**Beach Volleyball Pose Estimation / Action Detection Project Summary**

**1. Project Goals**

* Develop a robust pipeline for player detection, player pose estimation, and (in the future) action recognition (e.g., spikes) from video recordings of beach volleyball training.
* Main focus for now is on player detection and pose estimation (performance, precision, reliability). Action recognition / event detection will come later.
* Leverage human pose estimation to enhance action understanding and potentially automate annotation of long video datasets (~2 hours videos).

**2. Context**

* The project focuses on training sessions involving elite beach volleyball athletes.
* Input video data are a 1920x1080, 50fps, h264-encoded video data. Video is filmed from the back of the court, camera is positioned approximately 5m behind the back line and 4m high.

**3. Technology Stack and Pipeline (So Far)**

**Detection:**

* **Model:** RTMDet (from OpenMMLab)
* **Format:** ONNX, running on ONNX Runtime with CUDA backend
* **Frame processing:** One frame at a time, full-frame detection

**Tracking:**

* **Tracker:** ByteTrack
* **Tracking IDs:** Assigned per detection, passed to pose estimator
* Tracking time is negligible compared to detection and pose estimation.

**Pose Estimation:**

* **Model:** RTMPose (from OpenMMLab)
* **Format:** ONNX running on ONNX Runtime with CUDA backend
* **Pipeline:** Crops are extracted per tracked bounding box, then pose is estimated

**Inference Backend:**

* ONNX Runtime (CUDA)

**Output:**

* Pose, detection, and ID data saved to HDF5
* Video output optionally generated with overlayed keypoints and IDs

**Hardware:**

* Home PC: RTX 5070 (formerly 1070 Ti)
* Lab PC: RTX 4060 (target machine for performance optimization)

**4. Current status**

We have a full functioning pipeline (script MAIN.py):

1. read video frames using OpenCV
2. perform player detection using RTMdet
3. track bboxes using ByteTrack
4. perform pose estimation on tracked bboxes using RTMpose
5. generate video overlaid with bboxes and poses, with some performance metrics (video capture, detection, tracking, pose estimation, display, save results times)

We are satisfied for now with the precision of the detection, tracking and pose, and are more focused on performance (inference time).

Initial tests indicate the following:

ONNX models running on CUDA (RTX 5070), medium size models (RTMdet-M, RTMpose-M):

1. frame capture time: approx. 5ms
2. detection time: approx. 40 ms
3. tracking time: approx.. 1ms
4. pose estimation time: approx. 40ms

The above is for 5 individuals tracked and posed per frame (4 players + coach). Surprisingly performance is close (+/- 20%) on GPUs with quite different compute power (1070Ti, 4060, 5070). This suggests some bottleneck in either on overhead, preprocessing, postprocessing, or memory transfers.

When running inference on GPU the following warnings appear:

“[W:onnxruntime:, session\_state.cc:1168 onnxruntime::VerifyEachNodeIsAssignedToAnEp] Some nodes were not assigned to the preferred execution providers which may or may not have an negative impact on performance. e.g. ORT explicitly assigns shape related ops to CPU to improve perf.”

It is possible some ops being given to CPU cause too many memory transfers which create the bottleneck. Although I am very unsure it is the case, just a potential lead to investigate.

Same parameters but running on CPU (ONNX runtime backend as well):

1. frame capture time: approx. 5ms
2. detection time: approx. 250 ms
3. tracking time: approx.. 1ms (this runs on GPU separately)
4. pose estimation time: approx. 100ms

**5. Next Steps**

For now we are focused on improving speed (inference time).

**6. Miscellaneous**

We are working in Visual Studio Code, in a conda environment called “HPE\_volleyball”.

We have a github public repo called “HPE\_volleyball”.

We are using RTMpose and RTMdet through RTMlib (installed as a python package).

Ignore scripts “test\_unifi\_stream\_read.py” and “convert\_onnx\_to\_FP16.py” in /scripts, they were test scripts not relevant to current situation.

Ignore “streaming delay optimisations 2025 04 01.md” in “/misc project docs”, it pertains to a different project.