Soccer Player Re-Identification Assignment Report

1.This project focuses on solving a real-world computer vision problem: re-identifying soccer players across frames in a single camera feed. The goal is to maintain consistent player IDs throughout a 15-second clip, even when players exit and re-enter the frame or overlap.

To accomplish this, I used:

- **YOLOv11** for player detection
- **DeepSORT** for consistent player tracking (re-identification)
- **OpenCV** to read/write video frames and overlay bounding boxes and labels

2. Approach and Methodology

Step-by-step process:

1. **Input Preparation**
   Placed the video file 15sec_input_720p.mp4 in the project's root directory.
2. **Model Setup**
   The provided YOLOv11 model (yolov11_soccer.pt) was missing, so I used a placeholder detection approach with a sample YOLO model from Ultralytics for demonstration and testing.
3. **Detection**
   I used a YOLO model to detect soccer players in each frame.
4. **Tracking with DeepSORT**
   Player detections were passed to DeepSORT, which assigned and preserved unique IDs across frames using a Kalman Filter and cosine appearance metrics.
5. **Visualization**
   Bounding boxes and player IDs were drawn on each frame, and the output video was saved for evaluation.

3. Techniques Used

- **YOLOv11 (placeholder YOLOv8 used)**
  Object detector to localize players in the video frame by frame.
- **DeepSORT**
  Tracker to associate detections over time and assign consistent IDs.
- **OpenCV**
  Used for video frame extraction, drawing, and writing the final output video.
- **Python Virtual Environment**
  Isolated dependencies and ensured reproducibility.

4. Setup Instructions

Environment Setup:

bash

Copy Edit

Step 1: Create a virtual environment

python -m venv venv

venv\Scripts\activate #

Windows

Step 2: Install dependencies

pip install -r requirements.txt

Manually install tracking library if missing

pip install deep_sort_realtime

File Structure:

bash

CopyEdit

soccer_reid/

│

```
├── 15sec_input_720p.mp4        # Input video
├── models/
│   └── yolov11_soccer.pt       # (Missing model, replace with placeholder if needed)
├── track_players.py            # Detection + Tracking code
├── output.mp4                  # Tracked video output
```

Running the Script:

```
bash
CopyEdit
python track_players.py
```

5. Sample Code Snippet

```
python
CopyEdit
from ultralytics import YOLO
from deep_sort_realtime.deepsort_tracker import DeepSort
import cv2

# Load model and tracker
model = YOLO('models/yolov11_soccer.pt')
tracker = DeepSort(max_age=30)

# Open video
cap = cv2.VideoCapture('15sec_input_720p.mp4')
while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break
```

```
    # Detect players
    results = model(frame)
    detections = []
    for det in results[0].boxes.data:
        x1, y1, x2, y2, conf, cls = det
        detections.append(([x1, y1, x2 - x1, y2 - y1], conf, 'player'))

    # Track players
    tracks = tracker.update_tracks(detections, frame=frame)
    for track in tracks:
        if not track.is_confirmed():
            continue
        track_id = track.track_id
        l, t, w, h = track.to_ltrb()
        cv2.rectangle(frame, (int(l), int(t)), (int(l+w), int(t+h)), (0,255,0), 2)
        cv2.putText(frame, f'ID {track_id}', (int(l), int(t)-10), cv2.FONT_HERSHEY_SIMPLEX,
0.5, (255,255,255), 2)

    # Save or display frame
    # cv2.imshow('Tracked', frame)

cap.release()
cv2.destroyAllWindows()
```

6. Challenges Encountered

- **Model Missing**
  The actual yolov11_soccer.pt model file was not provided in the assignment document. I used a pretrained YOLOv8 model for demonstration.
- **ID Switching**
  When players were close together, DeepSORT occasionally switched IDs briefly, especially during overlaps or occlusions.
- **Dependency Issues**
  deep_sort_realtime was not included in requirements.txt, and had to be installed manually.
- **Frame Rate Drops**
  Real-time performance slowed down on a CPU-only setup, especially when working with high-resolution input video.

7. What Remains / How I Would Proceed With More Time

If I had more time and compute resources:

- I would fine-tune YOLOv11 specifically on a soccer dataset to better detect players.
- I'd train a custom Re-ID embedding network for DeepSORT to better handle reappearances.
- Integrate jersey number recognition or color-based features to aid re-identification.
- Add support for player action detection (goals, passes, fouls) using sequence modeling.
- Improve real-time efficiency with model quantization and tracking optimizations.
- Package the entire pipeline into a single GUI-based tool for non-technical users.

8. Summary and Conclusion

This project taught me the core mechanics of combining object detection with multi-object tracking. I understood how YOLO and DeepSORT complement each other in creating a consistent and real-time tracking solution. It also gave me experience debugging real-world setups including handling missing assets, managing Python environments, and thinking ahead in terms of performance bottlenecks and scalability.