Player Re-Identification in Sports Footage

Brief Report

This report outlines the approach, methodology, and challenges encountered during the implementation of player re-identification for the "AI Intern" assignment by Liat.ai. The focus is on **Option 2: Re-identification in a Single Feed**.

---

1. Approach and Methodology

The objective was to identify and consistently track players within a single 15-second video clip (15sec_input_720p.mp4), ensuring that players retain their assigned IDs even after temporary occlusions or leaving and re-entering the frame.

The chosen approach leverages the **Ultralytics YOLOv11 model** for object detection, combined with its integrated **ByteTrack** algorithm for multi-object tracking and re-identification. This method was selected for its efficiency and direct compatibility with the provided fine-tuned model.

The core steps involved:

- **Model Loading**: Loading the provided best.pt YOLOv11 model, which is specifically fine-tuned for detecting players, goalkeepers, referees, and the ball.
- **Video Processing**: Reading the input video frame by frame.
- **Integrated Tracking**: Utilizing the model.track() function from the Ultralytics library. This function inherently handles both detection using the YOLO model and subsequent object association across frames using ByteTrack.
- **Persistent IDs**: The persist=True parameter in model.track() is crucial for enabling the re-identification capability, allowing the tracker to remember and re-assign IDs to returning objects.
- **Visualization**: Drawing bounding boxes, unique track IDs, and confidence scores on each player, and marking the ball separately.
- **Output Generation**: Saving the processed frames as a new video file.

---

2. Techniques Used and Their Outcomes

a. Object Detection with Fine-tuned YOLOv11

- **Technique**: The provided best.pt model, a fine-tuned version of Ultralytics YOLOv11, was directly loaded and used for object detection.
- **Outcome**: The model proved highly effective in accurately detecting players, goalkeepers, referees, and the ball in various poses and lighting conditions within the provided video frames. The confidence scores for detections were consistently high.

b. Multi-Object Tracking with ByteTrack

- **Technique**: ByteTrack, a robust and widely-used tracking-by-detection algorithm, was integrated via Ultralytics' model.track() method. Key parameters like persist=True were used to enable continuous tracking and re-identification.
- **Outcome**: ByteTrack successfully assigned unique, persistent IDs to individual players. Through visual inspection of the output video, players retained their IDs even after minor occlusions or brief moments where they might have been less clearly visible. This directly addressed the re-identification requirement.

c. Class Filtering and Visualization

- **Technique**: The model.names property was used to identify the specific class ID for 'player'. A conditional check (if cls_id == PLAYER_CLASS_ID:) was implemented to differentiate players from other detected objects (like the ball, goalkeepers, referees). Players were assigned dynamic, consistent colors based on their track_id, while the ball received a fixed color.
- **Outcome**: The visualization clearly distinguished players by their unique IDs and colors, making the re-identification visually evident. The ball was also correctly identified without a tracking ID, focusing the solution on the player re-identification aspect as per the assignment's primary focus.

---

3. Challenges Encountered

The primary challenge encountered during development was a **misconfiguration of the PLAYER_CLASS_ID**.

- **Issue**: Initially, all detected objects (players and balls) were being incorrectly labeled as "Ball".
- **Root Cause**: The default assumption was that 'player' would correspond to class ID 0. However, after inspecting the model.names output in the terminal ({0: 'ball', 1: 'goalkeeper', 2: 'player', 3: 'referee'}), it was revealed that 'player' actually corresponded to class ID 2. Consequently, the conditional logic to label players was only met for objects with class ID 0 (i.e., 'ball').
- **Resolution**: The PLAYER_CLASS_ID variable in the reid_script.py was adjusted from 0 to 2. This simple but critical correction immediately resolved the labeling issue, allowing players to be correctly identified and tracked with unique IDs, and the ball to be labeled separately.

Another minor challenge involved Git warnings about line ending conversions within the virtual environment files. This was resolved by ensuring the .gitignore file correctly excluded the venv_reid_project/ directory from Git's tracking.

---

4. What Remains and Future Work

While the core re-identification task is successfully implemented, here are areas for potential future work and improvements:

- **Goalkeeper and Referee Tracking**: Currently, goalkeepers and referees are detected as distinct classes but are implicitly handled as "non-players" in the current visualization logic.

For a more comprehensive solution, these could be separately identified and tracked with their own consistent IDs and distinct labels (e.g., "GK: X", "REF: Y"). This would involve modifying the if/else logic for cls_id in the drawing loop.

- **Robustness in Extreme Occlusions**: While ByteTrack handles moderate occlusions well, extremely long or severe occlusions (e.g., multiple players completely blocking another for an extended period) can still lead to ID switches or new ID assignments. More advanced re-identification techniques (e.g., integrating appearance features like ReID embeddings) could further enhance robustness in such scenarios.
- **Performance Optimization**: For very long videos or real-time deployment on less powerful hardware, further optimization of the inference pipeline could be explored, such as using half-precision (FP16) models or ONNX runtime for faster inference.
- **Output Options**: Adding command-line arguments for input/output paths, confidence thresholds, or an option to disable the live display window would improve usability.
- **Evaluation Metrics**: For a production system, quantitative evaluation metrics (e.g., MOTA, MOTP, IDF1) would be used to objectively measure tracking performance.