Human-Following Ackermann v2 — Refactor to OAK-D Pro W

Author: Senithu Dampegama — Robotics & AI Engineer
Date: 23 September 2025
Version: v2 (successor to the Arducam/PiNSight-based v1)

Contents

0) Executive Summary

- **Objective:** Upgrade the prior human-following Ackermann robot (v1) to reliably track **one person among many** while reducing Jetson load.

- **Approach:** Offload perception to **OAK-D Pro W** (Myriad X) running a **Spatial MobileNet-SSD**; add a **lightweight IOU DeepSORT** tracker; retain the proven **L5/L4 layered control** with **Teensy 4.1** actuation.

- **Safety & Robustness:** Stationary **SEARCH/RECOVER** (no motion), **serial auto-reconnect** on USB drops, **bounded runtime** flag for tests, capped PWM, and scalable dual preview (RGB + Depth).

- **Results:** Stable person lock with low Jetson usage; tighter right turns via **asymmetric steering gains** and **non-linear mapping**; unchanged ASCII protocol to Teensy; clean STUB→LIVE runbooks.

- **Deliverables:** A reorganized package `hf_ackermann/` ready for reuse, plus this report documenting architecture, parameters, tests, and future work.

1) Background & Motivation

v1 summary. The previous build transformed a failed RC toy into a research robot: Jetson Orin Nano for perception/decision (L5), Teensy 4.1 for actuation/safety (L3), and a simple ASCII protocol (L4). It followed a single person reliably but **could not lock onto one human** if multiple people entered the frame.

v2 goals. 1. **Resolve multi-person ambiguity** via per-person IDs.
2. **Reduce Jetson load** by moving heavy CV to the vision sensor.
3. Preserve the **safe, layered** control split and the exact L4 ASCII protocol to avoid firmware churn.

Key change: Replace Arducam/PiNSight with **OAK-D Pro W**; implement a lightweight, IOU-only DeepSORT to keep stable `track_id` while minimizing compute.

2) System Architecture (L5→L1)

```
OAK-D Pro W  ── RGB + StereoDepth + Spatial NN (on-device)
   │ detections {bbox, conf, z_mm}
   ▼
Tracking (IOU DeepSORT) → selected target {track_id, offx, area, z_m, quality}
   ▼
L5 (Jetson) Decision FSM  INIT → SEARCH → ACQUIRE → FOLLOW → FAULT
   │ emits high-level actions (ARC/DRIVE/STOP/CENTER)
   ▼
L4 (Jetson) ASCII Serial Client → Teensy 4.1 (safety, PWM bursts, stall guard)
   ▼
Drivers & Actuators (BTS7960 drive, BTS7980 steering)
```

What's new in v2 - On-device **Spatial NN** (x,y,z) from OAK-D; Jetson only does tracking + L5.
- **Dual preview**: RGB with overlays + depth false-color (scalable).
- **Stationary SEARCH/RECOVER** (no peeking) for safety and clarity.
- **Asymmetric steering** (right boosted) and **gamma-shaped turn mapping**.
- **Serial auto-reconnect** & **bounded runtime** for robust testing.

3) Hardware Overview

- **Compute:** NVIDIA Jetson (Orin Nano or better).
- **Camera: OAK-D Pro W** (wide FOV), Myriad X.
- **Controller:** Teensy 4.1 (Cortex-M7, 600 MHz).
- **Drivers:** BTS7960 (drive axles), BTS7980 (steering).
- **Power:** Dual 3S LiPo, separate regulators; common ground; shielded USB to Teensy.

Note: Keep OAK-D and high-current returns cleanly separated; add ferrites to USB if needed.

4) Software Architecture

4.1 Vision (OAK-D On-Device)

- **Pipeline:** ColorCamera 1080p + StereoDepth (aligned to RGB) → **MobileNetSpatialDetectionNetwork @ 300×300**.

- **Outputs per detection:** `bbox_px`, `confidence`, `label`, `spatialCoordinates.{x,y,z}` in mm.
- **VisionBus payload to L5:** `{track_id, offx∈[-1..1], area∈[0..1], quality, z_m (m), ts}`.
- **Preview (when enabled):** Composite window **[RGB annotated | Depth false-color]**, scalable via `--show-scale`.
- **Hotkeys:** `l` lock nearest, `r` release, `q/ESC` quit (optional `s` STOP, `c` CENTER if wired).

4.2 Tracking (Lightweight DeepSORT)

- IOU-only association with `max_age`, `iou_threshold`, `hit_streak` → **sticky IDs** at minimal compute cost.

- Optional heavy wrapper (deep-sort-realtime) available but off by default on Jetson.

4.3 Decision (L5) — FSM

States: `INIT → SEARCH → ACQUIRE → FOLLOW → FAULT` - **SEARCH / RECOVER: stationary**; on entry: `STOP + CENTER` once; wait for a valid target.
- **ACQUIRE:** quick steering impulses to bring target toward center; short forward nudges if far.
- **FOLLOW:** - **Steering:** non-linear mapping from `|offx|` with exponent `arc_gamma` and a **minimum turn PWM**; **asymmetric left/right gains & caps** (right boosted).
- **Distance:** prefer **depth `z_m`** (target-z), else **area** (target-area ± tol).
- **FAULT:** triggered by L4 disconnects; L5 stops issuing motion, then attempts **auto-reconnect** with backoff; on success: `STOP + CENTER` and resume `SEARCH`.

4.4 Interface (L4) — ASCII Protocol (unchanged)

- **Commands:** `STOP`, `CENTER`, `DRIVE F|B <pwm> <ms>`, `ARC L|R <pwm> <ms> <U|H>`.

- **Replies:** `OK`, `DONE <ARC|DRIVE>`, `ERR <…>`, `TELEM drive=<…> steer=<…> L=<…> R=<…> t=<…>`.

- **Timing:** 115200 baud; read timeout ≈ 0.05 s; cmd timeout ≈ 2 s; DONE timeout ≈ 3 s (or by duration).

- **Robustness:** auto-detect Teensy; **auto-reconnect** on SerialException; resume in SEARCH.

5) Repository Layout (v2)

```
hf_ackermann/
  app.py                          # CLI entry: STUB/LIVE, vision source,
tuning flags
  control/
    rc_layer4.py                  # L4 serial client (ASCII, OK/DONE/TELEM,
reconnect)
    rc_layer5.py                  # L5 FSM (stationary SEARCH), steering
shaping, distance logic
  vision/
    vision_oakd_onboard.py        # OAK-D spatial NN + depth + composite
preview
    vision_pinsight_legacy.py     # Legacy fallback (non-spatial)
  tracking/
    deepsort_iou.py               # Lightweight IOU tracker (default)
    deepsort_heavy.py             # Optional wrapper for deep-sort-realtime
  tests/
    test_l4_smoke.py              # OK/DONE/TELEM smoke
    test_l4_fault.py              # L4 disconnect → auto-reconnect (mocked)
scripts/
  run_stub_oakd.sh                # safe preview, no motion
  run_live_oakd.sh                # hardware-in-loop (wheels-up first!)
requirements.txt, README.md, LICENSE
```

6) Interfaces & Protocols

6.1 Command Protocol (Jetson → Teensy)

```
STOP\n
CENTER\n
DRIVE F 190 240\n
DRIVE B 140 150\n
ARC L 220 180 U\n
ARC R 220 180 U\n
```

6.2 Telemetry Protocol (Teensy → Jetson)

```
OK\n
DONE DRIVE\n
TELEM drive=1 steer=2 L=497 R=1022 t=25090824\n
```

`drive`/`steer` are state codes; `L`/`R` are sensor counts; `t` is uptime ms.

7) Control Parameters (defaults)

7.1 Steering & Turn Shaping

Parameter	Default	Purpose
`deadband_x`	0.16	Ignore small offsets to avoid chatter
`arc_gamma`	1.6	Non-linear turn shaping for large errors
`pwm_turn_min`	140	Enforce decisive minimum turn
`k_arc_pwm_left`	700	Left turn gain
`k_arc_pwm_right`	820	**Right turn boosted**
`arc_pwm_cap_left`	250	Max left arc PWM
`arc_pwm_cap_right`	300	**Max right arc PWM**
`arc_impulse_ms_left`	180 ms	Left arc duration
`arc_impulse_ms_right`	200 ms	**Right arc duration**
`steer_bias`	0.00	Optional static offset (normalize drift)

7.2 Distance & Motion

Parameter	Default	Purpose
`target_z_m`	(off)	Preferred distance (m) if depth available
`target_area`	0.18	Fallback distance via bbox area
`area_tol`	0.04	Area band around target
`k_drive_pwm`	2200	Forward PWM gain (capped)
`max_drive_pwm`	190	Safety cap for drive
`drive_impulse_ms`	240 ms	Forward impulse duration
`back_pwm`	140	Reverse PWM for back-off
`back_impulse_ms`	150 ms	Reverse impulse duration

7.3 Robustness

Parameter	Default	Notes
`search_mode`	`idle`	SEARCH/RECOVER are stationary
`--show-scale`	0.5–0.6	Smaller preview window
`--max-seconds`	(off)	Bounded runtime for tests

All are exposed via CLI flags in `app.py` for runtime tuning.

8) Setup & Runbook

8.1 Environment

```
python3 -m venv .venv && source .venv/bin/activate
pip install -r requirements.txt
```

OAK-D probe

```
python - <<'PY'
import depthai as dai
with dai.Device() as d:
    print('OAK-D connected:', d.getDeviceName(), d.getMxId())
PY
```

8.2 STUB (safe, no motion)

```
./scripts/run_stub_oakd.sh
# or
python -m hf_ackermann.app --mode STUB --vision OAKD --show 1 --max-seconds 8
```

8.3 LIVE (hardware-in-loop)

Safety: first runs **wheels-up** on a stand; ensure serial permissions (`dialout`).

```
./scripts/run_live_oakd.sh
# or
python -m hf_ackermann.app --mode LIVE --vision OAKD --show 1 --max-seconds
10
```

If auto-detect fails: `--port /dev/ttyACM0` (or a udev symlink `/dev/teensy`).

9) Validation & Observations

- **Stationary SEARCH:** verified — no ARC/DRIVE while waiting; transitions to ACQUIRE on fresh target.
- **FOLLOW:** decisive turns, especially to the right (boosted gains/caps); depth-preferred distance with smooth back-off when close.
- **Serial robustness:** induced disconnects yield `[FAULT]` → **auto-reconnect** → `STOP + CENTER` → back to SEARCH without crash.
- **Multi-person:** lock persists via `track_id`; `l/r` hotkeys control target selection.

Representative telemetry

```
TELEM drive=1 steer=2 L=497 R=1022 t=25090824
[FOLLOW] steer arcright pwm=220
```

```
TELEM drive=1 steer=2 L=526 R=1021 t=25091024
[FOLLOW] stale → center once
...
```

10) Risk & Limitations

- No wheel encoders → open-loop velocity; rely on depth/area and impulses.
- Donor wiring caps drive PWM to 190 for thermal safety; revisit after loom upgrade.
- IOU-only tracker can ID-swap under heavy, long occlusions (lock + stationary SEARCH mitigates most cases).
- Depth noise possible in strong sun/low-texture scenes; consider median filtering or depth caps.

11) Tuning Guide

- **Right turns weak?** Increase `--k-arc-right`, `--cap-right`, `--impulse-right`; optionally raise `--pwm-turn-min` and `--arc-gamma`.
- **Over-steer/oscillation?** Increase `--deadband-x` by +0.02 or lower `--arc-gamma` (1.4–1.6).
- **Far off-center?** Increase `--arc-gamma` (e.g., 1.8) and/or `--pwm-turn-min`.
- **Depth-first following:** set `--target-z 1.2` (example), otherwise use `--target-area`/`--area-tol`.

12) Future Work

- Appearance embeddings (heavy DeepSORT) when compute budget allows.
- Add encoders + basic odometry; consider SLAM for navigation.
- Wiring/driver upgrades to lift PWM caps safely.
- `systemd` auto-start (STUB by default), simple web telemetry, and a udev rule for `/dev/teensy`.

Appendix A — CLI Flags (selected)

```
--mode {STUB,LIVE}
--vision {OAKD,FAKE,PINSIGHT}
--port /dev/ttyACM0
--show {0,1}          --show-scale 0.5
```

```
--max-seconds 10

# Steering & distance
--deadband-x 0.16
--arc-gamma 1.6
--pwm-turn-min 140
--k-arc-left 700 --k-arc-right 820
--cap-left 250 --cap-right 300
--impulse-left 180 --impulse-right 200
--steer-bias 0.00
--target-z 1.2
--target-area 0.18 --area-tol 0.04
```

Appendix B — Serial Protocol Examples

```
> ARC R 220 180 U
< OK
< DONE ARC

> DRIVE F 190 240
< OK
< DONE DRIVE

< TELEM drive=1 steer=2 L=615 R=1022 t=25091724
```

Appendix C — Safety Checklist

- Start in **STUB**; only then run **LIVE** (wheels-up).
- Use `--max-seconds` during tests to avoid endless runs.
- Confirm TELEM rate (~5 Hz), caps (`max_drive_pwm`, `arc_pwm_cap_*`).
- Verify **auto-reconnect** path by briefly replugging USB (observe FAULT→recover).
- Keep grounds clean; use short, shielded USB.