Player Identification and Tracking from a Single Camera Feed

📁 Project Description

This project implements a complete computer vision pipeline to detect and track players in a sports video captured from a single static camera. The main objective was to utilize a pretrained YOLOv5 model (`best.pt`) — provided as part of the assignment — to generate player detection bounding boxes, assign consistent tracking IDs, and produce both an annotated video and structured tracking output in CSV format.

🗂️ Project Structure

```
player_reid_single_feed/
├── .gitignore
├── README.md
├── detector.py              # Handles player detection using YOLOv5
├── main.py                  # Main script to run detection + tracking
+ save output
├── requirements.txt         # Python dependencies
├── tracker.py               # Implements simple IOU-based tracking
├── models/
│   └── best.pt              # Provided pretrained YOLOv5 model
├── outputs/
│   ├── output.mp4           # Final annotated video
│   └── tracking_data.csv    # CSV file with frame-wise tracking info
├── videos/
│   └── 15sec_input_720p.mp4 # Input test video
├── venv/                    # Python virtual environment (optional)
└── __pycache__/             # Compiled Python files
```

⚙️ Tools & Technologies Used

- **Python 3.9+**
- **OpenCV** – for video handling and annotation
- **YOLOv5** (PyTorch) – object detection using pretrained model
- **IOU-based Tracking Algorithm** – simple and effective for short sequences
- **CSV** – for exporting tracking metadata in structured format

---

🧠 Approach & Methodology

1. Frame Extraction

- Video is read frame-by-frame using OpenCV's `cv2.VideoCapture`.

2. Player Detection

- Each frame is passed to the `best.pt` model (YOLOv5) for inference.
- Only detections labeled as **class 0** (presumably players) are retained.
- Bounding boxes and confidence scores are extracted.

3. Tracking with IOU Matching

- A basic Intersection Over Union (IOU) algorithm was implemented to maintain identity across frames.
- Existing detections are matched with previous ones using IOU thresholding.
- New IDs are assigned to unmatched detections.

4. Output Generation

- Each frame is annotated with bounding boxes and `track_id` labels.
- Annotated frames are stitched together using OpenCV to produce the output video.

A CSV file `tracking_data.csv` is generated with the format:

CopyEdit

```
frame,track_id,x1,y1,x2,y2
```

-

5. Pretrained Model Constraint

- No training or fine-tuning was performed on `best.pt` due to assignment constraints.

🧪 Techniques Tried & Outcomes

Technique	Outcome
Confidence Threshold Tuning	Lowered YOLOv5 threshold to `0.15` to improve detection of distant or partially visible players.
Contrast Enhancement	CLAHE (Contrast Limited Adaptive Histogram Equalization) was tested to boost visibility in dim frames.
Advanced Trackers (Explored, Not Used)	Explored potential use of DeepSORT/ByteTrack for better tracking, but kept scope limited to IOU.
Result Analysis	While some players were tracked consistently across frames, others were missed, especially under occlusion or low confidence.

🧱 Challenges Faced

- **Detection Gaps:**
  Despite multiple players on screen, the model often detected only 3–4 consistently.

- **Fixed Model:**
  Could not retrain or improve the model due to restrictions on modifying `best.pt`.

- **Tracking Drift:**

  - Players crossing paths caused **ID switches**.

  - **Occlusions** or players temporarily leaving the frame caused identity loss.

- **Video Constraints:**
  Fast movement, resolution changes, and background noise affected IOU-based tracking.

✅ Results

- **Output Video:**
  → `outputs/output.mp4` – Annotated with bounding boxes and tracking IDs.

Tracking CSV:
→ `outputs/tracking_data.csv` – Frame-wise tracking results with format:

CopyEdit

```
frame,track_id,x1,y1,x2,y2
```

-

Sample row:

```
29,3,645,484,1310,987
```

- Tracking was successful for some players over hundreds of frames but incomplete for others due to the limitations discussed.

📝 Conclusion

The project accomplished its core objectives:

- Applied pretrained YOLOv5 model to detect players in a sports video.
- Built a simple yet functional object tracker using IOU matching.
- Successfully generated an annotated video and structured CSV tracking output.

🚀 Suggestions for Future Work

- **Advanced Tracking:** Integrate DeepSORT, ByteTrack, or Kalman Filter for robust identity preservation.
- **Model Fine-tuning:** If permitted, retrain or fine-tune YOLOv5 on custom sports dataset.
- **Occlusion Handling:** Implement re-identification features or embeddings for more reliable tracking.
- **Post-processing:** Use trajectory smoothing or temporal consistency checks to reduce ID switching and jitter.