Report on Player Tracking Script Implementation

Date and Time of Report: 09:15 PM IST, Friday, July 11, 2025

Script Overview:

The provided Python script implements a player tracking system using the YOLO (You Only Look Once) object detection model and the DeepSort tracking algorithm. The objective is to detect players in the video /2932301-uhd_4096_2160_24fps.mp4, assign unique IDs, and display these IDs with bounding boxes in the output video output_tracked_video_deepsort.mp4.

Approach and Methodology

Initialization and Setup:

- Libraries: The script utilizes cv2 (OpenCV) for video processing, numpy for numerical operations, ultralytics.YOLO for object detection, and deep_sort_realtime.deepsort_tracker.DeepSort for multi-object tracking.
- Model and Video Loading: The YOLO model is loaded from /best (2).pt, and the input video is accessed via cv2.VideoCapture with error handling to exit if loading fails.
- Output Configuration: A video writer is initialized with the MP4V codec, preserving the original frame rate, width, and height for the output video.

Detection and Tracking Pipeline:

- YOLO Detection: The script processes each frame with the YOLO model without a class filter (results = model(frame)), enabling detection of all classes. A confidence threshold of 0.3 is set to include more potential detections, and the class ID (cls) is extracted for debugging.
- DeepSort Tracking: Detected bounding boxes are fed into the DeepSort tracker, which assigns unique track_id values. Default parameters (max_age=30, nn_budget=100, embedder="mobilenet") are used for appearance-based re-identification.

- ID Assignment: During the first 3 seconds (initial frames), player_ids stores initial track IDs with their bounding box coordinates to establish baseline identities.

Visualization:

- Bounding Boxes: Green rectangles are drawn around detected objects using cv2.rectangle.
- ID Display: IDs are rendered as white text with a black background rectangle using cv2.putText, with a font scale of 2.0 and thickness of 2 for enhanced visibility. The text position adjusts dynamically to avoid frame edges.
- Real-Time Simulation: A time.sleep delay ensures processing aligns with the video?s frame rate.

Cleanup:

- Resources are released with cap.release(), out.release(), and cv2.destroyAllWindows(), though the latter may cause issues in headless environments.

Techniques Tried and Outcomes

Initial Detection Approach:

- Technique: Used the default YOLO model with no class filtering and a confidence threshold of 0.3.
- Outcome: Resulted in zero detections (e.g., Detections in frame 367: 0), likely due to the high resolution (4096x2160) exceeding the model?s training resolution or an incompatible model.

Debugging and Logging:

- Technique: Added debug prints to log detected classes and confidence scores.
- Outcome: Provided insight into the lack of detections, confirming the model was not identifying objects, which guided further adjustments.

Resolution Adjustment (Previous Iterations):

- Technique: In prior versions, frames were resized to 1280x720 to match typical YOLO training resolutions, with scaled coordinates for drawing.
- Outcome: Improved detection in some cases, but the current script reverts to the original resolution, potentially reintroducing the issue.

DeepSort Parameter Tuning (Previous Iterations):

- Technique: Adjusted max_age to 50 and nn_budget to 150 in earlier versions to enhance re-identification.
- Outcome: Improved tracking consistency when detections were present, though not applicable here due to zero detections.

Team-Based ID Assignment (Previous Iterations):

- Technique: Implemented color-based team classification (red vs. white) using HSV thresholding.
- Outcome: Successfully assigned team IDs (1 for red, 2 for white) when detections occurred, but this feature is absent in the current script.

Challenges Faced

Zero Detections:

- Challenge: The YOLO model consistently failed to detect objects, as evidenced by Detections in frame 367: 0. This is likely due to the UHD resolution (4096x2160) or an untrained/incompatible /best (2).pt model.
- Impact: Prevents ID assignment and display, rendering the tracking ineffective.
- Mitigation Attempt: Lowering the confidence threshold to 0.3 and removing class filters were tried, but the resolution mismatch or model issue persists.

GUI Error with cv2.destroyAllWindows:

- Challenge: The script terminates with an error (error: (-2:Unspecified error) The function is not implemented) due to a lack of GUI backend support in the OpenCV build.

- Impact: Cosmetic only, as no windows are displayed, but it interrupts clean termination.

- Mitigation Attempt: Previous reports recommended removing this line, which should be applied here.

Re-identification and Ball Detection:

- Challenge: Earlier versions struggled with re-identifying the same person and assigned IDs to the ball. The current script lacks the team-based logic and ball filtering.

- Impact: Inconsistent IDs and unwanted tracking of non-player objects when detections occur.

- Mitigation Attempt: Prior adjustments (e.g., if cls == 0, color-based team IDs) were effective but are not included here.

Performance:

- Challenge: Processing a high-resolution video in real-time without resizing may strain resources, especially without GPU acceleration.

- Impact: Potential delays or crashes, though not reported yet.

- Mitigation Attempt: Resizing was tried previously but reverted in this version.

Recommendations

- Resolve Detections: Reintroduce frame resizing to 1280x720 and scale coordinates back, or switch to a pre-trained model like yolov8n.pt if /best (2).pt fails.

- Fix GUI Error: Remove cv2.destroyAllWindows() since no windows are used.

- Enhance Tracking: Add team-based IDs and ball filtering (e.g., if cls == 0) from prior versions.

- Testing: Run the script, check debug output for detections, and inspect the output video.