**Player Re-identification in One Feed**

**(Report)**

****1) My final approach****

**Objective: Detect, assign unique IDs, and track players in video feeds.**

* **Key Libraries**: Uses cv2 (OpenCV) for video/image processing, numpy for numerical operations, csv for data logging, and ultralytics (YOLO) for object detection.
* **System Architecture**: Sequential pipeline involving video input, frame sampling, YOLO-based object detection, feature extraction, custom player tracking/re-identification, data logging, visual output, video output, and track management.
* **Object Detection**: Employs a YOLO model (best.pt) to identify player bounding boxes in each frame.
* **Tracking Methodology**: A custom algorithm maintains a track\_db (tracking database) for active players.
  + **Features Used**: Centroid coordinates (spatial location), dominant jersey color (hue), and bounding box height.
  + **Matching Logic**: New detections are matched to existing tracks based on spatial proximity (min\_dist), color similarity (color\_thresh), and height similarity (height\_thresh). If no match, a new ID is assigned.
  + **Track Management**: Disappeared players are removed from track\_db after a set period (e.g., 2 seconds) to handle temporary occlusions.
* **Outputs**: Generates an annotated video (player\_tracking\_output.avi) with visual tracking (circles and IDs) and a CSV file (player\_tracking\_output.csv) containing frame-wise tracking data (frame, ID, cx, cy, height, jersey\_color).

Manus has completed the current task

**Techniques I tried and their outcomes:**

**1)** In this technique :

* **YOLOv8 Detection** – Detects players (class 2) in each frame.
* **IoU(Intersection over Union) Filtering** – Removes overlapping detections using IoU > 0.7.
* **DeepSORT Tracker (Standard Built-In Algorithm)** – Tracks players using Kalman filter and Re-ID features.
* **Smooth Bounding Boxes** – Uses Kalman-predicted boxes for stable tracking.
* **CSV Logging** – Saves tracked player IDs and positions frame-by-frame.

**Output:**

**Pros:**

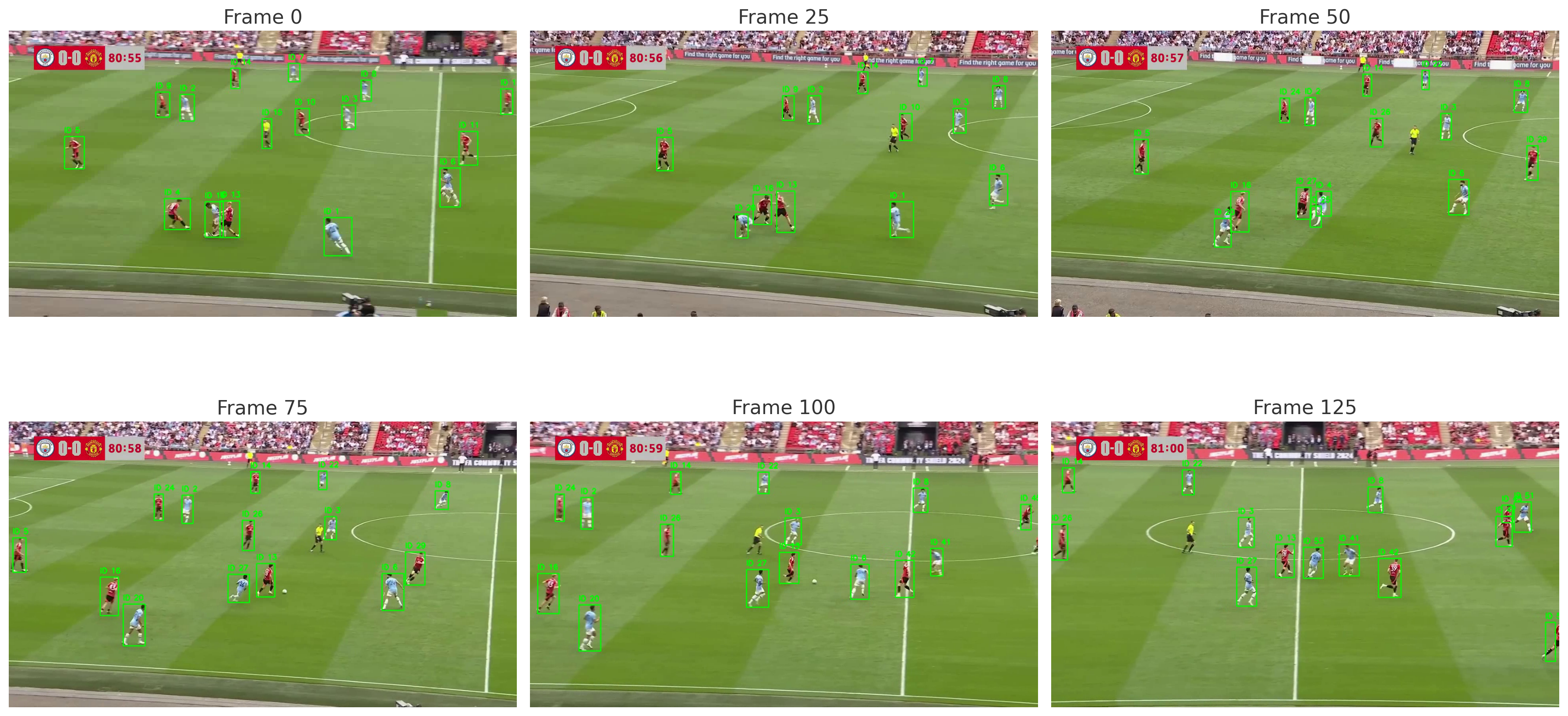
**Accurate Detection, Smooth Tracking, ID Persistence, Simple Integration, CSV Logging**

### **Cons:**

**ID Switching, Over-filtering**

****2)** In this technique :**

* **YOLOv8 Detection – Detects players (class 2) with confidence > 0.7.**
* **ByteTrack Tracker** – Tracks players using IoU-based matching (no Re-ID).
* **Built-in Tracking Pipeline** – Uses model.track() with persist=True.
* **Custom Tracker Config** – Tweaked using bytetrack\_custom.yaml for better performance.
* **CSV Logging** – Logs player positions once per second.

**Output:** **25 frames per second && below data taken at the 1 sec interval:**

**Pros:**

* **Fast & lightweight – Runs in real-time using YOLOv8 + ByteTrack.**
* **No external tracker needed** – Uses built-in model.track() API.
* **Good ID consistency** – Works well when players are spaced apart.
* **Customizable tracking** – Configurable via .yaml file.
* **Simple logging** – Easy to log data once per second.

**Cons:**

* **Frequent ID switching – Happens when players are close or occluded.**
* **No appearance-based Re-ID** – Tracker can’t remember players visually.

**3) In this technique:**

* **YOLOv8 Detection** – Detects only players (class 2) with confidence > 0.7.
* **BoT-SORT Tracker(standard built in tracking algorithm)** – Uses improved tracking with motion + appearance features.
* **Custom ID Mapping** – Replaces track IDs with your own consistent Player\_IDs.
* **Disappearance/Reappearance Logic** – Tracks status (disappeared / reappeared) manually.
* **Detailed CSV Logging** – Saves ID, position, and player status once per second.

**Output :**

****pros:****

* Player Re-identification: System re-identifies players across frames.

• Precise Bounding Boxes: Provides accurate X, Y, Width, Height data.

• Granular Frame Data: Detailed info available for each frame.

• Multi-Player Tracking: Tracks multiple players simultaneously.

**Cons:**

* Short Tracking Durations: Many players tracked for only a few frames.

• Frequent Appearance/Disappearance: High player ID turnover suggests tracking issues.

• Limited Status Info: Only 'reappeared' status, lacks detail on tracking state.

• No Tracking Loss Info: Doesn't show when players are lost or gaps occur.

• Potential ID Switches: Reappeared' status might not always be correct re-identification.

4) Techniques Used:

• YOLOv8: Player detection.

• Ultralytics Tracking: Built-in multi-object tracking.

• Kalman Filter: Bounding box smoothing.

• Rule-Based Filters: Speed, field bounds, and basic color-based team classification.

### Pros:

• Accurate Detection: Good at finding players.

• Smooth Movement: Kalman filter helps with fluid tracks.

* Team ID: Basic team classification.

•Clear Visuals: Bounding boxes and IDs are well-displayed.

### Cons:

•Redundant Smoothing: Kalman filter might be unnecessary with built-in tracker.

•Basic Team ID: Color-based classification can be unreliable.

•Hardcoded Settings: Requires manual tuning for different videos.

•Tracking Gaps:

5) Techniques Used:

1. YOLO (You Only Look Once): For fast object detection (identifying players).

2. DeepSORT: For robust object tracking (maintaining player IDs).

## **Key Pros (Combined System):**

1. Real-time Performance: Fast detection and tracking suitable for live video.

2. Accurate Tracking: Maintains consistent player identities even with occlusions.

3. Versatility: Applicable to various video analysis tasks.

## Key Cons (Combined System):

1. Cumulative Errors: Detector errors can negatively impact tracking.

2. Computational Demands: Requires powerful hardware for optimal performance.

6) **Techniques Used**

1. **YOLOv8 for Detection Only** – Tracking is not handled by a tracker but manually using centroids.
2. **Centroid-Based Matching** – IDs assigned based on Euclidean distance between current and previous detections.
3. **Custom ID Management** – Each new unmatched detection gets a new unique Player ID.
4. **Re-ID Handling via Last Seen Frames** – Players are removed if not seen for 2 seconds (based on fps).
5. **Random Color per ID** – Each player gets a unique random color for better visualization.

### ****Pros****

1. **Stable IDs** in moderately crowded scenes with consistent player movement.
2. **Simple to implement** and customize (no deep feature extractor or third-party tracker required).
3. **Good control** over ID assignment and removal via centroid and frame-based logic.
4. **Lightweight solution** – works well on CPU with fewer dependencies.
5. **Scene change handling** – skips frames with >30 detections to avoid audience or noise.

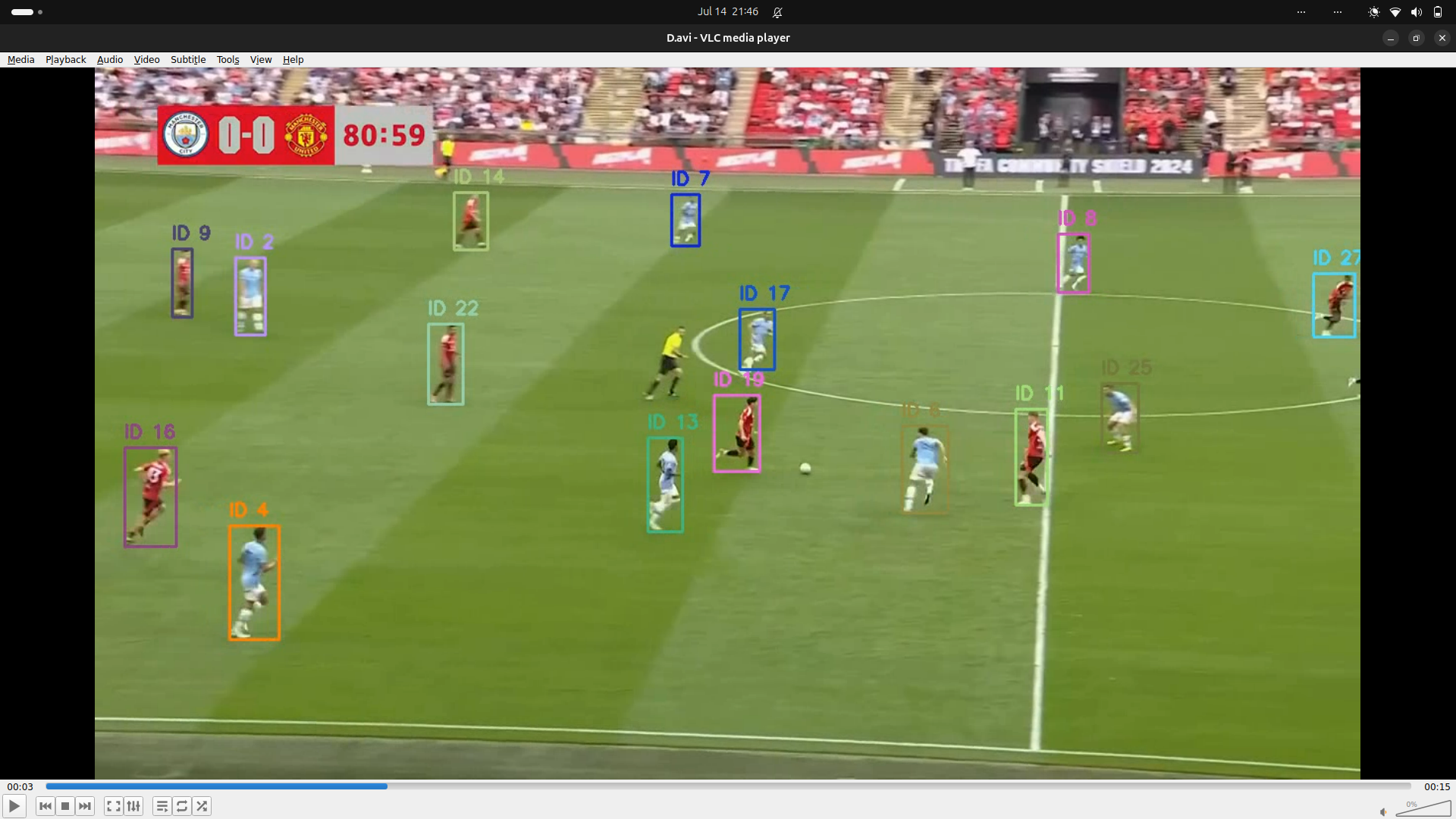
### ****Cons****

1. **Fails when players come close** – ID switching or duplication occurs in congested areas.
2. **No appearance matching** – color or jersey pattern isn’t considered, only location.
3. **New ID every time player reappears after 2s** – poor long-term ID consistency.
4. **Not robust to occlusion or fast motion** – IDs break if players suddenly change direction.
5. **Manual tuning required** – threshold distance, disappearance time, and frame-skipping need careful tuning.

7)

### Technique Used:

I am using a **custom YOLOv8 model for player detection**, combined with a **basic centroid-based tracking** algorithm enhanced by **IoU (Intersection over Union)** and **Euclidean distance** for player re-identification. Each player is given a unique Player\_ID, and tracking is maintained frame-by-frame.



### Pros:

* **Efficient Detection:** YOLOv8 offers fast and accurate player detection.
* **Custom Model:** Tailored to your dataset (likely trained for football players), ensuring better performance than generic models.
* **Simple Yet Effective Tracking:** Using both **centroid distance** and **IoU** balances position and shape consistency.
* **Unique ID Assignment:** Players retain their IDs well, making analysis and logging easier.
* **CSV Logging:** Easy to use for downstream tasks like performance analysis or heatmaps.
* **Visual Feedback:** Real-time bounding boxes and IDs aid in validation.

### Cons:

* **Basic Tracker:** Lacks robustness in occlusion, abrupt motion, or overlapping players.
* **No Re-ID Module:** If a player disappears and reappears, they might get a new ID.
* **Limited Temporal Consistency:** No use of Kalman filter or motion prediction.
* **Crowd Sensitivity:** Skips frame if too many detections (e.g., audience), which may miss real frames.
* **Static Thresholds:** Fixed dist < 60 and iou > 0.1 may not generalize to all scenes.
* **No Team Classification:** All players are treated equally—no team/role separation yet.

8) **Techniques Used**

* **Centroid Distance Matching** – track players by proximity
* **HSV Dominant Color** – detect jersey color for team ID
* **Bounding Box Height Comparison** – ensure scale consistency
* **Manual ID Assignment** – assign new ID if no match
* **Frame Skipping** – control processing load for higher FPS
* **Basic Visualization** – draw bounding boxes and ID labels.



### ****Pros****

* Simple and easy to implement
* Lightweight (no heavy models)
* Uses color for team differentiation
* Customizable tracking logic
* Can run in real-time on CPU

### ****Cons****

* No motion prediction (no Kalman filter)
* Sensitive to occlusion and lighting changes
* Prone to ID switches in crowded scenes
* Thresholds need tuning for each video
* No appearance (Re-ID) learning

**9) Techniques Used**

* **YOLOv8 + Custom Trained Model (**best.pt**)**
  + Detects players in the soccer video.
* **BoT-SORT Tracker (**botsort\_custom.yaml**)**
  + Assigns consistent Track\_IDs to players across frames.
* **OpenCV**
  + Handles video reading, drawing bounding boxes, and writing annotated video.
* **CSV Logging**
  + Logs Frame, Track\_ID, X, Y, Width, Height every second for tracking data.

### ****Pros****

* **High Accuracy Detection**  
  YOLOv8 with a fine-tuned model ensures reliable player detection.
* **ID Consistency Across Frames**  
  BoT-SORT keeps IDs consistent, even during overlaps and occlusions.
* **Efficient Logging**  
  Frame-level tracking data is logged once per second to reduce size.
* **Clear Visual Output**  
  Green boxes with ID labels allow easy human verification of tracking.
* **Real-Time Capable**  
  Optimized for reasonably fast processing using efficient models.

### ****Cons****

* **Limited Object Classes**  
  Only tracks players (classes=2). Ball, referee, etc., are ignored.
* **Hard-Coded Parameters**  
  Confidence, IOU thresholds, and FPS interval are fixed — not adaptive.
* **No Error Handling**  
  If video or model paths are incorrect, the code crashes silently.
* **Single-Class Limitation**  
  Difficult to distinguish between team colors or referee without additional logic.
* **No Smoothing/Post-Processing**  
  Bounding boxes and IDs can jitter without a temporal filter.

**10)Techniques Used**

* **YOLOv8**: For detecting players in each frame.
* **ByteTrack**: For assigning consistent player IDs across frames.
* **Tracking-by-Detection**: Uses YOLO + Tracker per frame.
* **CSV Logging**: Saves player positions once per second.
* **OpenCV Visualization**: Draws boxes and IDs on players.

### ****Pros****

* Fast and accurate (YOLOv8).
* Tracks multiple players reliably (ByteTrack).
* Easy post-game analysis via CSV.
* Simple to visualize and debug.

### ****Cons****

* Loses identity during occlusion or overlaps (ID switch).
* No team or jersey detection.
* Tracking only while player is visible.
* Misses fast actions between logged frames.

**11) Techniques Used**

1. **YOLOv8 (Object Detection)**
   * Detects players (class 2) in each video frame.
   * best.pt model used (likely custom-trained).
2. **BoT-SORT (Object Tracking)**
   * Assigns unique track\_id to each detected player.
   * Maintains player identities across frames.
3. **Custom ID Remapping**
   * Converts tracker IDs to consistent Player\_IDs using a dictionary.
   * Tracks reappearance/disappearance status.
4. **Logging to CSV**
   * Records frame-wise player data: position, size, status.
5. **Video Annotation**
   * Draws bounding boxes and labels with Player\_ID on output video.

### ****Pros****

* **Accurate Tracking:** BoT-SORT effectively tracks multiple players.
* **Consistent IDs:** Custom remapping ensures readable and stable player IDs.
* **Efficient Detection:** YOLOv8 is fast and optimized for real-time performance.
* **Status Monitoring:** Tracks reappearance/disappearance of players.
* **Structured Output:** Saves data in CSV for post-analysis or model training.

### ****Cons****

* **Occlusion Challenges:** Overlapping players can confuse the tracker.
* **ID Switches Possible:** Trackers may reassign IDs incorrectly on reentry.
* **Detection Class Limitation:** Only class 2 (player) is used; others like ball/referee are ignored unless customized.
* **Accuracy Depends on Model Training:** Poorly trained model may lead to missed or false detections.
* **Processing Time:** Real-time tracking is CPU intensive and may lag on large videos or high FPS.

**12) Techniques Used**

1. **Screen recording/playback via VLC**
2. **Video encoding in AVI format**
3. **Frame-by-frame visual analysis (pause/play)**
4. (Possibly) Computer vision or object tracking (if the video involves detection)

### ****Pros****

* Easy and accessible tools (e.g., VLC)
* AVI allows high-quality frames
* Manual analysis gives control over what you see
* Good for documentation or demonstration

### ****Cons****

* AVI files are large and outdated
* VLC may show overlays during playback
* Manual frame analysis is time-consuming
* No automation (if no CV tools used)

****Challenges I encountered****

### **1. **ID Interchanging in Centroid-Based Tracking****

When two or more players come very close to each other, their centroids overlap or become nearly identical. This causes the tracking logic to **misidentify players**, resulting in **ID switching**. Since the matching is based on distance and appearance features (like color and height), highly similar or overlapping detections confuse the system.

### 2. ****High Frame Rate Complications (e.g., 50 FPS)****

Increasing the frame rate theoretically improves accuracy by reducing motion between frames. However, it introduces **unexpected ID inflation**, where **extra or unusually large IDs** are assigned. This occurs because:

* With 50 frames in 1 second, **subtle movements or noise** in detection can be misinterpreted as new objects.
* It's difficult to **visually verify such fast transitions**, leading to unexplained behavior.

Despite this, higher frame rates often **improve detection granularity**, especially for fast-moving subjects.

### 3. ****Bounding Box Instability with DeepSORT****

When using DeepSORT (or similar algorithms), the **bounding boxes sometimes expand unnecessarily** even when the player has not moved much. This happens due to:

* **Rapid frame transitions** with minor movement, leading to **predictive noise** in tracking.
* The algorithm trying to **"guess"** object motion, which may not align with actual movement.

This can result in **visual inconsistency** and tracking errors.

### 4. ****Overlapping Detections Causing Drawing Conflicts****

When multiple centroids are very close, **drawing circles** around each can cause **visual clutter** or overlapping annotations. To manage this, a minimum distance threshold is applied, but this sometimes leads to **valid detections being skipped for drawing**, reducing clarity.

**5. Appearance-Based Matching Limitations**

Using **dominant color (HSV hue)** and **bounding box height** as identity features works well in general. However:

* Lighting changes or camera exposure affect hue reliability.
* Similar jerseys across teams or players make visual distinction difficult.

This limits the tracking performance in real-world noisy conditions.

### 6. ****Webcam Limitations in Real-Time Mode****

While transitioning to real-time mode using the webcam:

* **Frame drops** or **delays** occur depending on hardware capability.
* The webcam may not deliver a consistent resolution or frame rate, affecting model accuracy.
* CPU/GPU usage can spike, leading to reduced performance.
* ****If incomplete, describe what remains and how you would proceed with more time/resources:****

**I explored numerous unique approaches to achieve the project goals, such as color mapping with player IDs, bounding box centroid-based tracking, circular bounding box tracking, jersey number tracking, ByteTrack, BoT-SORT, and others—each with its own strengths and limitations. I experimented with various methods to find the most effective solution. However, due to limited resources, especially the restricted GPU availability on Google Colab, I had to run everything on the CPU, which affected the efficiency of my experiments. Additionally, my college exams further constrained the time I could dedicate to the project. Despite these challenges, I gave my best effort within the given limitations. I am assuming that you also understand that there is no shortcut to achieving high accuracy, low time complexity, and optimal performance—it's a thoughtful, resource-intensive process. If given this internship opportunity, I am committed to doing my best and continuously improving.**

****I explored numerous unique approaches to achieve the project goals, such as color mapping with player IDs, bounding box centroid-based tracking, circular bounding box tracking, jersey number tracking, ByteTrack, BoT-SORT, and others—each with its own strengths and limitations. I experimented with various methods to find the most effective solution. However, due to limited resources, especially the restricted GPU availability on Google Colab, I had to run everything on the CPU, which affected the efficiency of my experiments. Additionally, my college exams further constrained the time I could dedicate to the project. Despite these challenges, I gave my best effort within the given limitations. I understand there is no shortcut to achieving high accuracy, low time complexity, and optimal performance—it's a thoughtful, resource-intensive process. If given this internship opportunity, I am committed to doing my best and continuously improving.****