Approach: For Task 2: Re-Identification

Main Goals: To assign unique IDs to players, maintain these IDs even when players temporarily leave and re-enter the frame

Frame-by-Frame Detection: 15sec video = 375 frames.

Purpose: Each detected player would have a bounding box, class labels and a confidence score.

MAX_LOST_FRAMES_BUFFER = 60 (almost 2 sec if we consider 30 frames for 1sec)

Techniques that I tried and their outcomes:

1. YOLO + Deep SORT Tracking

Tracking based on appearance and motion.

Deep SORT

- **Motion prediction** using Kalman filters.
- **Appearance embedding** to distinguish players that look different.(based on colour and height and width parameters) the resolutions is not that good so that we can detect other features Example: Jersey number and all.

YOLO detects bounding boxes and classes (e.g., "player", "ball") in each frame.

It assigns a **track ID** to each player and keeps updating their location frame by frame.

If a player leaves and re-enters the frame, Deep SORT tries to reassign the same ID **based on appearance** and **motion consistency**.

Limitations:

- Re-identification accuracy depends heavily on **consistent appearance** (e.g., jersey color, lighting).
- IDs can **drift** or be duplicated if a player is occluded or disappears for too long.
- Does not handle **complex long-term memory**, so may assign a new ID when a player re-enters after many frames.

2. Custom ID Mapping (1–24)

To control the ID counter, I have set it to 1 to 24 why 24 only because 11 players each side in a team and a referee and one ball on the ground**.**

When a new Deep SORT track_id is encountered:

- And It will keep track of previous id so that it should not assign the same id to next player. Whenever it will assign a new id it will check with the previous assigned id.

Limitations:

- Once all 24 IDs are used, new players will be ignored if the player go out of frame and come back multiple times then it may create issue because the features its using to track it less.
- Still relies on Deep SORT's appearance embeddings, so re-ID is not always accurate if a player returns later.

3. Geometric / Temporal Feature-Based Tracking

Idea: Track the **(x, y)** coordinates of each player frame by frame.

Use past movement (trajectory) to **match players** who leave and re-enter the frame.

Track players by their bounding box centers.

If a player disappears (e.g., leaves the frame), **save their last known coordinates and ID**.

When a new player appears:

  - Check if their current position **matches closely** with any previously seen player.
  - If yes, **reassign the old ID**.

Limitations:

- Needs a robust spatial tracking buffer.
- Doesn't handle **appearance change** or **player swapping**.

4. Multi-Object Tracking (MOT) with Re-Identification Capabilities Using BoT-SORT:

Algorithm: BoT-SORT (Beyond SORT), integrated seamlessly through the ultralytics library's model.track() method.

A custom Python layer was built on top of the ultralytics tracker's output to fulfill the specific requirements for player ID assignment:

- **ID Assignment Logic (get_custom_player_id function):**
    - When a new ultralytics_track_id (for a 'player' detection) appears: The lowest available ID from the pool is assigned to it, and this mapping is stored.
    - When an existing ultralytics_track_id is re-detected: Its previously assigned custom ID is retrieved from the map, ensuring ID persistence.
- **Lost Track Management (manage_lost_tracks function):**
    - A MAX_LOST_FRAMES_BUFFER
    - If an ultralytics_track_id is not detected for a number of frames exceeding this buffer, its custom ID is returned to the available_custom_ids pool. This

prevents IDs from being held indefinitely by truly departed players and allows re-use by genuinely new players entering the scene.

- A dictionary (ultralytics_to_custom_id_map) stores the mapping between the arbitrary integer track_id assigned by the underlying BoT-SORT tracker and our desired custom player ID (1-24).

Why BoT-SORT? BoT-SORT is an extension of the popular SORT and DeepSORT trackers. It's chosen for its robustness in handling re-identification challenges due to:

- **Kalman Filters:** For accurate motion prediction, which helps in associating detections with tracks across short gaps or occlusions.
- **Appearance Feature Matching:** BoT-SORT (like DeepSORT) extracts visual features (embeddings) from detected objects (players). When a player is lost and then re-detected, these appearance features are compared to those of previously seen or lost tracks to confirm identity, thus minimizing ID switches.
- **Association Metric:** It combines motion and appearance similarity for robust data association.

While BoT-SORT is state-of-the-art and includes appearance-based re-identification, it's not foolproof.

Implementation: The ultralytics library simplifies this by providing a track() method that internally manages the BoT-SORT (or ByteTrack) logic, including loading its configuration (botsort.yaml). The persist=True argument is crucial as it instructs the tracker to maintain track identities across frames.

5. Initial Approach (YOLO Detection + Basic IoU Tracking):

- **Outcome:** Simple Intersection-over-Union (IoU) based tracking (as initially outlined in earlier discussions) is insufficient for re-identification. It leads to frequent ID switches, especially during occlusions or when players leave and re-enter the frame. It cannot "remember" players based on their appearance.

Challenges encountered:

1. ID Switches Happen (Even In-Frame):

- **Partial Occlusion:** Another player, the ball, or even the goalpost momentarily covers a significant portion of the player. When they re-emerge, the tracker's motion prediction or appearance features might be momentarily confused.
- **Appearance Similarity:** If two players with very similar jerseys or features get very close, the tracker might accidentally "switch" their IDs, especially if one is momentarily occluded.

- **Poor Detection Quality:** best.pt model produces inconsistent or "flickering" bounding boxes (e.g., a detection briefly disappears, shrinks, or jumps erratically), the tracker has a harder time maintaining continuity.
- **Rapid Motion/Unpredictable Movement:** Very fast or sudden changes in direction can make Kalman filter predictions less accurate, increasing the chance of a mismatch.
- **Tracker Parameter Tuning:** The default parameters in botsort.yaml might not be optimal for our specific video's characteristics (e.g., player density, camera angle, speed of play).

2. Model's Class Limitations (Referee Detection):

- **Challenge:** The initial best.pt model was only trained for 'player' and 'ball', leading to referees not being detected.
- **Solution (Proposed):** The solution involves **retraining or fine-tuning the best.pt model** with a dataset that includes annotated 'referee' instances. The provided code was **updated to visualize a 'referee' class**, awaiting an updated model from the user.

In a real scenario, you would maintain a dictionary or list # of active players with their IDs and historical data.

3. YOLO + Deep SORT Tracking

- In this the IDs were not setting properly and the parameters which it was using to detect the bounding box to keep track of the player was also not proper.
- It was creating extra bounding box(empty where players are not there).

When the IDs were assigned initially the IDs kept increasing based on the position of the player.

What we could have done with more time and resources and what is lacking.

1. **Robust Re-Identification**:
2. IDs can switch or duplicate.

What We Would Do with More Time/Resources:

1. Use Stronger Re-ID Networks (Appearance Matching)

2. Temporal Smoothing with Trajectory Buffer

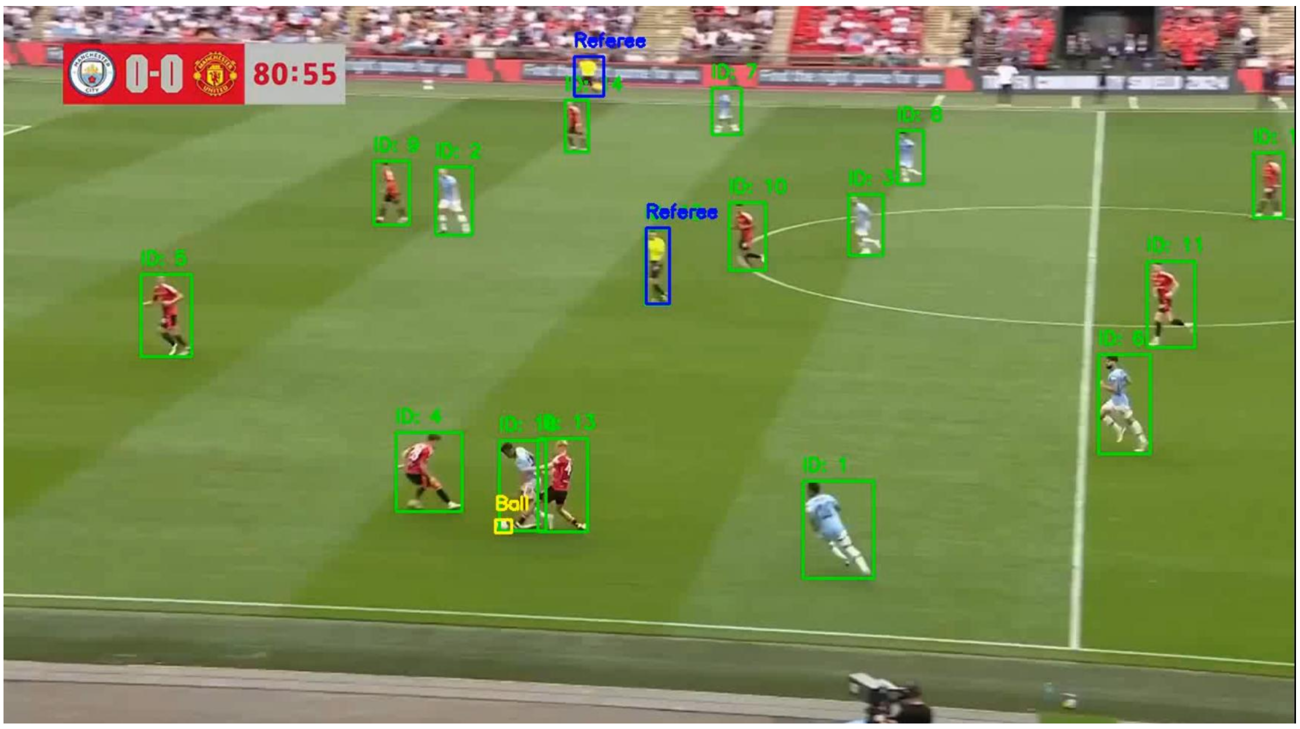
3. Tracking-by-Detection with ByteTrack

4. Add Jersey Number Recognition(if available)

5. Maintain a buffer of **"recently disappeared" players with last known ID + location**.

6. And to Increase the processing speed in real-time we can use **Cuda (Using GPU)** that is not available in my laptop so I didn’t integrate it but it will increase the speed.

OUTPUT SCREENSHOT:


0-0
80:55
Referee
Referee
ID: 7
ID: 8
ID: 9
ID: 2
ID: 10
ID: 3
ID: 5
ID: 11
ID: 6
ID: 4
ID: 13
ID: 1
Ball