Player Tracking and Re-Identification in Sports Video

Project Report: Methodology, Trials, and Lessons Learned

Introduction

This report documents the complete workflow, experiments, and lessons learned while building a robust player tracking and re-identification system for sports video. The goal was to maintain consistent player IDs across frames, even when players leave and re-enter the scene, using modern computer vision techniques.

1. Problem Statement

- **Objective:**
  Track each player in a 15-second sports video, ensuring that players who leave and reappear are assigned the same identity.
- **Challenges:**
  - Low video quality (blur, noise, missed detections)
  - Occlusions and players leaving/re-entering the frame
  - Similar appearances (e.g., same team uniforms)
  - Real-time or near-real-time processing

2. Initial Approach

2.1. Baseline: YOLO Detection + OpenCV Visualization

- Used YOLO (Ultralytics, `best.pt`) to detect players in each frame.
- Drew bounding boxes and assigned IDs based on detection order.
- **Limitation:**
  IDs were not consistent across frames; no tracking or re-identification.

3. Adding Tracking: DeepSORT Integration

3.1. DeepSORT for Multi-Object Tracking

- Integrated DeepSORT to assign persistent IDs to detected players.
- Passed YOLO detections (class index 2 for players) to DeepSORT.
- Visualized bounding boxes and IDs frame-by-frame.

Initial Results

- IDs remained consistent as long as players stayed in frame.
- Frequent ID switches or ghost tracks when players left and re-entered or during missed detections.

4. Addressing ID Switches and Ghost Tracks

4.1. Parameter Tuning

- **max_age:** Lowered to 7–10 to quickly remove ghost tracks when detections were lost.
- **Detection confidence threshold:** Increased to 0.5–0.7 to reduce false positives.
- **Track age filtering:** Only displayed tracks with `track.age ≥ 3` to suppress short-lived, unreliable tracks.

Effect

- Reduced ghost boxes and spurious IDs.
- Some increase in ID switches if detection was missed for several frames.

5. Appearance-Based Re-Identification

5.1. Color Histogram Features

- Extracted color histograms (HSV) for each player's bounding box.
- Maintained a rolling history (last 5 histograms) per track.
- **Limitation:**
  Color histograms were not distinctive enough for similar uniforms or lighting changes.

5.2. Deep ReID Model Integration

- Integrated OSNet (Torchreid) to extract deep appearance embeddings for each detected player.
- Used these embeddings as appearance features for DeepSORT.

- Fallback to color histogram if embedding extraction failed.

Key Improvements

- Significantly fewer ID switches.
- Better re-identification after occlusion or re-entry.
- More robust tracking in crowded scenes.

6. Performance Optimization

- Switched to a CUDA-enabled PyTorch installation to leverage GPU acceleration.
- Resulted in 5–20x faster inference and enabled near real-time processing.
- Ensured all `.cuda()` or `.to(device)` calls were conditional on GPU availability.

7. Visualization and Output

- Drew bounding boxes and IDs for confirmed, persistent tracks.
- Optionally visualized player trajectories by connecting center points across frames.
- Maintained an appearance history dictionary for future analytics.

8. Trials, Errors, and Lessons Learned

8.1. Common Issues and Solutions

Issue	Solution/Adjustment
Ghost boxes after players left	Lowered `max_age`, increased confidence threshold
Frequent ID switches	Added deep ReID, tuned detection/tracking params
Slow processing	Switched to GPU, used lightweight ReID model
ModuleNotFoundError (gdown)	Installed missing dependencies via pip
CUDA errors	Matched PyTorch install to system CUDA version

8.2. Methodological Takeaways

Building a reliable player tracking and re-identification system for sports video is not just a matter of plugging in a detector and visualizing the results. Throughout our experimentation and refinement, several key lessons emerged that shaped the final, robust pipeline:

- **Detection Alone Isn't Enough:**
  Relying solely on object detection (even with a strong model like YOLO) leads to inconsistent IDs, especially when players move quickly, cross paths, or leave and re-enter the frame. Detection provides the "what" and "where," but not the "who" over time. True tracking requires more than just spotting players in each frame—it demands a mechanism to persistently link those detections across time.

- **The Power of Combining Motion and Deep Appearance Features:**
  Real-world re-identification is challenging due to similar uniforms, variable lighting, and frequent occlusions. By fusing motion cues (like predicted position from Kalman filters) with deep appearance features (from a ReID network), the system can more confidently and accurately maintain player identities. This synergy dramatically reduces ID switches and enables the tracker to recover from missed detections or temporary occlusions.

- **Parameter Tuning Is Not One-Size-Fits-All:**
  The performance of tracking systems is highly sensitive to parameters like max_age (how long to keep a lost track alive) and the detection confidence threshold. These must be tailored to the quality and nature of the video. For noisy or low-quality footage, lower max_age and higher confidence thresholds help reduce ghost tracks and false positives. In higher-quality scenarios, more lenient settings can help the tracker bridge brief occlusions without losing track continuity.

- **Fallback Mechanisms Are Crucial for Edge Cases:**
  Even the best deep learning models can fail—due to poor lighting, extreme occlusion, or rare visual conditions. Incorporating fallback features, such as color histograms, ensures the system remains robust when deep embeddings are unavailable or

unreliable. These simple features act as a safety net, allowing the tracker to make reasonable associations in tough situations.

- **GPU Acceleration Unlocks Real-Time and Scalable Analytics:**
  Deep learning-based detection and ReID are computationally intensive. Running these models on a CPU is often prohibitively slow, especially for high-resolution or long-duration videos. Leveraging GPU acceleration transforms the workflow, enabling real-time or near-real-time processing and making it feasible to analyze large datasets or live streams. This not only improves user experience but also opens the door to advanced analytics and interactive applications.

In summary:
Robust player tracking and re-identification is a holistic challenge. It requires thoughtful integration of detection, motion, and appearance cues, careful parameter tuning, and practical engineering considerations like fallback features and hardware acceleration. Each lesson learned along the way contributed to a system that is both accurate and resilient, ready for the complexities of real-world sports analytics.

9. Final Pipeline Overview

1. **Detection:** YOLOv8 (`best.pt`)
2. **Tracking:** DeepSORT with motion and appearance association
3. **ReID:** OSNet deep embeddings (Torchreid), fallback to color histograms
4. **Track Management:**
   - Filter by confidence and age
   - Remove ghost tracks quickly
   - Maintain appearance history per track
5. **Visualization:** Bounding boxes, IDs, and optional trajectories

10. Recommendations for Future Work

- Fine-tune the ReID model on custom sports data for even higher accuracy.
- Integrate jersey number recognition for direct player ID assignment.

- Extend to multi-class tracking (goalkeepers, referees, ball).
- Add advanced analytics (player statistics, event detection).
- Save annotated videos and tracking data for further analysis.
- Getting a multiangle view or a better video quality can help us identify much better

Appendix: Key Parameters

Parameter	Typical Value	Purpose
`max_age`	7–15	Frames to keep lost tracks before deletion
`conf` threshold	0.5–0.7	Minimum detection confidence for tracking
`track.age`	3	Minimum frames before displaying a track

References

- Ultralytics YOLOv8 documentation
- DeepSORT tracking algorithm
- Torchreid (OSNet) model documentation
- OpenCV Python library

End of Report

⁂