated dataset to detect objects like players, goalkeepers, referees, and the ball, The training was over 50 epochs in a batch size of 6 with 1280 image size.

The model's performance was evaluated using mean average precision (mAP) metrics, ensuring accurate detection across all object classes.

Video frames were extracted and processed through the trained model, with bounding boxes drawn around detected objects. A specific ball label was used to highlight and track the ball's movement across the video.

ByteTrack, supported by the Supervision library, was utilized to provide robust tracking capabilities for continuous player and ball identification across frames.

Team classification was based on the visual features of player kits. However, challenges arose due to varying lighting conditions and background colors resembling player uniforms.

To address these challenges, advanced embedding techniques like SigLIP and dimensionality reduction using UMAP were employed. KMeans clustering was then applied to classify player embeddings derived from cropped images, then successfully grouping players into two teams.

Goalkeepers were identified by calculating their proximity to team centroids, which were determined as the mean or median position of players within each team.

To overcome perspective distortions in video frames, homographic transformation methods were employed.

Key points from the soccer field, such as the penalty areas, goal lines, and center circle, were leveraged to map the angular perspectives to an aerial view.

Annotated data was used to enhance the mapping accuracy, and frames were stretched to improve the aspect ratio for visualization. Detection of pitch lines and field points further refined the perspective transformation.

This pipeline integrates object detection, tracking, classification, and homographic transformations to enable precise event detection, player tracking, and intuitive visualizations, providing a scalable solution for soccer analytics.

The Sports Model :

This model is designed to apply computer vision techniques to sports(soccer), with a focus on annotating and detecting players, the ball, and classifying player actions. The program is organized into distinct modules and classes, each responsible for specific tasks.

Soccer Annotators (sports/annotators/soccer.py)

The Soccer Annotators module includes several functions that facilitate the visualization of the soccer pitch. The `draw_pitch` function generates a soccer pitch with customizable dimensions, colors, and scale.

The `draw_points_on_pitch` function is used to place points on the pitch, which can highlight specific locations or movements.

Common Entities

The Common Entities module contains the `BallAnnotator` and `BallTracker` classes. The `BallAnnotator` class is responsible for annotating video frames by drawing circles of varying radius and colors to represent ball detections.

The `BallTracker` class tracks the soccer ball's position across video frames by maintaining a buffer of recent ball positions.

Team Classification (`sports/common/team.py`)

The Team Classification module handles the classification of teams based on player image data. It uses several libraries, including numpy, supervision, torch, and transformers.

The `TeamClassifier` class employs a pre-trained SiglipVisionModel for feature extraction, UMAP for dimensionality reduction, and KMeans for clustering.

View Transformation (`sports/common/view.py`)

The View Transformation module includes the `ViewTransformer` class, which is used for transforming points and images using a homography matrix.

Soccer Pitch Configuration

The Soccer Pitch Configuration is managed by the `SoccerPitchConfiguration` class, which is defined using the data class decorator. This class contains properties for the dimensions of the soccer pitch, including the width, length, penalty box, and goal box.

The Soccer AI Implementation:

This model detects players in each video frame, annotates their positions using bounding boxes, and yields annotated frames for visualization or further processing.

Similarly, the run_ball_detection function tracks the ball using a slicing mechanism to enhance accuracy and applies a buffer to smooth its trajectory across frames.

The run_team_classification function extracts player crops from video frames, employs a TeamClassifier to assign team affiliations based on visual features, and determines goalkeeper team IDs by calculating their proximity to team centroids.

For visualizing the game dynamics, multiple detection tasks including pitch keypoints, player positions, and team classifications are combined and analyzed.

It maps the detections onto a radar-like pitch view using a View Transformer and overlays positional data to provide a comprehensive overview of player movements.

Each of these methods processes video frames iteratively, enabling both real-time and batch processing.

The outputs include annotated frames or visualizations, which can either be displayed live or saved as a video for later analysis, such as when an event is detected using different methods, then the analysis video can be tagged at the timestamps and then trimmed as per requirements.

The Main Program Structure:

1. Import libraries and modules.
2. Define constants and configurations:
   - Paths to model files.
   - Class IDs for ball, goalkeeper, player, and referee.
   - Colors for annotations.
   - Configuration for the soccer pitch.
3. Define the Mode enum class with different operation modes.
4. Helper functions:
   - `get_crops`: Extracts crops from frames using bounding boxes.
   - `resolve_goalkeepers_team_id`: Assigns team IDs to goalkeepers based on proximity to team centroids.
   - `render_radar`: Renders radar views of player positions.
5. Functions for each mode:
   - `run_pitch_detection`: Detects and annotates soccer pitch keypoints.
   - `run_player_detection`: Detects and annotates players.
   - `run_ball_detection`: Detects, tracks, and annotates the ball.
   - `run_player_tracking`: Tracks players across frames.
   - `run_team_classification`: Classifies players into teams.
   - `run_radar`: Combines all detection tasks to render radar visualizations.
6. Main function:
   - Parses command-line arguments, Calls respective functions based on the mode.
   - Writes annotated frames to a video.
   - Displays annotated frames in a window.

Open source models utilized:

- YOLOv8 (Player Detection) - Detects players, goalkeepers, referees, and the ball in the video.
- YOLOv8 (Pitch Detection) - Identifies the soccer field boundaries and key points.
- SigLIP - Extracts features from image crops of players.
- UMAP - Reduces the dimensionality of the extracted features for easier clustering.
- KMeans - Clusters the reduced-dimension features to classify players into two teams.
- Supervision - Draw detections on an image or video, or count how many detections are in a zone

The Intermediate Outputs:

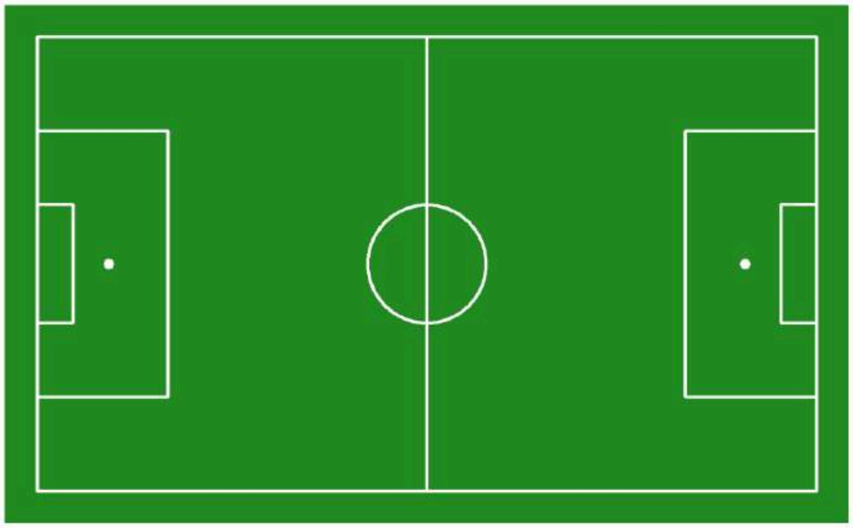


Fig 2.1.2 . The Pitch projection

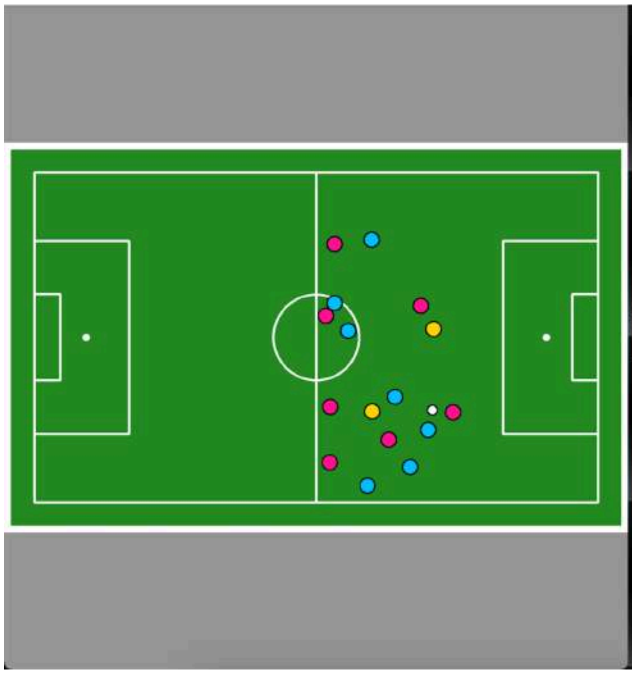
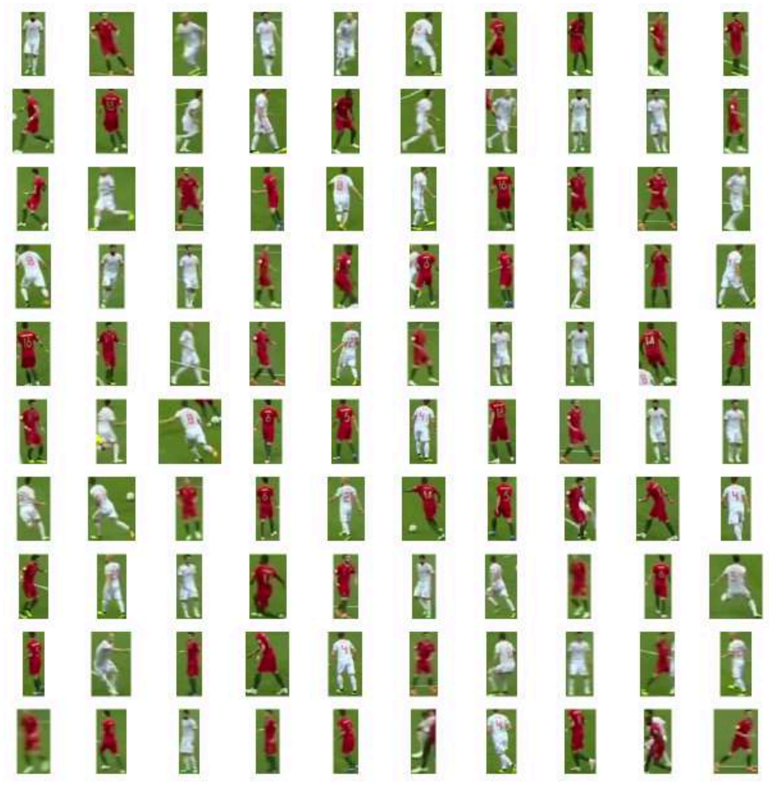
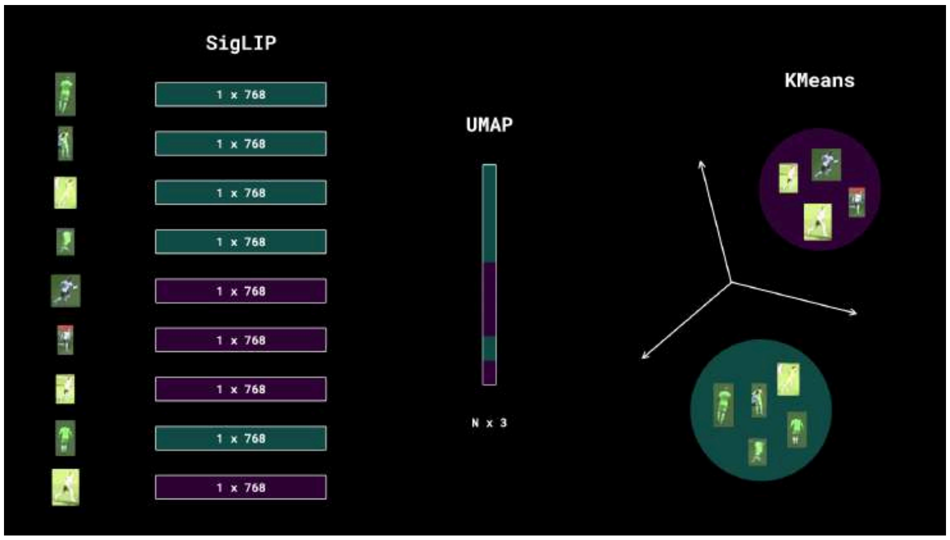


Fig 2.1.3 Radial Field view and Coverage



(a)



(b)

Fig 2.1.4 split players into teams using SigLIP, UMAP, KMeans (a), Datapoints of the Players (b), Process Flow

The Train result of the Models of Object Detection:

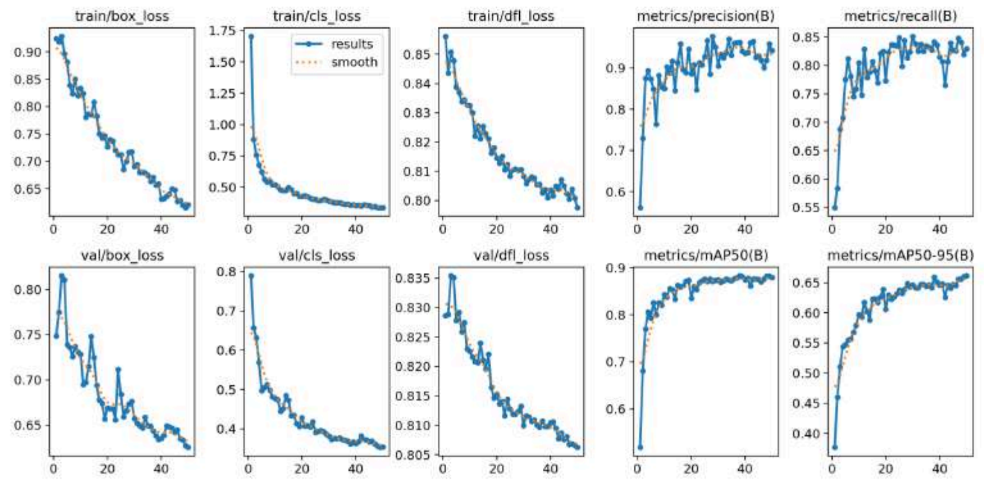


Fig 2.1.5 Metric graphs of the player detection model's training over 50 epochs.

The validation annotations of the player, referee and ball detections:

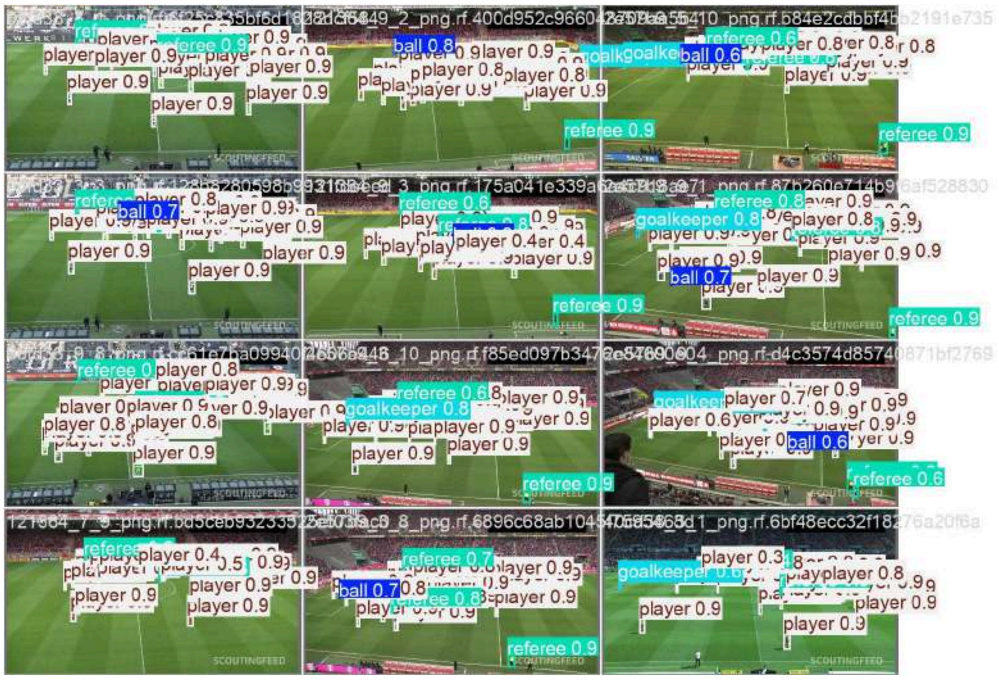


Fig 2.1.6 Detection over Validation set.

The Processed Outputs:

Link to Drive : Player_Ball_Refree_Pitch_detection

- The original clip of 30 sec of Portugal v Spain _ 2018 FIFA World Cup Match

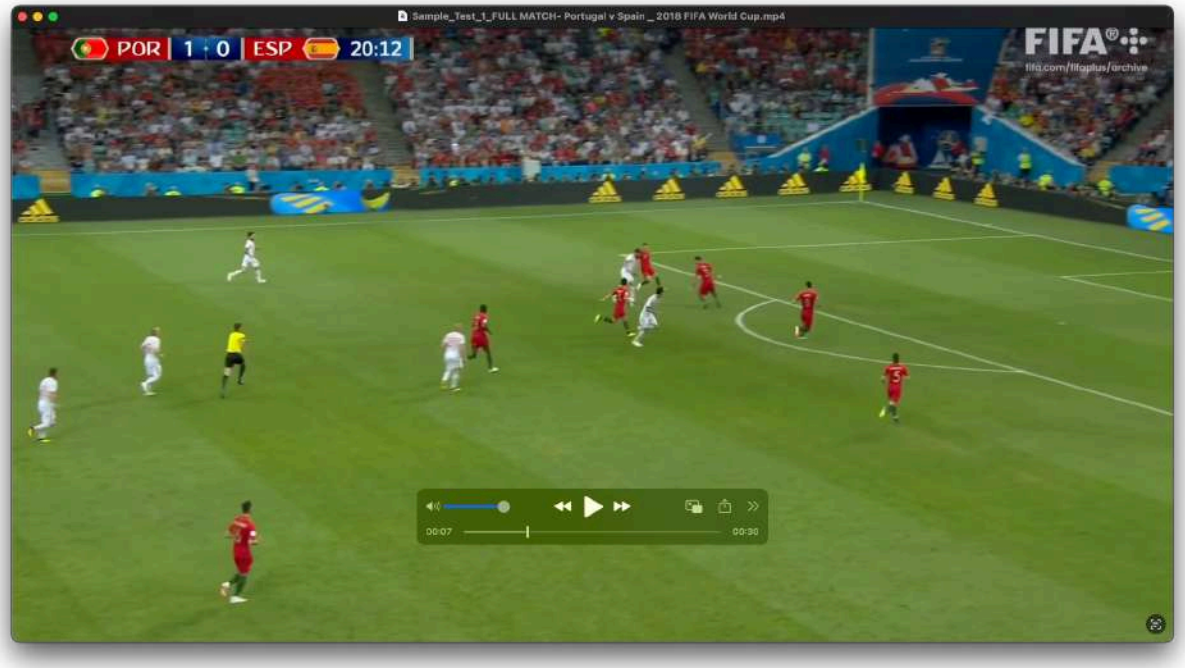

Sample_Test_1_FULL MATCH- Portugal v Spain _ 2018 FIFA World Cup.mp4
POR 1 : 0 ESP 20:12
FIFA
fifa.com/fifaplus/archive
00:07
00:30

Fig 2.1.7 Original Source Clip.

- Player Detection

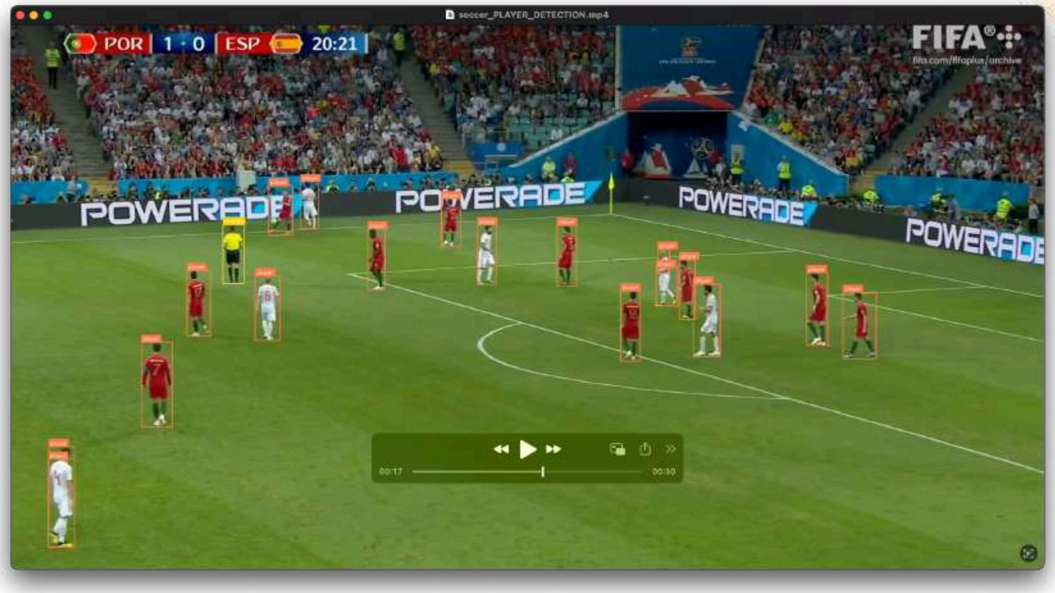


Fig 2.1.8 Player Detection.

- Player Classification

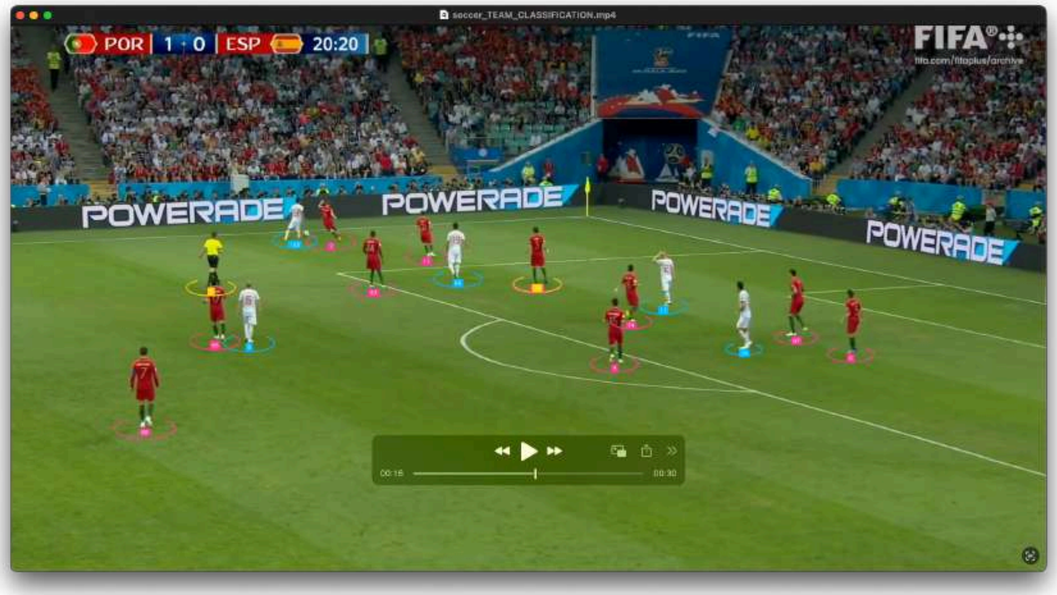


Fig 2.1.9 Player Classification Clip.

- Ball Tracking

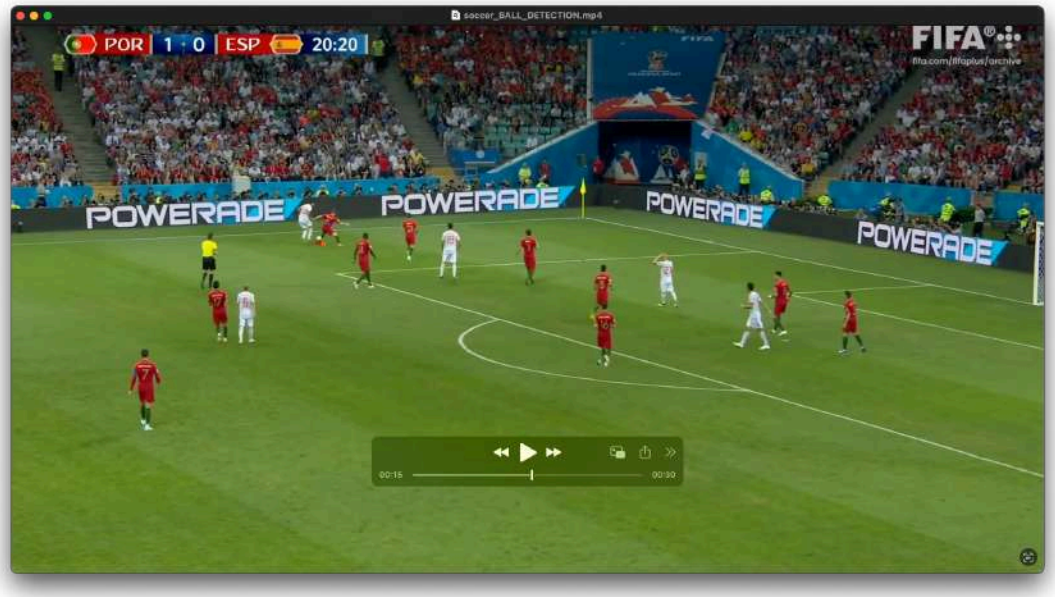


Fig 2.1.10 Ball Tracking.

- Radar view

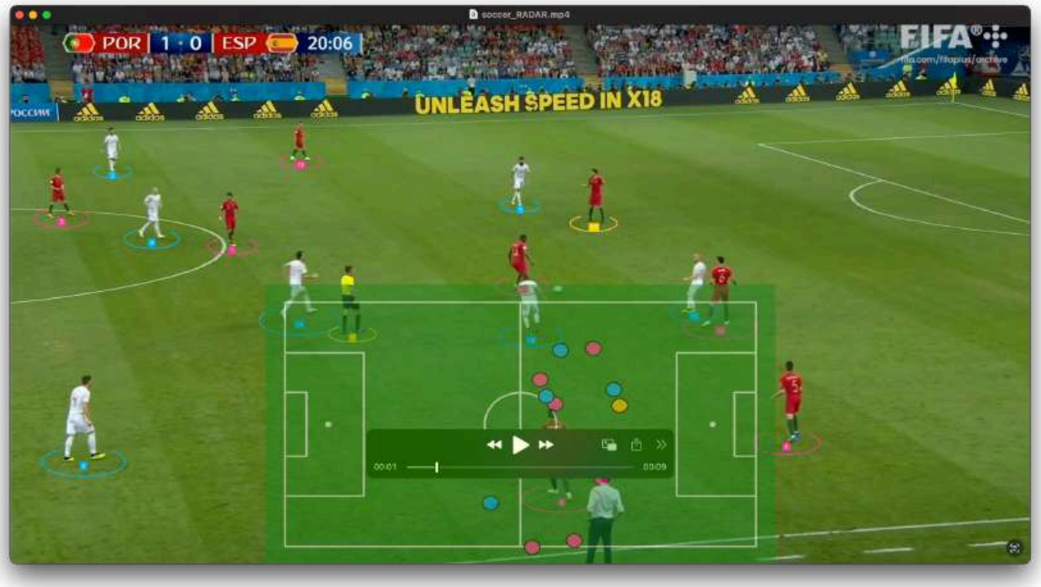


Fig 2.1.11 Radar View.

- Pitch Detection

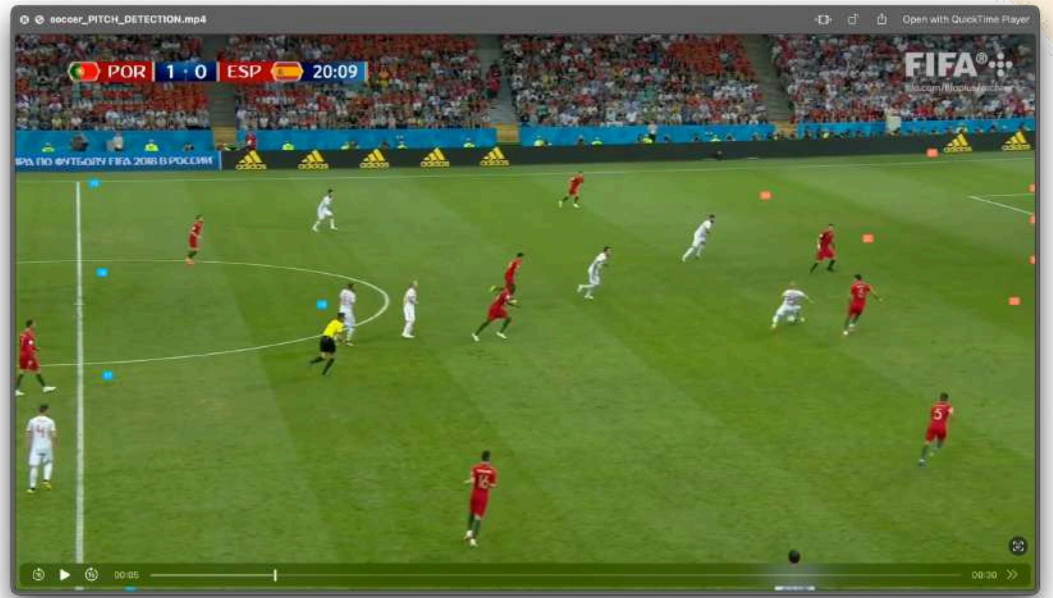


Fig 2.1.12 Pitch Detection.

- Project pitch lines on frame

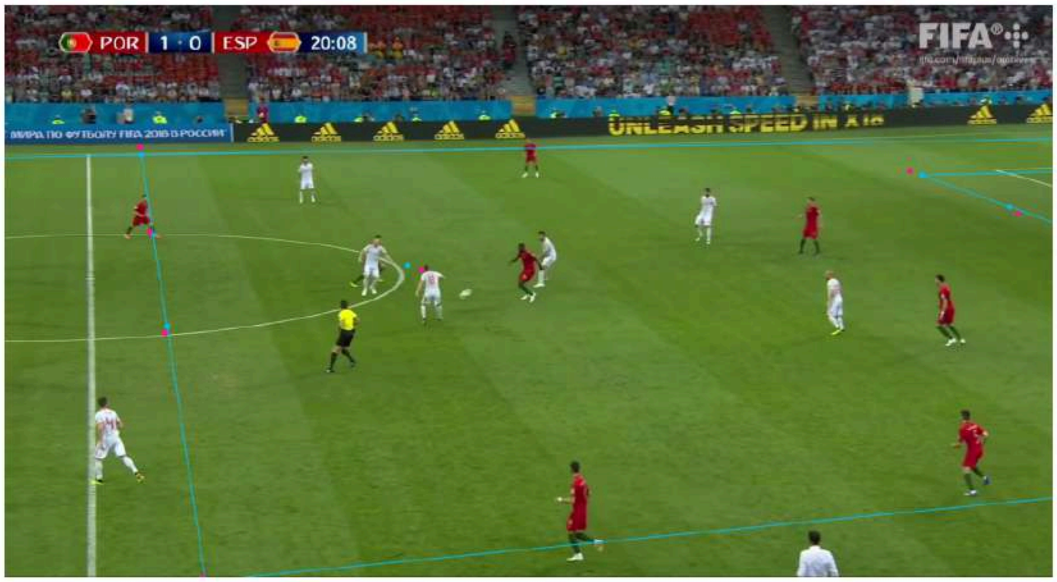


Fig 2.1.13 Pitch Lines Projection.

- Ball Trajectory:

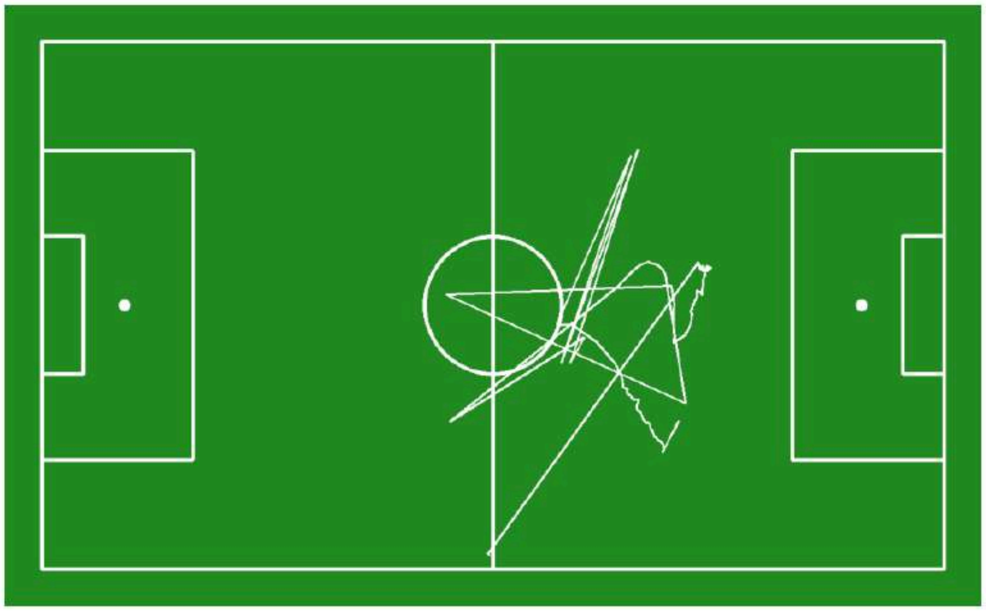


Fig 2.1.14 Depiction of Ball Trajectory

Links :

- Colab Link of the implemented program - Implementation_Jan_1
- Datasets Used for Implementation
    - football-field-detection Keypoint Detection Dataset and Pre-Trained Model by Roboflow
    - football-players-detection Object Detection Dataset and Pre-Trained Model by Roboflow
- Output Link - Player_Ball_Refree_Pitch_detection

Reference :

- Football AI Tutorial: From Basics to Advanced Stats with Python
- GitHub - roboflow/supervision: We write your reusable computer vision tools.

2.2) Football Object Detection

Dataset Used : SoccerNet

Soccer-Net

Approach

This part of the work focuses on analyzing football match videos using object detection algorithms, specifically YOLOv8, trained on the SoccerNet dataset for 10 epochs.

The model classifies objects into six classes:

1. Player,
2. Goalkeeper,
3. Ball
4. Main Referee,
5. Side Referee
6. Staff Members.

It enhances video analysis by determining the positions and roles of different entities on the pitch. The system also dynamically assigns team affiliations and positional labels during inference.

The first step involves extracting the grass color from the initial video frame. This is achieved by isolating green hues, masking out other elements, and computing the average color of the unmasked areas.

Next, the team kit colors are determined by extracting player bounding boxes, filtering out grass backgrounds, and calculating the average pixel colors. These colors are clustered using K-Means, where the centroids represent the kits for future comparisons. The system then assigns team labels ("Left" or "Right") based on the average x-axis positions of players in the frame.

For each video frame, the model performs inference to classify detected objects into predefined classes. Player kit colors are re-evaluated by removing grass backgrounds and calculating average colors. Players are categorized into teams by comparing their kit colors to the precomputed K-Means centroids. The positional labels ("Team Left" or "Team Right") are reassigned based on team labels derived from the first frame.

Goalkeepers are further classified as "GK Left" or "GK Right" depending on their position relative to the frame.

This approach integrates efficient color-based segmentation, clustering, and YOLOv8's robust object detection capabilities, creating a cohesive pipeline for detailed football match analysis.

Code Walkthrough

Stage 1 : Data Exploration

1. Environment Setup and Data Loading

   The dataset consists of a large collection of images and labels for football object detection. To optimize the loading process, We implemented a compressed data transfer approach. Directory structure was established for clear organization:

2. Dataset Structure Analysis

   A. Class Distribution

   Using YAML configuration, We identified 8 distinct classes in the dataset

   - Player team left (0)
   - Player team right (1)
   - Goalkeeper team left (2)
   - Goalkeeper team right (3)
   - Ball (4)
   - Main Referee (5)
   - Side Referee (6)
   - Staff Members (7)

   B. Dataset Split Analysis

   Analyzed the distribution of imaged across training, validation, and test sets:

   Distribution findings:

   - Training set: 10,212 images (80%)
   - Validation set: 1,276 images (10%)
   - Test set: 1,276 images (10%)

3. Class Frequency Analysis

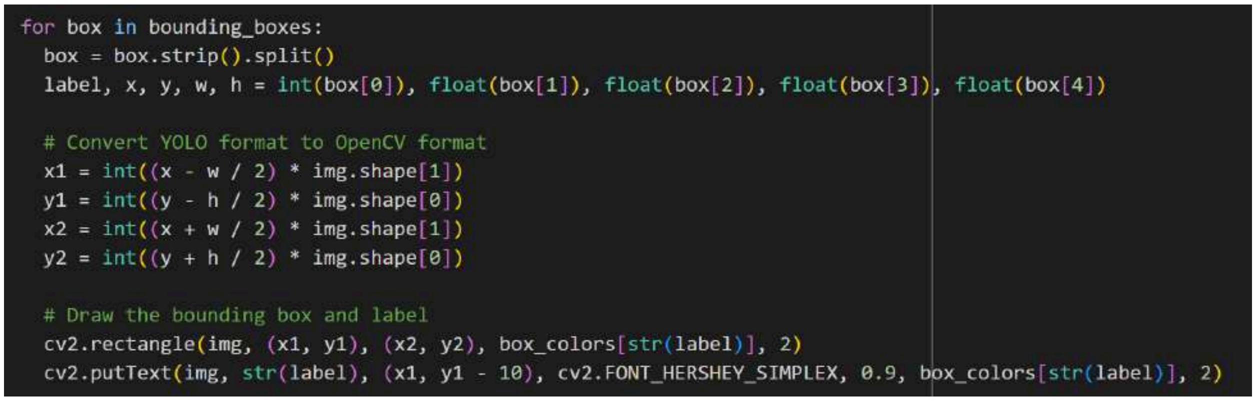
   Analyzing the frequency of each class in the training set

Class imbalance identified:

- Players(both teams): ~ 60,000 instances each
- Goalkeepers: ~ 3,500 instances each
- Ball: 7,798 instances
- Main Referee: 7,146 instances
- Side Referee: 5,361 instances
- Staff members: 2,211 instances

4. Bounding box analysis and image resolution analysis

   A. Scale Analysis



```python
for box in bounding_boxes:
  box = box.strip().split()
  label, x, y, w, h = int(box[0]), float(box[1]), float(box[2]), float(box[3]), float(box[4])

  # Convert YOLO format to OpenCV format
  x1 = int((x - w / 2) * img.shape[1])
  y1 = int((y - h / 2) * img.shape[0])
  x2 = int((x + w / 2) * img.shape[1])
  y2 = int((y + h / 2) * img.shape[0])

  # Draw the bounding box and label
  cv2.rectangle(img, (x1, y1), (x2, y2), box_colors[str(label)], 2)
  cv2.putText(img, str(label), (x1, y1 - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.9, box_colors[str(label)], 2)
```

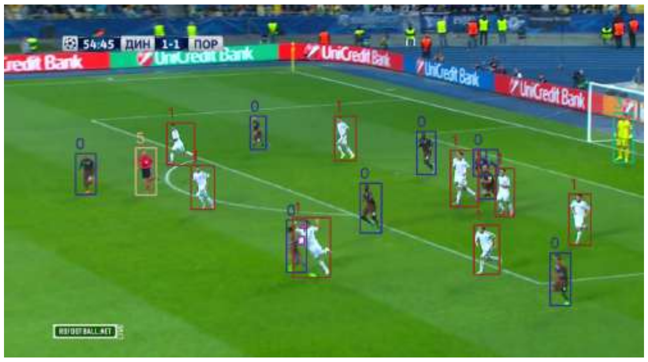


Fig 2.2.1 Output for the bounding box around detected objects

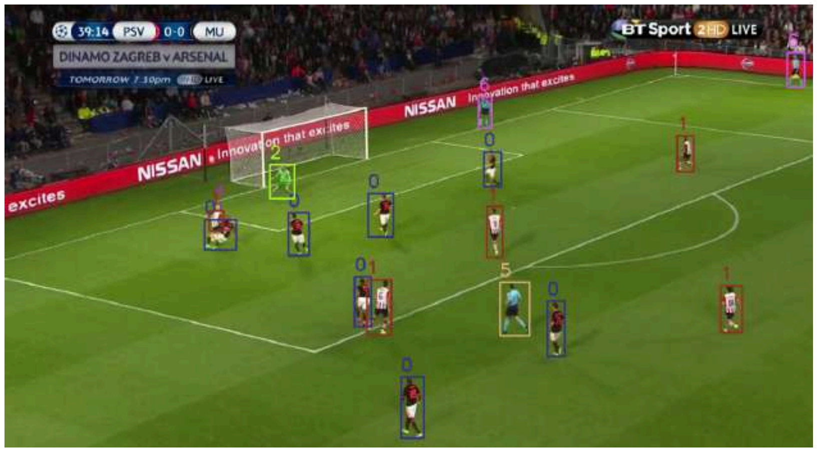


Fig 2.2.2 Output for the bounding box around detected objects

B. Image Resolution Analysis

- 1280x720: 6,877 images
- 1920x1080: 3,237 images
- 854x480: 51 images
- 1366x768: 31 images
- 704x576: 16 images

Stage 2 : Data Pre Processing

We performed necessary preprocessing steps to consolidate similar classes and update the dataset configuration.

1. Class Consolidation

   A. Implementing mapping strategy to consolidate classes

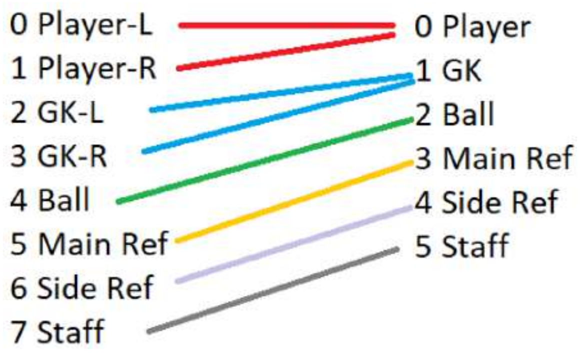


Fig 2.2.3 Reduction of classes

2. The consolidation reduced the number of classes from 8 to 6, making the following combinations

   B. Label File Updates

   Processing both training and validation sets to update the label files, this process

   - reads each label file

   - maps old class IDs to new ones

   - writes updated labels back to files

3. Configuration Updates

   A. The dataset configuration required updates to reflect the new class structure

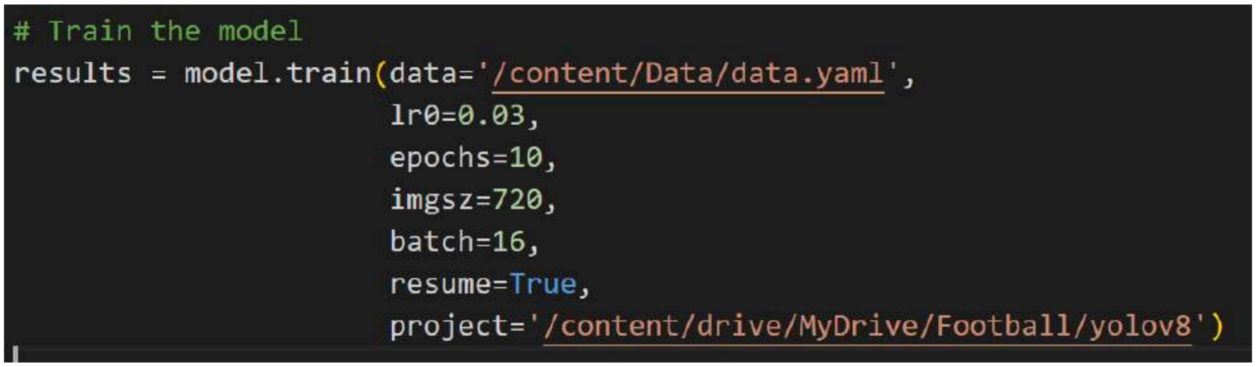
   B. Verification at each of the stages of updation

Stage 3 : Model Training

1. Model Selection and Training

   We chose to use YOLOv8m (medium) for this object detection task, utilizing transfer learning rather than training from scratch.

   A. Training Parameters



```
# Train the model
results = model.train(data='/content/Data/data.yaml',
                      lr0=0.03,
                      epochs=10,
                      imgsz=720,
                      batch=16,
                      resume=True,
                      project='/content/drive/MyDrive/Football/yolov8')
```

Fig 2.2.4 The training parameters for the model

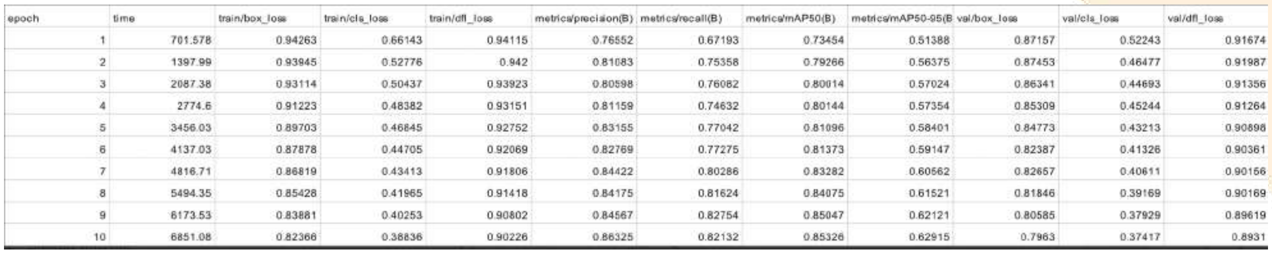
2. Model Architecture

   The YOLOv8m architecture consists of:

   - 295 layers

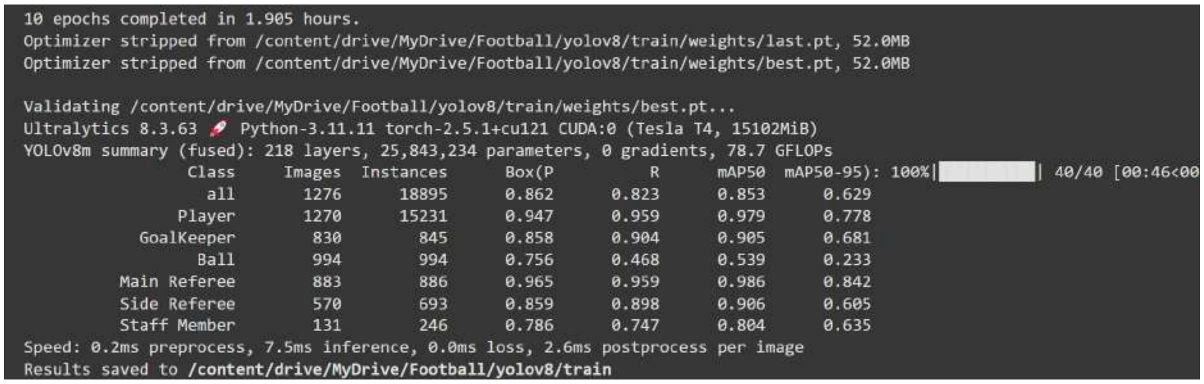
   - 25,902,640 parameters

   - 25,902,624 gradientes



epoch	time	train/box_loss	train/cls_loss	train/dfl_loss	metrics/precision(B)	metrics/recall(B)	metrics/mAP50(B)	metrics/mAP50-95(B	val/box_loss	val/cls_loss	val/dfl_loss
1	701.578	0.94263	0.66143	0.94115	0.76552	0.67193	0.73454	0.51388	0.87157	0.52243	0.91674
2	1397.99	0.93945	0.52776	0.942	0.81083	0.75358	0.79266	0.56375	0.87453	0.46477	0.91987
3	2087.38	0.93114	0.50437	0.93923	0.80598	0.76082	0.80014	0.57024	0.86341	0.44693	0.91356
4	2774.6	0.91223	0.48382	0.93151	0.81159	0.74632	0.80144	0.57354	0.85309	0.45244	0.91264
5	3456.03	0.89703	0.46845	0.92752	0.83155	0.77042	0.81096	0.58401	0.84773	0.43213	0.90898
6	4137.03	0.87878	0.44705	0.92069	0.82769	0.77275	0.81373	0.59147	0.82387	0.41326	0.90361
7	4816.71	0.86819	0.43413	0.91806	0.84422	0.80286	0.83282	0.60562	0.82657	0.40611	0.90156
8	5494.35	0.85428	0.41965	0.91418	0.84175	0.81624	0.84075	0.61521	0.81846	0.39169	0.90169
9	6173.53	0.83881	0.40253	0.90802	0.84567	0.82754	0.85047	0.62121	0.80585	0.37929	0.89619
10	6851.08	0.82366	0.38836	0.90226	0.86325	0.82132	0.85326	0.62915	0.7963	0.37417	0.8931

Fig 2.2.5 Epoch wise results



```
10 epochs completed in 1.905 hours.
Optimizer stripped from /content/drive/MyDrive/Football/yolov8/train/weights/last.pt, 52.0MB
Optimizer stripped from /content/drive/MyDrive/Football/yolov8/train/weights/best.pt, 52.0MB

Validating /content/drive/MyDrive/Football/yolov8/train/weights/best.pt...
Ultralytics 8.3.63 🚀 Python-3.11.11 torch-2.5.1+cu121 CUDA:0 (Tesla T4, 15102MiB)
YOLOv8m summary (fused): 218 layers, 25,843,234 parameters, 0 gradients, 78.7 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 40/40 [00:46<00
                   all       1276      18895      0.862      0.823      0.853      0.629
                Player       1270      15231      0.947      0.959      0.979      0.778
            GoalKeeper        830        845      0.858      0.904      0.905      0.681
                  Ball        994        994      0.756      0.468      0.539      0.233
          Main Referee        883        886      0.965      0.959      0.986      0.842
          Side Referee        570        693      0.859      0.898      0.906      0.605
          Staff Member        131        246      0.786      0.747      0.804      0.635
Speed: 0.2ms preprocess, 7.5ms inference, 0.0ms loss, 2.6ms postprocess per image
Results saved to /content/drive/MyDrive/Football/yolov8/train
```

Fig 2.2.6 Training Summary

3. Performance Analysis

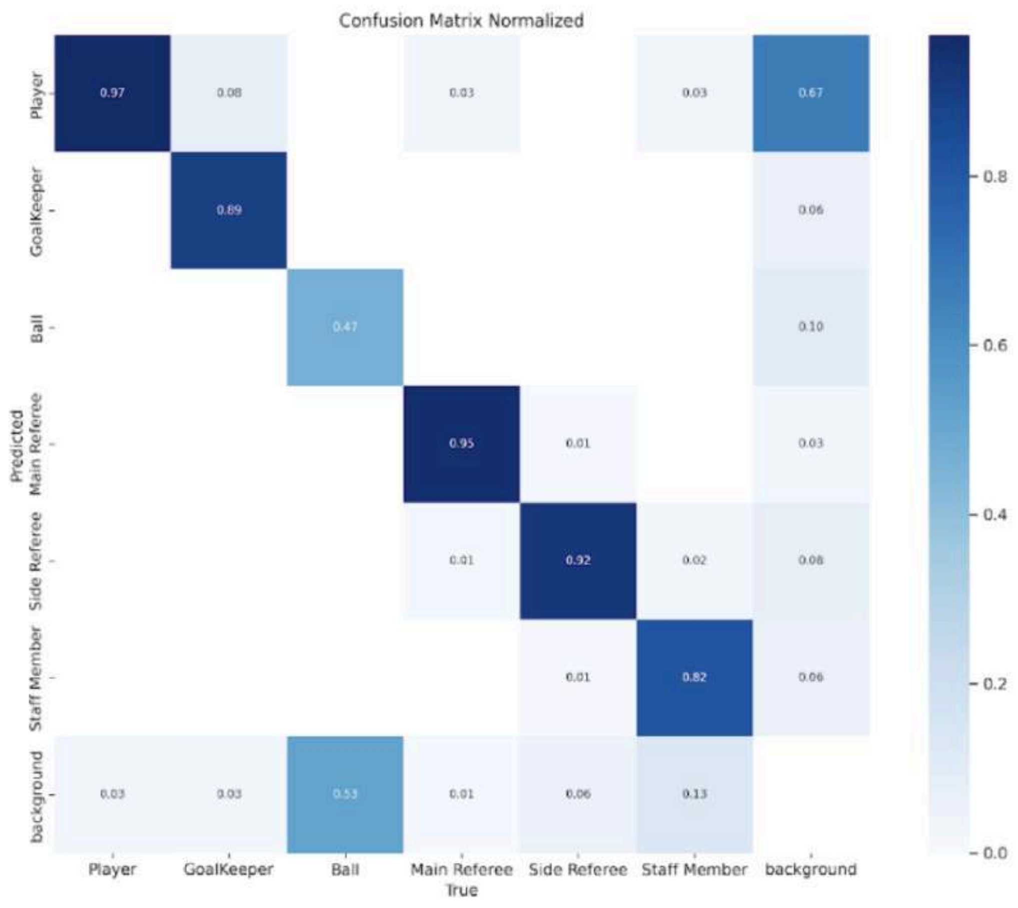


Fig 2.2.7 Model's confusion matrix on the validation set

A. Class Performance

- Most classes showed good detection performance
- The “Ball” class demonstrated lower performance

Stage 4 : Model Inference

We loaded the trained Yolov8 model using the best weights from trainings

A. Label Configuration

Defining the class labels and creating a random colour scheme for visualization

B. Visual Implementation

Implementing a visualization pipeline that draws bounding boxes and labels on detected objects

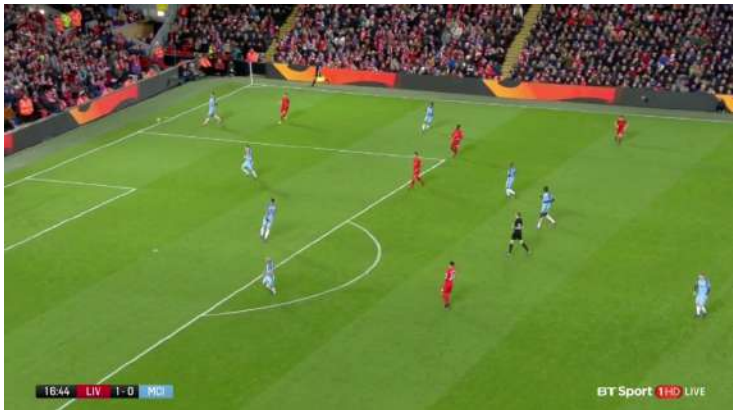


Fig 2.2.8 The input to label configuration

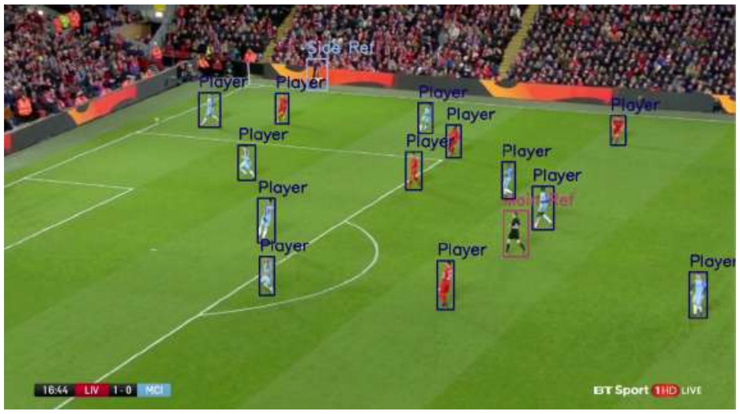
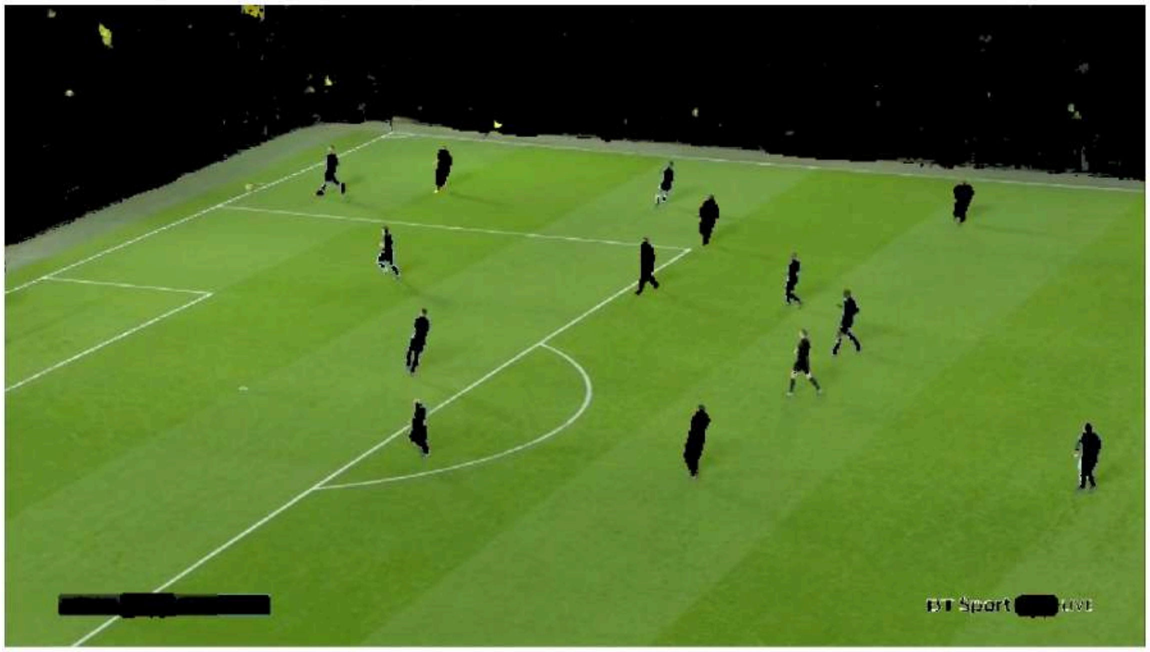


Fig 2.2.9 The output to label configuration

Stage 5 : Post Processing

The post-processing stage aims to classify detected players back into their respective teams using kit color analysis, the process mimics human visual perception by analyzing player uniforms and their positions on the field

1. Grass Color Detection



```
Grass BGR color:  (53.95960946650855, 147.50698347798883, 106.90536983550741)
```

Fig 2.2.10 Grass Colour Detection Output

- Provides baseline for each player extraction

2. Player Kit Color Extraction

The next step is to extract each player from the image and remove the grass from the background of each player by masking out the grass color (using the average color that we found in the previous step) and removing the bottom half of the image to remove any influence from the shorts color. Finally, we get the color of the kit by averaging out the remaining pixels

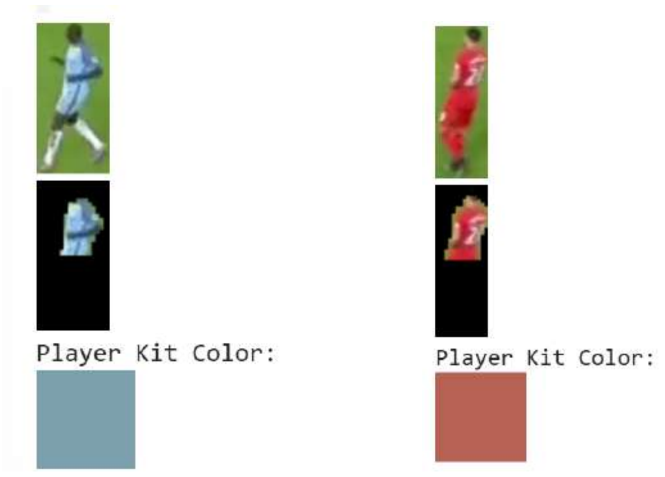


Fig 2.2.11 The Output for Kit Colour Extraction and ignoring the lower half to ignore various colours for shorts

3. Team Classification using K-means Clustering
   The next step is to cluster the kits colors into 2 groups, each containing the kits that have a similar color to each other, and each group will represent one team. This step can be performed by using a K-Means clustering algorithm on the kit colors list that we

found in the previous step.

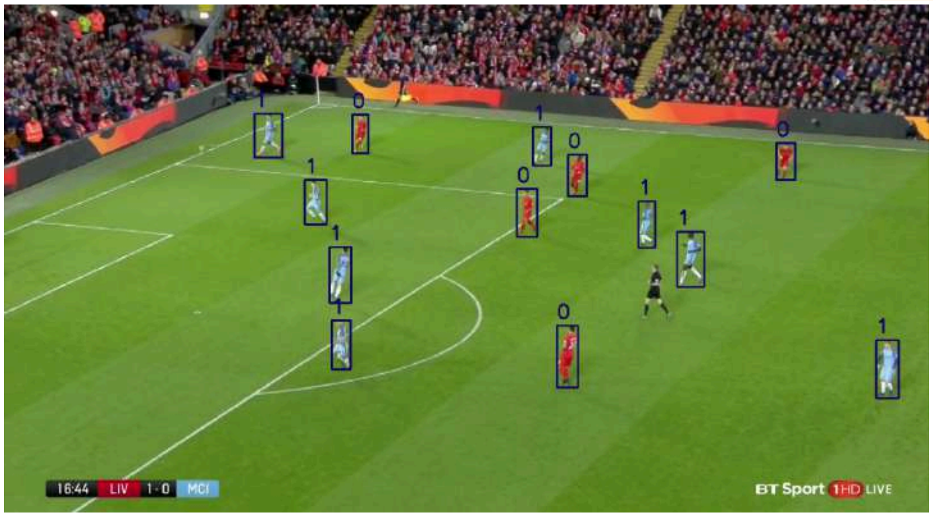


Fig 2.2.12 Output for Team Classification using K-Means Clustering

4. Left Right Team Assignment

   The next step is to find out which team is "Team Left" and which one is "Team Right", this is performed by finding out the average bounding box x-axis position of each team and then labeling the team with the lower x value as "Team Left" and labeling the other team as "Team Right"

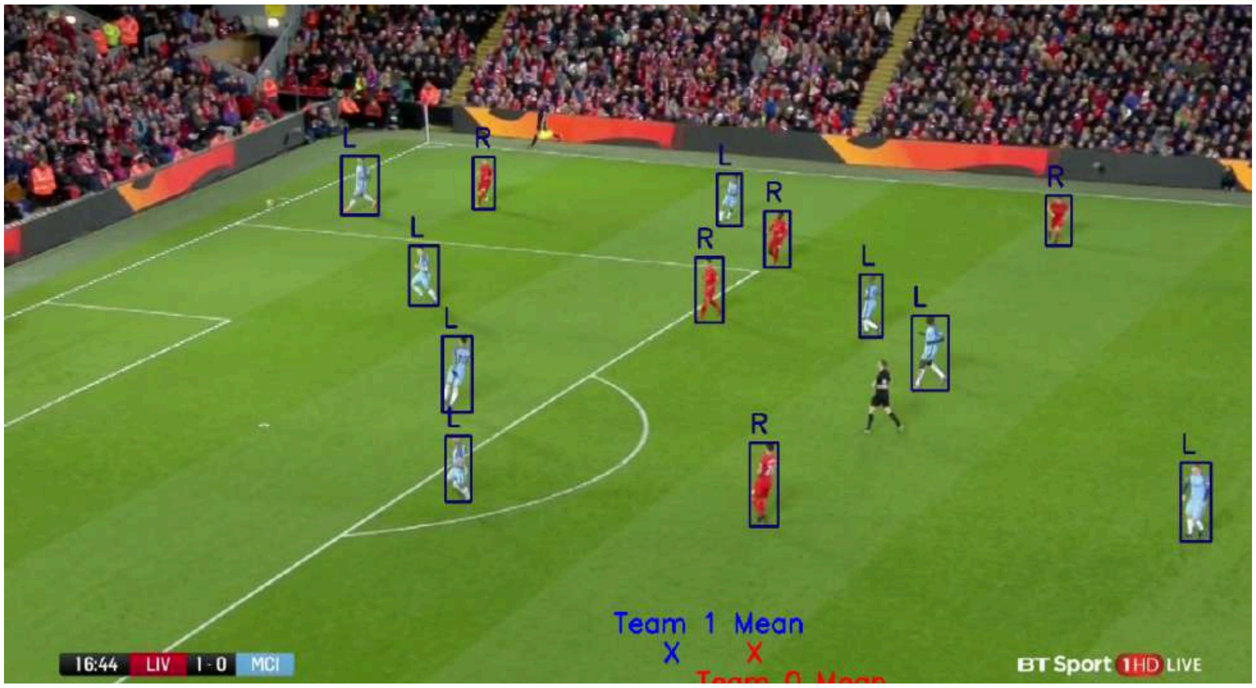


Fig 2.2.13 Output for Assignment of the Left and the Right team

GitHub : Repository Private (Access Available on Request)

Google Drive Link : Football

Colab Notebook Link : Soccor OCR Implementation

Input:

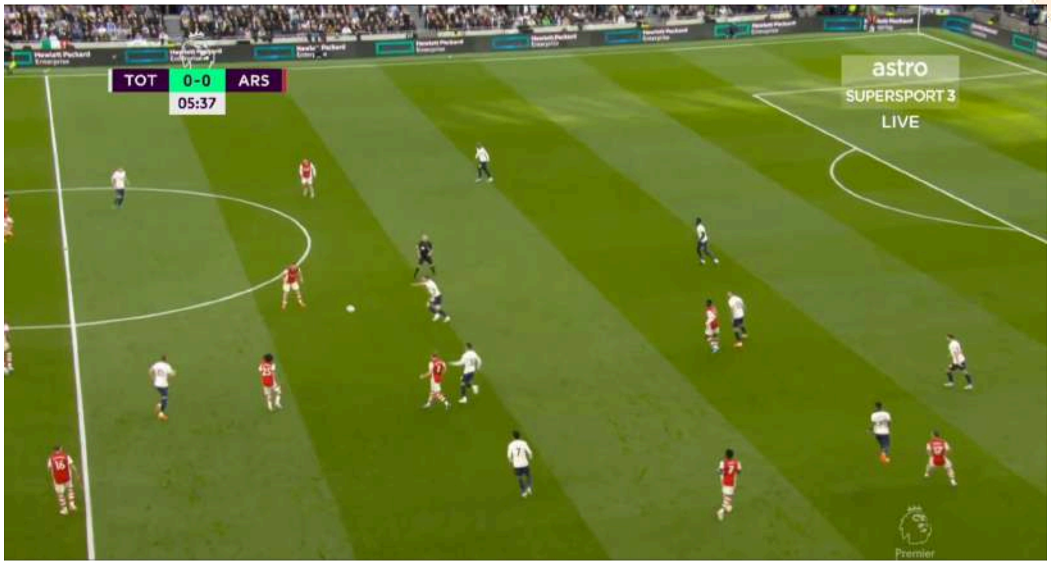


Fig 2.2.14 Snapshot from input video 1

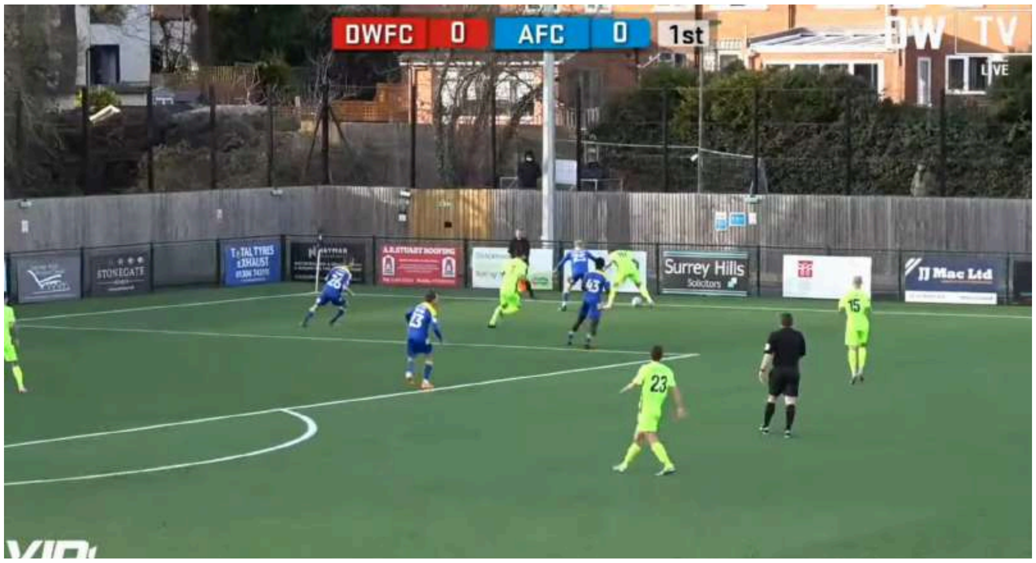


Fig 2.2.15 Snapshot from input video 2

Outputs :

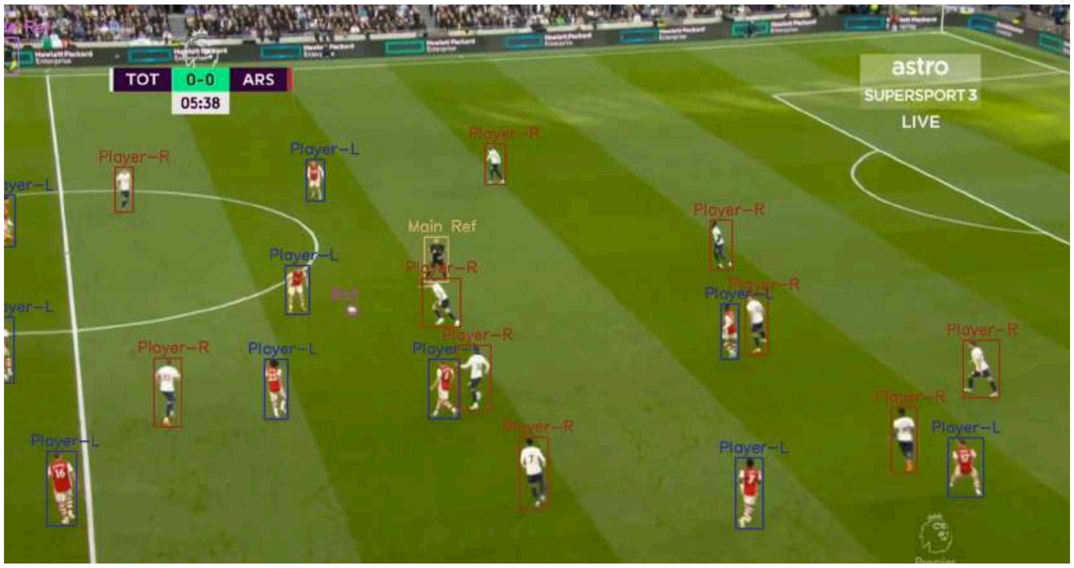


Fig 2.2.16 Snapshot from output video for the input video 1

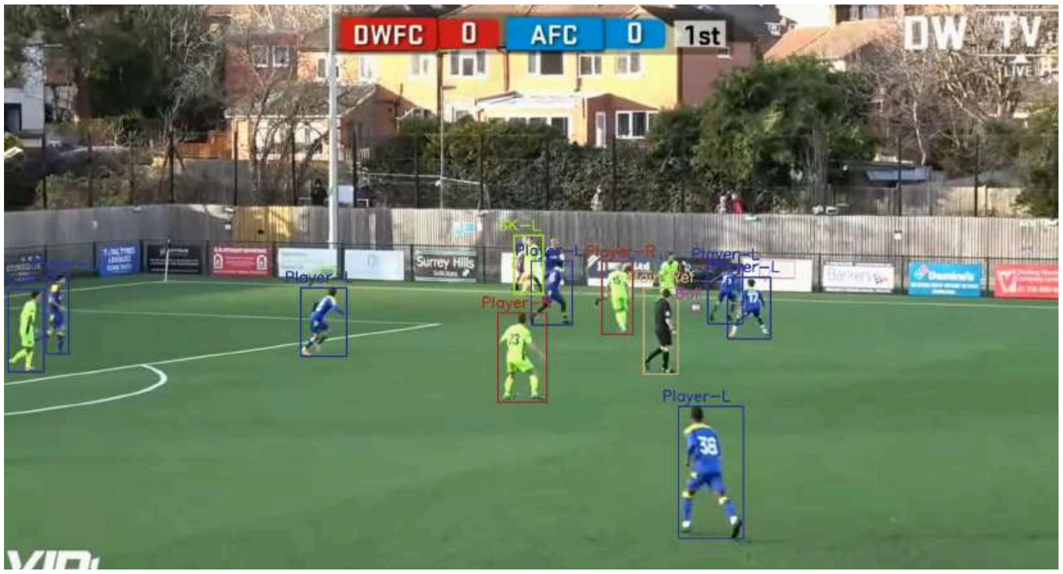


Fig 2.2.17 Snapshot of output video for input video 2

Section 3: Challenges and Future Work

Challenges

- Identifying Individual Players:
  From a bird's-eye view, players' names and jersey numbers are not visible, making it difficult to identify specific players. This limits the ability to provide player-specific insights or maintain consistent tracking throughout the game.

- Dynamic Camera Angles:
  The camera frequently changes its angle and position based on the ball's location and the team in control. This creates challenges in maintaining consistent player tracking IDs as players move rapidly across the field or temporarily exit the frame.

- Rapid Movements:
  Soccer involves high-speed player movements, leading to abrupt positional changes. Tracking systems must handle these rapid transitions while maintaining accuracy, which can be difficult with limited data or suboptimal models.

- Untrained Perspectives:
  Different camera perspectives, such as close-ups, side views, or high angles, can expose models to scenarios not encountered during training. This can degrade performance, especially in detecting or tracking players and the ball.

- Wide Variety of Training Data:
  To overcome the variability in camera angles and perspectives, a large and diverse dataset covering various scenarios, lighting conditions, and angles is required. Ensuring this variety demands significant time and effort in data collection, annotation, and model fine-tuning.

Future Work:

In the future work, Fine-tuning of the YOLOv8 model on the local system will be implemented, followed by migration to YOLOv11 to achieve improved accuracy and speed will be prioritized.

Further, Key event detection and frame-based score tagging will be explored to enable timestamping for faster retrieval and concise summaries by leveraging object and spatial detection models.

Also Language models and NLP techniques will be utilized to analyze audio commentary, identifying patterns such as key phrases and tone changes, which will be synchronized with video tagging to enrich metadata will be explored. Furthermore, the refined and trained models will be leveraged to implement an optimized pipeline for enhanced performance and efficiency.

Internship Confirmation Letter

Dear _____ ,

We are happy to inform you that you have been selected for an internship with **ITTIAM Systems Private Limited**. We're excited to have you join our team and look forward to a productive and enriching experience for both you and the organization.

Internship Details -

Position	: Research Intern
Duration	: **9th June 2025** to ___________
Stipend	: ___________

During your internship, we encourage you to contribute actively and make the most of the learning opportunities available. Please maintain professionalism at all times and respect the confidentiality of any company-related information you may come across.

At the end of your internship, you will be expected to compile a brief project report summarizing your insights and learnings, along with any suggestions or recommendations you may have.

While your time here will be collaborative and open, we do expect you to maintain confidentiality on company-related matters. Additionally, any work, ideas, or materials created during your internship will remain the property of **ITTIAM Systems Private Limited.**

Welcome aboard!

Prof. Prasad B Honnavalli
Director C-ISFCR & C-IoT
Professor CSE Dept.