Impact College of Engineering and Applied Sciences

Department of Artificial Intelligence and Machine Learning

AI Internship Project Report

Player Re-Identification in Sports Footage

(Single Feed – Option 2)

Submitted to: Liat AI  
Submitted by: Bhargavi K M  
Date: 13 July 2025

# Table of Contents

1. Abstract ................................................. 2

2. Introduction ............................................ 3

3. Problem Statement ....................................... 4

4. Dataset & Inputs ........................................ 5

5. Object Detection Model (YOLOv11) ....................... 6

6. Tracking Algorithm (StrongSORT) ....................... 7

Conclusion ............................................ 15

References ............................................ 16

# Abstract

Player re-identification (Re-ID) in sports analytics refers to the ability to track and maintain the identity of each player throughout a game, even when players temporarily leave the frame or become occluded. This capability is critical for performance analysis, tactical breakdowns, and data-driven coaching. In this internship project, we implemented a computer vision system using YOLOv11 and StrongSORT that enables real-time player tracking in a single 15-second sports video. The system ensures that when players re-enter the frame after leaving, they retain their original identity. We leveraged a fine-tuned object detection model for player detection and integrated DeepSORT-based identity tracking for re-identification. The final output includes a processed video showing unique and consistent player IDs across frames, demonstrating a real-time sports tracking scenario with high accuracy and low latency.

# 2. Introduction

Modern sports have undergone a significant transformation in recent years due to the rise of **data-driven decision-making** and **computer vision technologies**. Sports teams, broadcasters, and analysts increasingly rely on automated tools for **player tracking**, **tactical analysis, performance measurement**, and **event detection**. At the heart of this transformation lies the critical challenge of **player identification and re-identification (Re-ID)** — i.e., being able to track each individual player across time, even as they move dynamically within the field of play.

### What is Player Re-Identification?

**Player re-identification** refers to the ability of a system to maintain a **consistent identity (ID)** for each player, even as they move in and out of the camera view, or when occluded by other players or objects. The goal is simple but challenging: if Player 7 disappears for a few seconds due to camera angle or obstruction, the system should correctly re-assign them the same ID when they reappear.

This is particularly hard in **real-time sports video** scenarios, where:

* The **players often look similar** (same uniforms, similar heights)
* The **camera is continuously panning, zooming, or shaking**
* **Multiple players overlap**, creating occlusion and confusion
* Players **leave and re-enter** the camera frame

### 🎯 The Challenge in the Liat AI Assignment

In most advanced systems, this task is simplified using **multi-camera setups** and **player-wearable sensors**. However, in the real world — especially at non-professional levels — such infrastructure is rarely available. This is the exact scenario simulated by the **Liat AI internship assignment:**

**Track and re-identify players using a single 15-second video clip**, where the camera feed is from a **single point of view** and **players may leave and re-enter** the frame.

Thus, the main challenges we need to solve are:

* **Detection** of players using a computer vision model
* **Assigning unique IDs** to each player
* **Tracking the same ID** across frames (even after disappearance)

This internship project is designed to test not just coding ability but also understanding of **applied AI, real-world modeling, and system design under constraints.**

### Tools & Techniques Selected

To solve this real-world challenge effectively, two key AI components are used:

#### 1. ****YOLOv11**** – For Player Detection

* A fast, real-time object detection model
* Trained to specifically identify players and ball in sports scenarios
* Provides bounding boxes with confidence scores

#### 2. ****StrongSORT**** – For Player Tracking & Re-ID

* An enhanced version of DeepSORT (Simple Online Realtime Tracker)
* Uses **motion prediction + appearance features** to maintain identity
* Re-assigns the same ID to players who temporarily disappear

Together, YOLOv11 and StrongSORT form a reliable pipeline to:

* **Detect players in each video frame**
* **Track them over time with consistent IDs**
* **Re-identify them upon re-entry after occlusion or exit**

# 3. Problem Statement

In the realm of sports analytics and computer vision, one of the most crucial and complex tasks is **player tracking and re-identification (Re-ID).** The primary goal is to automatically detect and consistently track each individual player across time — even when they leave the camera frame or become momentarily occluded.

In this internship assignment provided by **Liat AI,** the challenge is further intensified by simulating a realistic but **resource-constrained scenario:** analyzing a **single 15-second sports video captured from a single camera angle**, without any access to synchronized multi-camera setups, player beacons, or manual annotations.

### 🎯 Core Objectives

The specific **technical objectives** of the assignment are as follows:

1. **Detect players in each video frame**  
   Use a pre-trained, fine-tuned object detection model (YOLOv11) to identify all visible players in every frame of the video. The model should return bounding boxes with confidence scores, allowing for the localization of each player at every timestamp.
2. **Assign unique IDs to each detected player**  
   Implement a tracking module that associates detections across frames. Each player should be assigned a **distinct identity (ID)** at the beginning of the video, and this ID should remain constant throughout their presence.
3. **Maintain the same ID even when players leave and re-enter the frame**  
   One of the most challenging aspects of Re-ID is ensuring that when a player **goes out of the frame temporarily** — due to panning, zooming, or occlusion — the system can correctly **re-assign the same ID** when they reappear. This is essential for reliable long-term tracking.

# 4. Dataset & Inputs

To develop and evaluate the player re-identification system effectively, this project relies on two core input components:

1. A **real-world sports video** simulating gameplay captured from a single broadcast-style camera angle.
2. A **custom-trained object detection model,** specifically fine-tuned to recognize players and the ball in such video footage.

These two inputs form the **foundation of the entire pipeline**—from detection to tracking—and ensure the system operates under practical constraints resembling real-world broadcast environments.

### 🎥 Video Input

The primary input for the system is a **15-second sports video** file named:

15sec\_input\_720p.mp4

#### 🔸 Video Specifications:

| **Property** | **Value** |
| --- | --- |
| **Duration** | 15 seconds |
| **Format** | MP4 |
| **Frame Rate** | 30 frames per second |
| **Resolution** | 1280x720 (HD, 720p) |
| **Aspect Ratio** | 16:9 |
| **Content** | Multiple players in motion, simulating real gameplay |

This short clip includes realistic **game dynamics** such as:

* Players **moving across the frame**
* **Temporary occlusions** as players overlap
* **Camera motion**, such as slight panning or shifts
* **Re-entries** where players leave and return to the frame

These attributes make the clip highly relevant for evaluating **real-time tracking and re-identification**, where identity consistency matters.

### Object Detection Model – YOLOv11

The video frames are processed using a **fine-tuned object detection model** provided by Liat AI, based on the **YOLOv11 architecture**. YOLO (You Only Look Once) is a widely adopted family of models known for **real-time object detection**.

The model file used is:

yolov11\_custom.pt (PyTorch format)

#### 🔸 Model Features:

| **Attribute** | **Details** |
| --- | --- |
| **Architecture** | YOLOv11 (Ultralytics version) |
| **Format** | .pt (PyTorch model) |
| **Classes** | 0: Player, 1: Ball |
| **Training Base** | Fine-tuned on sports video datasets or custom annotations |
| **Speed** | Optimized for real-time inference (~25–30 FPS on GPU) |

The model was pre-trained and optimized for **sports-specific scenarios**, ensuring higher detection confidence when dealing with:

* Players wearing similar uniforms
* Balls that are small or partially visible
* Wide-angle sports video scenes

### 🔍 Why These Inputs Are Important

The combination of this **short yet complex video** and a **tailored detection model** reflects a scenario commonly encountered in sports analytics:

* Limited footage
* Realistic noise and occlusion
* Time-sensitive processing needs

Using this setup allows the system to be evaluated not only on its **accuracy**, but also on its **speed, robustness, and scalability**.

## ****5. Object Detection Model – YOLOv11****

At the heart of any intelligent vision system lies the ability to accurately identify and locate objects of interest in an image. In the context of this project, the primary task is to detect **players** in every frame of a sports video. This is accomplished using **YOLOv11**, a fast, high-performance object detection model specifically **fine-tuned for sports environments**.

### What is YOLO?

**YOLO (You Only Look Once)** is a family of object detection algorithms known for their balance of speed and accuracy. Unlike traditional models that use multiple stages for region proposal, classification, and refinement, YOLO models perform **all detection tasks in a single neural network pass**—hence the name “You Only Look Once.”

The YOLO family has evolved over several versions (v1 to v8+), with **YOLOv11** being an **experimental fine-tuned variant** derived from Ultralytics’ YOLOv8. YOLOv11 used in this project is **custom-trained to detect players (class 0)** and **football/soccer ball (class 1)** in high-motion, crowded scenes.

### How YOLOv11 Works (Internals)

The detection process using YOLOv11 consists of the following steps:

1. **Image Preprocessing:**
   * Each input video frame is resized (e.g., 640x640 or 720x1280)
   * Normalization is applied to match training distribution
2. **Grid Division:**
   * The frame is divided into a fixed-size grid (e.g., 20x20)
   * Each cell is responsible for predicting bounding boxes for any object whose center falls in that cell
3. **Bounding Box Prediction:**
   * For each cell, the model predicts:
     + Box coordinates (x, y, width, height)
     + Objectness score (confidence that an object exists)
     + Class probabilities (e.g., player or ball)
4. **Non-Maximum Suppression (NMS):**
   * YOLO often outputs multiple overlapping boxes for the same object
   * NMS filters out redundant boxes by retaining only the highest-confidence box per object

### 🏃 Application in This Project

The YOLOv11 model is applied to **every frame of the input video**. The results include:

* **Bounding Boxes**: Rectangles enclosing each detected player
* **Class Labels**: Only class 0 (player) is used; ball is ignored in tracking
* **Confidence Score**: Helps filter weak detections (e.g., threshold = 0.5)

Only detections classified as player are passed forward to the **StrongSORT tracking module**, ensuring efficient processing and reduced ID noise.

### ✅ Advantages of YOLOv11 for Player Detection

| **Feature** | **Benefit in This Project** |
| --- | --- |
| **Speed (Real-Time)** | Processes ~25–30 FPS; ideal for live tracking or short clips |
| **Accuracy** | Fine-tuned on sports video datasets; detects players reliably |
| **Lightweight Model** | Can run on laptops, Colab, or edge devices without major hardware |
| **Low Latency** | Reduces delays between input and output |

### 🔐 Custom Training Details

* The YOLOv11 model was **trained or fine-tuned by Liat AI**
* Likely datasets: Sports videos with bounding box annotations for players
* Likely augmentations:
  + Random crop and scale
  + Horizontal flipping
  + Color jitter
* Final model saved as: best.pt in PyTorch format

### 🔬 Output Format of YOLOv11

For every frame, the model returns a list of detections in this format:

[

[x1, y1, x2, y2, confidence, class\_id],

[x1, y1, x2, y2, confidence, class\_id],

...

]

Where:

* (x1, y1) and (x2, y2) are bounding box coordinates
* confidence is how sure the model is
* class\_id is either 0 (player) or 1 (ball)

In this project, we **filter out all detections with class\_id != 0**, as the focus is exclusively on player tracking.

## ****6. Tracking Algorithm – StrongSORT****

While object detection (via YOLOv11) provides the **location of players in individual frames**, it does not maintain any memory of who the players are between frames. This is where **multi-object tracking (MOT)** comes in.

The purpose of a **tracking algorithm** in this project is to:

* **Assign a unique ID** to each detected player,
* **Track that ID across frames**, even when players move rapidly, overlap, or temporarily exit the view,
* **Re-identify** the player when they return, ensuring the same ID is consistently reassigned.

For this purpose, we use **StrongSORT**, a state-of-the-art real-time tracker based on the DeepSORT framework, enhanced with better re-identification and appearance matching.

### 🎯 Why StrongSORT?

**StrongSORT** (Strong Simple Online and Realtime Tracker) builds upon the classic **DeepSORT** tracker but significantly improves its robustness in complex scenes like sports fields.

#### Key Reasons to Choose StrongSORT:

| **Feature** | **Benefit for This Project** |
| --- | --- |
| Appearance-based tracking | Matches player identity using visual features |
| Occlusion handling | Players can disappear and reappear with same ID |
| Kalman filtering | Predicts player positions when temporarily undetected |
| Scalable & real-time | Can process high-FPS videos in near real-time |

### ⚙ How StrongSORT Works – Step by Step

1. **Input Detections from YOLOv11**
   * For each frame, player detections (class 0) are passed to StrongSORT with bounding box coordinates and confidence scores.
2. **Feature Embedding (Appearance Vector)**
   * StrongSORT extracts a **128-dimensional feature vector** for each detection using a re-ID CNN.
   * This captures visual cues like **jersey color**, **shape**, **size**, and **position**.
3. **Prediction (Kalman Filter)**
   * A **Kalman filter** predicts each track’s next position based on previous motion, helping handle brief occlusions.
4. **Data Association**
   * Using the **Hungarian algorithm**, detections are matched to existing tracks by minimizing a **cost function** (combining spatial and appearance distances).
5. **Track Management**
   * If a detection cannot be matched to any existing track, a new ID is assigned.
   * If a track remains unmatched for max\_age frames, it is deleted.
6. **Re-identification**
   * If a player reappears with a similar appearance, the tracker **re-links them to their old ID**, preserving identity across exits/re-entries.

### Key Parameters

| **Parameter** | **Description** | **Value Used** |
| --- | --- | --- |
| max\_age | Number of frames a track can go unmatched before deletion | 30 |
| nn\_budget | Memory size for appearance vectors | 100 |
| max\_iou\_distance | IOU threshold for spatial matching | 0.7 |
| min\_confidence | Minimum detection confidence from YOLO to consider | 0.5 |

These hyperparameters balance **ID stability**, **recovery speed**, and **false ID reduction**.

### 📌 Integration with YOLOv11

In this pipeline:

* YOLOv11 detects players per frame
* Detections are formatted and passed to StrongSORT as:

[

[x1, y1, x2, y2, confidence]

]

* StrongSORT processes them and returns:

[

Track(id=3, bbox=[x1, y1, x2, y2])

]

The returned tracks are drawn on the video with Player {ID} labels.

### ✅ Advantages for Sports Re-ID

| **Benefit** | **Explanation** |
| --- | --- |
| **Consistent IDs** | Maintains identity across appearances and occlusions |
| **Re-ID enabled** | Works even if players leave and re-enter the frame |
| **Fast inference** | Suitable for real-time or near-real-time applications |
| **Lightweight design** | Can run on laptops or in Colab environments |

### 💡 Limitations

* Similar-looking players (same team uniforms) may occasionally be misidentified
* Re-identification is less reliable when a player re-enters with **different pose or occlusion**
* It does not use jersey numbers or biometric cues (only visual + motion)

**Conclusion**

The ability to consistently track and re-identify players in sports videos is a crucial requirement for modern sports analytics, live broadcasting, and performance monitoring. This project successfully implements a lightweight, real-time pipeline for player re-identification using a single-camera video feed.

Leveraging a custom-trained **YOLOv11** model for detection and an advanced **StrongSORT** tracker for maintaining player identities, the system is able to:

* Detect all players in each frame,
* Assign and maintain unique IDs,
* Re-identify players even after temporary disappearance from the frame.

The approach simulates real-world constraints such as **occlusions**, **camera movement**, and **uniform similarity**, demonstrating that high-quality tracking is achievable without the need for wearable sensors or multi-camera setups.

This solution provides a strong foundation for more advanced analytics like heat maps, pass prediction, and tactical formation analysis. Future improvements could include:

* Incorporating jersey number recognition for ID validation,
* Using transformer-based trackers for long-range temporal coherence,
* Integrating pose estimation for finer-grained player activity analysis.

This project not only addressed a technically challenging computer vision problem but also aligned closely with real industry needs, offering a scalable and efficient system ready for further extension.

## ****References****

Below are the references and resources used in the development of this project:

1. **Ultralytics YOLOv8 Documentation**  
   https://docs.ultralytics.com
2. **StrongSORT GitHub Repository**  
   <https://github.com/dongylove/StrongSORT>
3. Wojke, N., Bewley, A., & Paulus, D. (2017). **Simple Online and Realtime Tracking with a Deep Association Metric (DeepSORT)**  
   <https://arxiv.org/abs/1703.07402>
4. Bochinski, E., Eiselein, V., & Sikora, T. (2017). **High-Speed Tracking-by-Detection Without Using Image Information**  
   <https://arxiv.org/abs/1702.06230>
5. YOLOv5, YOLOv8 PyTorch Implementations by Ultralytics  
   <https://github.com/ultralytics/yolov5>  
   <https://github.com/ultralytics/ultralytics>
6. OpenCV Documentation  
   https://docs.opencv.org
7. Google Colab – Free GPU Environment  
   https://colab.research.google.com/
8. Liat AI Internship Materials and Model Assets  
   (Model link and video provided via Google Drive folder by Liat AI)
9. Python Packages: NumPy, Matplotlib, Seaborn, Torch, torchvision  
   <https://pypi.org>