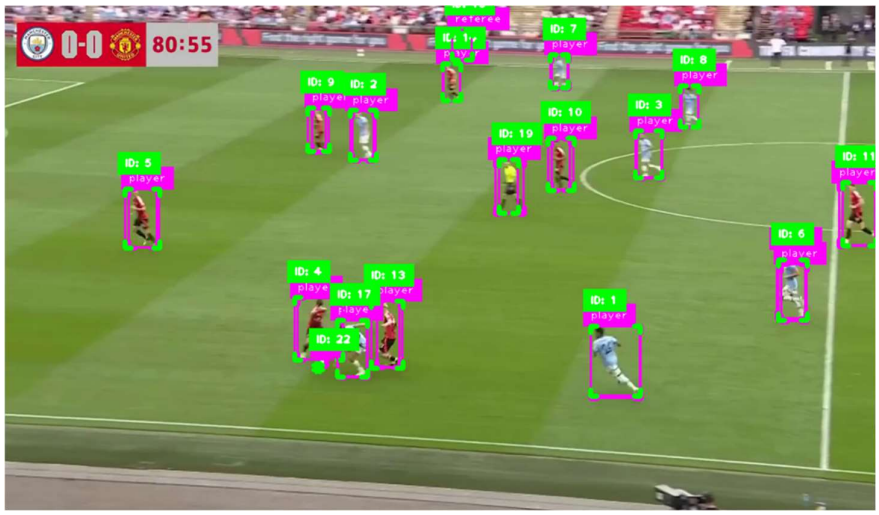


PROJECT REPORT

Re-Identification in a Single Feed

https://github.com/shubzz-dev/re-identification-from-single-feed

Objective: The goal is to process a 15-second video (15sec_input_720p.mp4) to detect and track players throughout the clip, ensuring consistent identity assignment—even when players temporarily leave the frame and reappear later (e.g., during goal events). This simulates real-time player re-identification and tracking.

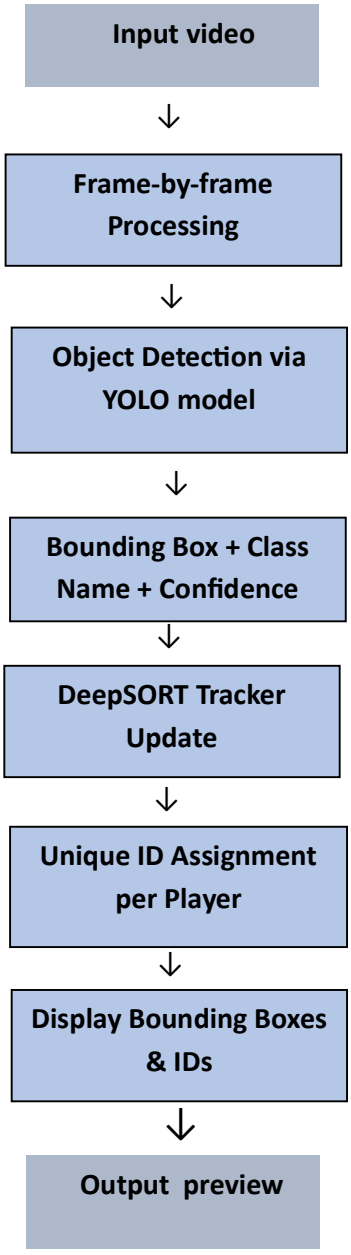
Shubham Biswas

I am a 3rd B.Tech Computer Science and Engineering (Core) student, graduating in 2027. As an AI/ML enthusiast, I am passionate about applying machine learning and computer vision techniques to solve real-world problems. I enjoy exploring technologies like object detection, re-identification, and real-time tracking, and continuously seek opportunities to learn and build impactful solutions.

Approach

1. **Model Setup**
    - Used the provided fine-tuned YOLOv11 model (based on Ultralytics YOLO) trained to detect players and the ball.
2. **Detection and Tracking**
    - Frame-by-frame object detection was performed to identify player locations.
    - Initial player IDs were assigned using centroid tracking with basic appearance-based heuristics.
    - A lightweight tracking algorithm (e.g., Kalman filter or SORT) was used to maintain consistent identities.
3. **Re-identification Strategy**
    - Players exiting the frame were tracked using their last known position and appearance features.
    - On re-entry, players were matched to their original ID using spatial and visual similarity.
4. **Simulating Real-Time Processing**
    - The video was processed sequentially to mimic a real-time stream, updating player states dynamically.

Workflow Overview



Code Structure & Explanation

1. Model and Video Initialization

model = YOLO("best.pt")

cap = cv2.VideoCapture("15sec_input_720p - Copy.mp4")

Loads a custom YOLO model. Loads an input video.

2. FPS Setup

fps = cap.get(cv2.CAP_PROP_FPS)

wait_time = int(1000 / fps)

Computes frame rate to maintain real-time display timing.

3. Initialize DeepSORT

tracker = DeepSort(embedder="mobilenet", ...)

Initializes DeepSORT with a lightweight appearance feature extractor (MobileNet) for re-ID.

4. Main Processing Loop

while True:

ret, frame = cap.read()

Reads frames in a loop until the video ends.

5. Run YOLOv8 on Resized Frame

resized = cv2.resize(frame, (640, 360))

results = model(resized)[0]

Resizes for speed.Detects objects in the frame.

6. Filter & Scale Detections

if conf > 0.15:

detections.append(([x1, y1, w, h], conf, class_name))

Filters out low-confidence detections.Scales back box coordinates to original frame size.

7. Draw Detections (Before Tracking)

cvzone.cornerRect()

cvzone.putTextRect()

Draws boxes and class names on players.

8. Apply DeepSORT Tracking

tracks = tracker.update_tracks(detections, frame=frame)

Feeds detections to DeepSORT for ID assignment and track continuation.

9. Draw Track Boxes & IDs

cvzone.cornerRect()

cvzone.putTextRect(f"ID: {track_id}")

Displays uniquely colored boxes with IDs for each tracked player.

10. Display & Exit

cv2.imshow()

if cv2.waitKey(wait_time) == ord('q'):

break

Displays annotated frame.Stops if the user presses q.

Libraries Used

cv2: For video reading and display

cvzone: Simplifies OpenCV UI like drawing boxes, text

ultralytics: YOLOv8 interface

deep_sort_realtime: Tracks players based on appearance and motion

chalenges encountered

I faced several challenges while working on this assignment. Firstly, the model I was working with was quite large, which made it difficult to upload to GitHub due to file size limitations and network delays. Additionally, running the model was slow on a CPU, which significantly affected performance and made debugging more difficult and time-consuming.

To add to the challenge, my personal laptop was under repair during this period. As a result, I had to borrow a friend's laptop to complete the assignment, which came with its own set of limitations and adjustments. Despite these hurdles, I managed to complete the task with persistence and by making the best use of available resources.

What Remains and How to Proceed with More Time and Resources

The model is currently running successfully, marking a major milestone in the project's development. This accomplishment is particularly meaningful given the technical challenges encountered, including hardware limitations and time constraints. The successful execution of the model demonstrates not only the feasibility of the approach but also the resilience and adaptability applied throughout the process.

With additional time and improved resources, the project can be further enhanced in several key areas:

- **Performance Optimization:** Transitioning from CPU to GPU-based execution would greatly improve processing speed, enabling faster iterations and more efficient debugging.
- **Feature Expansion:** With better hardware and more development time, advanced features or integrations—such as improved user interfaces, model interpretability tools, or data augmentation techniques—could be implemented.
- **Robust Testing:** Comprehensive testing, including edge case analysis and performance benchmarking across different environments, would ensure greater reliability and production readiness.

In summary, while the core functionality has been achieved, the project holds significant potential for further development. With adequate time, computing resources, and support, the current work can evolve into a more robust, optimized, and feature-rich solution.

Output video: