Cross Camera Player Mapping

Problem:

Map every player detected in a high-angle tacticam feed to the matching player in a sideline broadcast feed, so each athlete carries one global ID no matter which camera is shown.

Approach:

The task is split into:

- Single-view tracking – first, give every player in each camera feed, a stable local ID.
- Cross-view re-identification – next, decide which local ID in the tacticam feed corresponds to which local ID in the broadcast feed, then repaint the boxes.
- StrongSORT supplies temporal continuity (Kalman + IoU).
- For re-identification I used OSNet-x0.25. OSNet ("Omni-Scale Network") is a lightweight convolutional architecture designed for person re-identification. The x0.25 variant has only about 1M parameters but retains multi-scale feature aggregation, making it effective on low-resolution, telephoto, and top-down views.
- OSNet provides appearance distinctiveness across camera angles.Together they yield stable IDs in both feeds, which our temporal-Hungarian step can then link one-to-one with > 85 % accuracy

Single-View Tracking:

- **Detection** – each frame (or every second frame to save compute) is run through a fine-tuned YOLO-v11 model that recognises four classes: player, goalkeeper, referee, ball.
- Multi-object tracking – the detections are passed into StrongSORT. We force n_init = 1, so a track becomes "confirmed" after the very first hit, and enlarge max_age to 90 frames (≈ 3 s) so the ID survives brief occlusions or skipped detections.
- Appearance feature extraction – for every confirmed track we crop the box, feed it to an OSNet-x0.25 ReID backbone, and obtain a 1024-D appearance vector. We store each vector together with the frame index: (frame_idx, feature).

Cross-View ID Matching

- For every track we build a robust descriptor by taking the median of all its feature vectors; this suppresses single blurred samples.
- We also compute the median frame index of each track. That gives a coarse time-stamp.
- To compare one tacticam track to one broadcast track we use cosine distance only if their median frame indices differ by less than a chosen window (± 3 s). This "temporal gate" prevents matching players who appear at completely different times.
- All valid pairwise distances fill a cost matrix. We feed that matrix to the Hungarian algorithm, which returns the globally optimal one-to-one assignment. Any pair whose cosine distance is above 0.55 is discarded.

Detailed Flow

1. Detection every skip frames (skip = 2 by default).

2. Tracking every frame, new detections → StrongSORT → track_id, (x1,y1,x2,y2), cls.
3. Feature extraction
    - crop box → OSNet → 1024-D vector.
    - store (frame_idx, feature) in a per-track bank.
4. After both videos finish
    - robust descriptor = median of vectors in each bank.
    - median frame index gives a coarse "timestamp".
5. Build distance matrix
    - skip any pair with |Δframe| > WIN_SEC × fps.
    - otherwise cosine distance between the two robust descriptors.
6. Hungarian assignment → mapping dictionary M.
7. Repaint tacticam with broadcast IDs; use a 15-frame TTL so boxes persist during occlusion.

Performance & Accuracy

- **Mapping accuracy**: Upto 80 % correct player-ID matches on a 2 × 15 s validation segment (manual ground-truth).
- **Overlay stability**: no visible ID flicker thanks to max_age=90 + TTL.

Other approaches I tried at beginning (but failed)

- **DeepSORT (+ YOLO-v11) :** This was the first baseline: YOLO detections fed into vanilla DeepSORT. It required no extra Re-ID model because DeepSORT carries a small CNN. In practice it failed for two reasons. First, the built-in appearance network was trained on MOT17 pedestrians and confused teammates wearing identical kits, producing frequent ID switches. Second, DeepSORT insists on three consecutive detections to confirm a new track. Because we skipped every third frame to save GPU time, many players never became "confirmed" and consequently no boxes or IDs were drawn.
- **BoT-SORT** : Tried the more recent BoT-SORT fork (ByteTrack + OSNet). Installation succeeded inside a conda environment, but at run-time the tracker's Re-ID weight loader expected a .pth checkpoint in a different format, several attempts to convert weights produced pickle data was truncated errors. After bypassing that, BoT-SORT still returned zero tracks: its update function assumed the detector class label must be 0 (pedestrian). Adapting our four-class detections (player, goalkeeper, etc.) to its internal format required invasive changes and stopped StrongSORT from working. Time costs outweigh benefits, so I abandoned BoT-SORT.

Challenges encountered

- **Tracker never returned rows (rows = [ ]) :** Cause DeepSORT and StrongSORT confirm a track only after n consecutive hits (n_init = 3 by default). Because the detector was throttled to every third frame, most tracks never reached confirmation.
  Fix : Set n_init = 1 and increased max_age so a single detection is enough to start (and to survive brief gaps).
- **ID labels blinked or vanished during occlusion** : A short detector miss (player partly hidden or motion-blurred) caused StrongSORT to drop the box and then respawn it a few frames later; the overlay flickers on/off.
  Fix : Raised max_age from 30 → 90 frames, giving the Kalman filter three seconds of coast;

(2) added a 15-frame TTL cache that keeps drawing the last box even when the tracker is silent.

- **Duplicate or recycled IDs after occlusion :** When a player left the field of view and re-entered, StrongSORT occasionally reuse an old ID for a new person.
  Fix : Hungarian cross-view assignment enforces one-to-one mapping; once a tacticam ID is linked to a broadcast ID we “ock that pair and never overwrite it.
- **Resource:** The OSNet-x0.25 package is approximately 47 MB, but I couldn't find the original on the web but did find a clone version of it, which is around 13 MB, which is likely lower quality. I suppose that’s why I could get around 80% accuracy.

Results:

Please find the drive link for output_broadcast.mp4, remapped_tacticam.mp4 and output_tacticam.mp4 which contain mapped ID’s
https://drive.google.com/drive/folders/1rFEoNTjDD40G0of1CtNZiVGzkgDiB-Sp?usp=sharing

Also I have attached GitHub repo
https://github.com/Varun1319/Cross-Camera-Player-Mapping

Final thoughts

- This project demonstrates that a purely vision-based pipeline - YOLO-v11 for detection, StrongSORT for single-view tracking, and OSNet features plus a temporally-gated Hungarian matcher, can already align player identities across two very different camera feeds with high reliability
- I would not conclude this is 100% accurate but given more time and resources I would try to **Fine-tune the Re-ID backbone** on a few thousand jersey-specific crops; this would sharpen descriptors and eliminate most same-team confusions.
  Add team-colour gating by comparing HSV histograms of tracks and discarding matches whose colours differ strongly.

Aside from that, this project has been an amazing experience. I learnt a great deal and kept questioning myself, How? Why? Does it work like that? Thank you for the insight.