Player Re-Identification with YOLOv8 and Deep SORT

Approach and Methodology

The objective was to build a robust soccer player tracking and re-identification system. The pipeline was based on:

- **YOLOv8**: Used for accurate real-time detection of players and referees.
- **Deep SORT (Real-Time)**: Provided multi-object tracking with appearance-based feature association.
- **Custom Re-ID Module**: Designed to strictly manage ID count (max 26), re-assign IDs to returning players, and preserve identities despite occlusions or off-screen events.

The methodology involved enhancing appearance features (histogram, texture) for re-ID, refining Deep SORT configuration, and applying strict pre-filtering to reduce false detections. Stability tracking and motion analysis were also used to avoid ID flickering.

Techniques Tried and Their Outcomes

Final Working Pipeline:

- **YOLOv8 + deep-sort-realtime**: Achieved stable tracking, re-ID capability, and strict ID enforcement (max 26: 22 players + 4 referees).
- Re-ID logic matched dormant tracks using cosine similarity and spatial proximity, helping reconnect returning players.

Other Implementations That Failed:

- **Official Deep SORT (external repository)**: Required manual feature extraction using a separate encoder model; extremely slow and unreliable for real-time use.
- **BoT SORT**: Despite improved matching logic, it performed poorly in this setting due to over-aggressive suppression and unstable ID assignment. Tuning the botsort.yaml config didn't yield improvements.

These approaches were dropped in favor of the more controllable and customizable deep-sort-realtime Python implementation

Challenges Encountered

- **Blank Space False Positives**: YOLO occasionally detected shapes in grass lines or shadows as players. Custom validation logic (entropy, texture, edge density) was added to counter this.

- **Re-ID Feature Instability**: Appearance features alone were not enough; positional weighting had to be added to boost re-ID accuracy.

- **ID Recycling**: Preventing duplicate IDs while maintaining a strict max count of 26 was non-trivial and required layered logic.

Limitations & Future Improvements

Initial Success, Later Degradation: Initially, the system correctly identifies players with stable IDs. However, after some time, blank spaces begin to get incorrectly identified as players. Moreover, in certain frames, a single player ends up receiving multiple simultaneous IDs, indicating a flaw in re-identification robustness under complex scenes.

Visualization and Statistics for Debugging: Bounding boxes are color-coded to convey identity roles and assist with debugging:

    - **Green**: IDs 1–22 (Players)

    - **Yellow**: IDs 23–26 (Referees)

    - **Red**: IDs beyond 26 (Excess — should not occur)

After processing the video, detailed statistics are generated: raw detections, filtered detections, valid tracks, total IDs, and timing accuracy. These insights help understand detection behavior and fine-tune the system.

While the system achieves re-ID and ID limit enforcement, a few areas can be improved with more resources:

- **ID Swapping During Occlusion**: Occasionally, IDs still swap when players overlap too long. A Kalman filter extension or temporal re-ID smoothing could help.

- **No True ID Remapping in Deep SORT**: Re-identification doesn't re-assign Deep SORT's internal IDs. A custom tracker or deeper hack into the tracker module would be needed to enforce this.

- **Real-Time vs. Slowed-Down Processing**: Currently, the system slows down the FPS to give enough time for accurate detection and tracking, without dropping any frames. A key improvement would be to achieve detection and tracking at the original video speed (e.g., 15 seconds real-time) while maintaining full accuracy. This would require optimizations in model size, parallel processing, or hardware acceleration.

With more resources, these can be implemented to push it closer to a production-grade system.