ASSIGNMENT REPORT

Objective

The primary objective of this project was to map football players between two different video feeds — a **broadcast view** and a **tacticam view** — in such a way that each player retained a consistent identity (player_id) across both perspectives. I was provided with a YOLOv11 model fine-tuned specifically to detect players and the ball. My task was to leverage this model and build a reliable mapping pipeline based on detections, tracking, and feature matching.

My Approach & Methodology

1. Player Detection

I started by integrating the provided fine-tuned YOLOv11 model using the Ultralytics library. Since the model was trained to detect players as class 2, I configured the detector to filter only those detections. To ensure precision, I added a confidence threshold filter, which I eventually tuned to > 0.91 after experimentation.

2. Player Tracking

Once players were detected in each frame, I implemented a **centroid-based tracker**. This tracker assigned persistent track_ids to each detected player across time. This ensured that each player maintained a consistent ID within a single video feed.

3. Feature Extraction

To uniquely identify players across the two camera views, I extracted **visual features** from the cropped player bounding boxes in each frame. These features (e.g., color histograms or embeddings) were averaged over all frames for each player to create a single, stable descriptor vector.
In my pipeline, **spatial matching was primarily handled through centroid-based tracking** and bounding box analysis. By comparing the positions of player detections across consecutive frames, I was able to maintain consistent track IDs for each individual. Even after shifting to visual feature–based matching, spatial cues remained essential—helping to associate detections that appeared in expected zones of the field and

eliminating false positives outside valid play areas. This ensured that player identities were not only visually similar but also made sense based on their physical location on the field.
Temporal consistency played a key role in both tracking and identity recognition. Instead of relying on single-frame detections, **I averaged visual features for each player across multiple frames to create stable identity representations**. This smoothing helped account for momentary occlusions, motion blur, and appearance changes due to lighting or pose variation. Additionally, the tracker preserved player identities over time, allowing the system to link detections across a sequence and reduce identity swaps during continuous play.

4. Cross-Camera Mapping

After gathering features for all players from both feeds, I calculated pairwise **cosine similarity** scores between tacticam and broadcast player vectors. I then used the **Hungarian algorithm** to perform optimal bipartite matching, mapping each player from the tacticam video to their most likely counterpart in the broadcast video.

<u>Techniques I Tried & Algorithms Tested</u>

- I initially implemented **centroid-based tracking**, where players were tracked by comparing the spatial proximity of their centroids across consecutive frames. It worked well in isolated cases but failed when players crossed paths or occluded each other.
- To improve identity retention, I switched to a **visual feature–based tracking** approach. I extracted appearance features (e.g., color histograms or embeddings) from each player's bounding box and averaged them over time. This made tracking more robust in crowded or ambiguous frames.
- For mapping players across camera views, I computed **cosine similarity** between the averaged features of tracked players in the broadcast and tacticam videos. This allowed me to evaluate how visually similar two players were.
- I used the **Hungarian algorithm** to perform the final one-to-one player matching across views. It guaranteed the most optimal

assignment based on similarity scores and handled cases better than simple thresholding or greedy matching.

- I tuned the **confidence threshold** for the YOLOv11 detector extensively. Lower values like 0.4 or 0.6 introduced many false positives (especially from the crowd), whereas a high threshold of **0.91** significantly improved detection precision with minimal impact on recall.
- To filter out bad detections, I added a **minimum bounding box size constraint**. This helped eliminate small, partial detections that typically weren't actual players.
- I implemented **temporal feature smoothing** by averaging visual features over multiple frames. This made identity features more stable and resilient to frame-level noise like motion blur or lighting shifts.
- Later, I integrated **jersey number OCR using EasyOCR** to supplement visual features. When jersey numbers were detected clearly, they provided a deterministic method of player matching and greatly improved cross-view consistency.

Challenges I Faced & How I Solved Them

- **Over-detection of Non-Players:**
  Initially, the YOLOv11 model detected a large number of false positives — including referees, crowd members, and even background textures. In some frames, it returned more than 70 detections.
  Solution: I tackled this by increasing the detection confidence threshold to **0.91**, which drastically reduced false positives while still retaining actual players. I also filtered out small bounding boxes that were unlikely to be real players.
- **Missed or Weak Detections for Real Players:**
  With a high confidence threshold, some real players — especially those far from the camera or partially occluded — were not detected.
  Solution: I applied **temporal feature smoothing** by aggregating player features across multiple frames. This helped recover

consistency for players even if they were briefly missed in a few frames.

- **Bounding Box Inaccuracy and Misalignment:**
  Some bounding boxes extended beyond frame boundaries or were loosely aligned, especially in wide-angle shots.
  Solution: I added clamping and validation checks to ensure bounding box coordinates stayed within image dimensions, and skipped detections that were clearly invalid or too small.

- **Tracking Instability in Crowded Scenes:**
  The initial centroid-based tracking method often broke down when players crossed paths or occluded each other, causing frequent identity switches.
  Solution: I transitioned from a purely spatial tracking approach to a **visual feature–based method**, where tracking relied on player appearance, making it more robust to overlap and jitter.

- **Cross-View Identity Mapping Errors:**
  Mapping players between the tacticam and broadcast feeds was challenging, especially when multiple players looked similar or had no distinctive features.
  Solution: I implemented **cosine similarity + Hungarian matching** for optimal identity mapping and supplemented this with **OCR-based jersey number recognition**. This added a deterministic layer to the identity assignment when jersey numbers were visible.

- **Model Re-downloading or Unnecessary Processing:**
  The model occasionally re-downloaded or reprocessed videos unnecessarily, especially when running multiple times.
  Solution: I added checks to skip the download if the model file already existed and made sure to cache intermediate steps like extracted features to avoid repeated processing.

- **Incorrect Output Videos (Corrupted or Misaligned Annotations):**
  At one point, the output videos had corrupted frames or incorrect bounding boxes.
  Solution: I revised the output writer logic to ensure consistent frame dimensions, correct codec settings, and only wrote annotated frames when tracking and detection were valid.

How I Can Make It Better

- **Replace Basic Tracking with Deep SORT or ByteTrack:**
  My initial tracking was either centroid-based or feature-averaged, which worked decently in simple scenes. However, in crowded or fast-motion scenarios, more advanced multi-object tracking (MOT) algorithms like **Deep SORT** or **ByteTrack** could offer better robustness by combining appearance, motion, and temporal consistency at a deeper level.
- **Incorporate a Re-Identification (Re-ID) Network:**
  Instead of handcrafted or averaged visual features, I could integrate a **pretrained ReID model** (e.g., OSNet or FastReID). These models generate more discriminative embeddings that are robust across different views, lighting, and camera angles — ideal for cross-camera mapping.
- **Fine-Tune the YOLO Model with Hard Negatives and Uniform Variability:**
  The current detection model sometimes confuses referees, coaches, or the crowd with players. By retraining the YOLOv11 model using **hard negative samples** (non-players that resemble players) and adding data with varying lighting, occlusion, and jersey styles, I could greatly reduce false positives.
- **Optimize Runtime with Batch Processing and Frame Skipping Logic:**
  The current system runs on a frame-by-frame basis. I could explore **batch detection and embedding extraction**, or dynamically **skip frames based on scene stability**, to accelerate processing without sacrificing accuracy.