# ASSIGNMENT

# PLAYER RE-IDENTIFICATION IN SPORT FOOTAGE

PROVIDED BY :

**Company a Name: Liat.ai**
**In GitHub Repository**

SUBMITTED BY :

**SHIVAM**
**B.TECH (ECE) - 2025**
**Delhi Technological University**

- **VS CODE FILE LINK :**
  https://drive.google.com/file/d/1H4ZgaIhScs2Dbqwb_cOBpbR_694vLbdP/view?usp=drive_link
- **TRACKED OUTPUT VIDEO:**

  https://drive.google.com/file/d/16FyX31-0At0gjzlFjUScWRJm_R7eThDq/view?usp=sharing

- **ORIGINAL VIDEO:**
  https://drive.google.com/file/d/1DxNwPNYM0A4Iskif7yDsOSPY03NSix1Y/view?usp=sharing

- **Custom YOLOv8 model:**

- https://drive.google.com/file/d/1RaHJr0X_1fD6A7Nbn2McbCu5V2B-p83j/view?usp=sharing

## Introduction

The goal of this project was to build a working player re-identification and tracking system for a football match video using computer vision. The idea was to automatically detect all players on the field, track them across frames consistently, and clearly label them with unique IDs.

This was an exciting and challenging assignment as it mimicked real-world problems — dealing with motion, occlusion, similar jerseys, changing angles, and ball/player confusion. I handled everything from dataset/model setup to detection, tracking, ID remapping, and visualization.

## 2. Project Setup and How to Run

### Steps to Run:

1. Clone this repository or download the folder.
2. Make sure you have Python 3.8 or higher installed.
3. Install the required dependencies using:
4. Place your input football video (e.g., `15sec_input_720p.mp4`) inside the project folder.
5. Run the script with:
6. The output video with tracked players will be saved as tracked_output.mp4.

## Dependencies:

- Python ≥ 3.8
- OpenCV
- Ultralytics YOLOv8
- Deep SORT Realtime

Installed all dependencies via:

**pip install ultralytics opencv-python deep_sort_realtime**

## 3. My Approach and Methodology

I broke the assignment into the following components:

### Detection

I used a pretrained custom YOLOv8 model (best.pt) to detect objects in each frame. The model is capable of recognizing:

- Players

- Ball
- Referee

I filtered the detections to track only **players** (based on their class name).

➢ **2. Tracking**

To track players frame-by-frame, I used the **Deep SORT** tracking algorithm, which keeps identities consistent using a combination of bounding box position and appearance.

Initially, Deep SORT was assigning huge, inconsistent track IDs (e.g., 131, 237, 354). This made the output confusing, especially when re-identifying players. I solved this by introducing a **mapping layer**:

- Internally store Deep SORT's raw ID
- Assign new consistent IDs like Player 1, Player 2, etc.

➢ **3. Ball Detection**

The model also detects the ball, but I deliberately excluded the ball from tracking. However, I labeled it clearly in the visualization to show the system's capability.

➢ **4. Visualization**

Each player is enclosed in a green box with a clear label (Player 1, Player 2, etc.). Detections and their class names are also optionally shown in light blue for debugging.

## 4. Techniques I Tried

| Technique | Purpose | Outcome |
|---|---|---|
| YOLOv8 Custom Training | Foraccurate detection football frames | Performed well on |
| Deep SORT | For identity tracking | Successfully maintained consistent IDs |
| ID Remapping | To make results interpretable | Worked perfectly (Player 1, 2, 3...) |
| Label Filtering | Only tracked 'player' class | Prevented ball/referee from being misidentified |

## 5. Challenges Faced

- **Inconsistent DeepSORT Track IDs**: At first, DeepSORT gave IDs like 154, 213, 83 even for the same 11 players. This was confusing.

    - **Fix**: I implemented ID remapping using a dictionary (id_map) and counter (next_id).
- **Ball detected as Player**: Since the model labeled the ball as class_id = 0 sometimes, I added clear filtering logic to only track players.
- **Fast Player Movement**: Sudden motion changes led to dropped tracks.
    - **Fix**: I tweaked max_age=20 and n_init=3 in DeepSORT config to handle brief occlusions.
- **Model Class Labels**: My YOLOv8 model had different label mappings. I had to inspect and adjust PLAYER_LABELS, BALL_LABELS, etc.

### 🚀 What Remains / Improvements

If I had more time and compute resources, I would:

- Add **jersey color classification** to group players by team (red vs blue)
- Refine ball tracking logic to assign possession to the nearest player
- Make the code modular with separate tracker.py, utils.py, etc.
- Run this on a **longer video (full match)** and test across different angles

### 🚀 Sample Output Frame

Here's an example from the output (attached in report):

- Player 1, Player 2, etc. are correctly tracked
- Ball is separately labeled
- All detections are visually clear and accurate

### ✅ Final Words

This project gave me hands-on exposure to real-time multi-object tracking, re-ID challenges, and deploying a full pipeline. I truly enjoyed building this end-to-end. I faced real bugs and fixed them through debugging, learning from official docs, and fine-tuning configurations.

This solution is entirely my work — from model setup to writing custom remapping logic and visualization. The result is a reproducible, working assignment that can track football players accurately using modern AI tools.