Player Re-Identification with YOLOv11, CLIP-ReID, and AURA

Mihika Usgaonker

1 Abstract

This project presents a modular and scalable system for Player Re-Identification in broadcast-style soccer videos. The system aims to maintain consistent IDs for players throughout a 15-second single-camera video, even in the presence of occlusions, visual similarity in uniforms, and varying visibility. It uses a combination of **YOLOv11** for detection, a customized version of **CLIP (ViT-B/16)** for appearance-based embeddings, and a novel **AURA (Anchor Unified Re-Identification Algorithm)** module to handle occlusions and stabilize identities over time. All modules are designed to function in a lightweight environment (Kaggle GPU sessions) without hardcoded assumptions about team colors.

2 Initial Experiments

In early stages, traditional trackers such as **ByteTrack** and **BoT-SORT** were evaluated. While computationally efficient, these trackers primarily rely on bounding box IoU and temporal coherence, which proved insufficient in crowded soccer scenes with highly similar players. ID switches were frequent, especially during overlaps and re-entries.

To address this, visual feature embeddings were explored. Standard feature extractors like **ResNet-18** and **ResNet-50** were tested. However, both failed to capture discriminative features, especially under motion blur, partial occlusion, or similar uniforms. These experiments highlighted the need for a model pretrained specifically on humans and capable of capturing appearance-level cues.

This led to adopting the open-source implementation of **CLIP-ReID** from Konrad Habel's CLIP-ReID repository. It provides a ViT-B/16 model pretrained on large-scale image-text data and fine-tuned on person Re-ID datasets. The model was integrated without the projection head to retain high-dimensional visual embeddings suitable for fine-grained similarity matching.

3 The Approach

The final system is composed of the following modular stages:

3.1 1. Detection and Cropping

- Players are detected using YOLOv11 (‘class=2‘) per frame.
- Detections below 30×30 pixels are discarded.
- Player crops are extracted, flattened, and stored for downstream processing.

3.2 2. Video Decoding and Frame Handling

The input ‘.mp4‘ video is loaded and stored as individual frames using OpenCV. Output frames are rendered and saved as ‘.mp4‘ for visualization.

3.3 3. Jersey Clustering with Green Suppression

- Dominant color is extracted using KMeans ($k = 1$) per crop.
- Crops with dominant green (HSV-based) are optionally filtered.
- KMeans ($k = 4$) is used to form team clusters using non-green crops.
- Green clusters with low variance or low count are excluded from matching.

3.4 4. CLIP Embedding Extraction

- Each valid crop is resized to 224×224 and transformed using CLIP-ReID's augmentations.
- Batched inference is run on GPU using the ViT-B/16 backbone.
- Output is a 768-dimensional embedding per crop, representing visual appearance.

3.5 5. AURA ID Assignment

The **AURA (Anchor Unified Re-Identification Algorithm)** is the core innovation:

- Players are processed in windows of 50 frames.
- For overlapping players (based on IoU ¿ 0.5), an *anchor* is selected based on visibility.
- This anchor retains its ID and acts as a reference for others in the overlap group.
- New players compare their embeddings to the memory bank (cosine similarity $\geq$ 0.85) **within their team cluster**.
- If no match is found, a globally unique ID is assigned.

3.6 6. Rendering and Output

Each frame is overlaid with:

- Bounding boxes
- Unique ID
- Team label
- Frame number

4 Future Work and Extensions

- **Train CLIP on sports datasets:** The current CLIP-ReID model captures mostly global features. Given the nature of soccer footage (low resolution, motion blur), the model should be fine-tuned or pretrained on sports-specific datasets with partial occlusion and varying viewpoints.
- **Experiment with CLIP ViT-L/14:** Larger CLIP backbones (like L/14) have been shown to better capture fine-grained visual features. With more memory, I would compare ViT-B/16 and ViT-L/14 using the same pipeline.

- **Implement occlusion detection (AETHER):** The current pipeline assumes overlaps indicate occlusion but does not explicitly model visibility edges or motion boundaries. I plan to add a separate occlusion handler named **AETHER** which uses edge maps and consistency to track visibility. This would enable better anchor assignment during total occlusion.
- **Post-hoc ID merging:** Even with AURA, over-fragmentation occurs. A final post-processing step using trajectory smoothness or graph-based ID merging can reduce the 68 IDs closer to the ground truth (˜22).
- **Introduce temporal fingerprints:** Embedding temporal cues like stride pattern or movement consistency would aid in resolving ambiguity, especially for players with identical uniforms.

Notes

- All paths in the notebook are defined for Kaggle. Running in Kaggle avoids the need to change directories or handle local I/O.
- The project maintains clean modularity with functions separated by section.
- All models and visualizations were run under GPU memory constraints (Tesla P100 on Kaggle).