Name – Prerna Banthiya

Github Submission Link - [github.com/prernabanthiya/RealTime-Player-ReIdentification-and-Tracking](https://github.com/prernabanthiya/RealTime-Player-ReIdentification-and-Tracking)

Created for Liat.ai Internship Task

**Task Opted – 2. Re – Identification in a single feed**

**Objective:**

The task was to build a system that:

* Detects all players in a video.
* Assigns each player a unique ID.
* Maintains consistent IDs even when players leave and later re-enter the frame.

This simulates a real-world sports analytics problem — tracking player movements for post-match analysis.

**Methodology and Planning:**

To address this, I broke the problem down into 3 stages:

1. Object Detection

* Tool Used: YOLOv11 (fine-tuned version provided)
* Why? YOLO models are known for real-time speed and competitive accuracy.
* I used the model to extract bounding boxes for "player" class in each frame with a confidence threshold of 0.4.
* Goal: Get precise player positions in each frame.

2. Tracking & Re-Identification

* Tool Used: DeepSORT tracker
* Why DeepSORT? It combines motion and appearance information — useful for tracking players even when they disappear briefly.
* Parameter Tuning:
  + max\_age = 60: Allows players to leave the frame for 2–3 seconds without losing their ID.
  + n\_init = 3: Reduces ID switching by requiring stable detections over a few frames.
  + max\_cosine\_distance = 0.2: Controls appearance similarity tolerance.
* How? I passed YOLO's output to DeepSORT and got consistent player IDs across frames.

3. Output & Logging

* Created a new video with bounding boxes and player IDs.
* Logged the frame numbers where each player appeared in player\_log.txt.

**Thought Process & Techniques Tried:**

I began by testing YOLO detections frame-by-frame to ensure reliable detection of players. Once confident, I integrated DeepSORT and validated its performance using small parameter sweeps.

Alternative ideas I explored:

* Tried increasing n\_init and max\_age to handle missed detections.
* Considered filtering low-confidence detections to reduce tracker noise.
* Verified consistency by manually tracking one player through the video.

**Challenges Encountered:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Challenge** | |  | | --- | |  |  |  | | --- | | **How I Handled It** | |
| 1. YOLO's detection sometimes skipped players | Lowered the confidence threshold and checked GPU memory constraints. |
| 2. Players re-entered the frame with new IDs | Increased max\_age to retain their past identity for longer durations**.** |
| 3. Bounding boxes fluctuated due to jitter | Considered adding Kalman smoothing (handled internally by DeepSORT). |
| 4. Format mismatch between YOLO and DeepSORT | Converted [x1, y1, x2, y2] to DeepSORT’s expected format: [x, y, w, h]. |

**Outcomes:**

* Successfully generated a video with persistent player IDs.
* Logged when each player appeared during the video.
* Created a simple, reproducible pipeline combining detection and tracking.

**Future Work:**

If given more time and resources, I would:

* Integrate pose estimation to improve ID re-identification when appearances are similar.
* Train a custom re-ID model using triplet loss for sports datasets.
* Build an interactive analytics dashboard to visualize:
  + Time spent in play zones
  + Player interactions with the ball
* Extend to live video input or full-match analysis

**Summary:**

|  |  |  |
| --- | --- | --- |
| Component | Tool Used | Purpose |
| Detection | YOLOv11 | Locate players in each frame |
| Tracking | DeepSORT | Assign and maintain player IDs |
| Visualization | OpenCV | Annotate video + draw bounding boxes |
| Output Logs | Python | Record player appearances |