📌 Cross-Camera Multi-View Player Tracking & Re-Identification

This project performs real-time multi-view player tracking and re-identification using synchronized Broadcast and Tacticam videos. It assigns globally consistent IDs to the same players across two different perspectives.

Problem Statement

Given two perspective-shifted video recordings of the same match (Broadcast and Tacticam), the goal is to assign consistent player IDs to the same individuals across both views — even under occlusions, exits/re-entries, or missed detections.

Key Challenges

- Occlusion and missed detections
- Players entering/leaving frame
- Similar appearance among players
- Perspective distortion across views
- Cross-camera synchronization

Key Components Used

Component	Role
YOLOv8	Player, referee, ball, GK detection
CLIP ViT-B/32	Appearance embedding (512-D) via OpenCLIP
MediaPipe Pose	Pose keypoint extraction (33×3 = 99D)
EasyOCR	Jersey number recognition
KMeans	Team clustering via hue
Hungarian Algorithm	1:1 player matching across views
Tracklet Buffer	Maintains short-term memory for unmatched IDs

Component	Role
Offset Smoother	Synchronizes video frames over time

⚙️ System Workflow

1. ⌛ Frame Alignment

- For every Broadcast frame, we align it with the best-matching Tacticam frame (±SYNC_WINDOW) using CLIP embedding similarity.
- A rolling median offset smoother (ROLL_WINDOW) stabilizes the cross-view alignment over time.

2. 🎯 Detection + Feature Extraction

For every detected player, we extract the following features:

Feature	Description	Purpose
bbox	Bounding box (x1, y1, x2, y2)	Spatial location
emb (CLIP)	512D embedding via CLIP ViT-B/32	Appearance-based matching
pose	99D keypoints from MediaPipe Pose	Shape/body posture matching
hue	Mean hue (from HSV)	Team separation
pos	Normalized center (x,y) of bbox	Positional matching
den	BBox area / frame area	Density awareness in crowded scenes
jnum	OCR result (jersey number)	Strong ID hint (if legible and consistent)

All feature extraction and pose logic is encapsulated in:

- clip_extractor.py — for CLIP embedding
- pose_utils.py — for pose feature extraction via MediaPipe

Note: Pose vectors are extracted every few frames to reduce compute and smoothed using history from tracklet memory.

🧮 Total Similarity Score

The matching score between two detections is:

score =
W_CLIP × CLIP cosine similarity
+ W_POSE × normalized pose similarity

+ W_HUE × hue closeness
+ W_POS × spatial proximity
+ W_DEN × density match
+ (optional +3 bonus if jersey numbers match)

Weights used:

Component	Weight
W_CLIP	1.0
W_POSE	0.5
W_HUE	0.4
W_POS	0.3
W_DEN	0.2

Threshold for match: SIM_THRESH = 1.5

Why 1.5?

- Empirically determined to avoid false matches while retaining re-identification accuracy.
- Too low → ID swaps; too high → re-ID fails after occlusion.

👥 Team Clustering

- KMeans is applied on hue values from detected boxes.
- Two clusters are formed: team A and team B.
- Ensures matching is only done within the same team (e.g., red ≠ blue).

🔀 Player Matching Logic

Step-by-step priority for ID assignment:

1. Jersey number + team match (if OCR available)
2. Memory match via tracklet buffer (based on CLIP similarity)
3. Hungarian Algorithm across current frame
4. Assign new ID

Tracklet buffer stores embeddings, pose vectors, and team cluster info.

Visualization

- Broadcast and Tacticam views are horizontally stacked
- Each player box is annotated with:
  - Global track ID
  - Jersey number (if OCR successful)
  - Color-coded per team

Output video: cross_view_output.mp4

Enhancements Added

Feature	Description
CLIP ViT-B/32	Lightweight 512D appearance embedding
pose_utils.py	Separated reusable pose extraction logic
OCR Confidence Filter	Accept only short digit-only strings with contrast
Team-aware OCR Mapping	Prevents same jersey reused across teams
Pose Smoothing	Pose vectors averaged over recent tracklet history
Frame Sync Buffer	Smooth ±frame offset using CLIP embedding
1:1 Matching	Hungarian algorithm ensures globally optimal match

---

How to Run

Place input files:

- broadcast.mp4
- tacticam.mp4
- best.pt (YOLOv8 weights)

Install dependencies:

```
pip install opencv-python numpy easyocr ultralytics open_clip_torch scikit-learn mediapipe
```

Run:

```
python project1.py
```

Output saved as:

```
cross_view_output.mp4
```

📂 File Summary

File	Description
project1.py	Main script
clip_extractor.py	CLIP ViT-B/32 wrapper
pose_utils.py	Pose keypoint extraction via MediaPipe
best.pt	YOLOv8 model weights
broadcast.mp4	Main video input (TV view)
tacticam.mp4	Tactical side video
cross_view_output.mp4	Output video with tracked players
