

A Precision Audit of Adduct, a Static Analyzer for Robot Demonstration Data

46 Detectors Measured Against 20 Public LeRobot Datasets

Prabhu Gopal

26 August 2026 · adduct 0.2.0 · measured at commit c8265c0

Abstract

Static analyzers for robot demonstration data promise to catch defects that degrade a behavior-cloned policy before training begins. Their detectors are typically validated by fault injection: plant a known defect in a fixture, confirm the detector fires. That measures recall and cannot measure precision, because a detector that fires unconditionally passes every such test. We report the complementary measurement for *adduct*, an open-source linter for robot demonstration datasets, across 20 public LeRobot datasets (4,342 episodes, 545,964 frames, 2–40 action dimensions, 5–50 Hz, simulated and real). Our most consequential result is not about the tool: **18 of 20 datasets declare robot_type: "unknown"**, so any method conditioning on embodiment via Hub metadata degenerates to one undifferentiated bucket on 90% of this corpus. That includes adduct’s own conformal gate, whose Mondrian taxonomy is keyed on that field. On the tool: all 20 datasets parsed without error in 22.5s on a laptop CPU, yielding 190 findings (23 HIGH, 121 MEDIUM, 46 LOW). Twenty-seven detectors fired at least once and nineteen never fired, but six fire on 70% or more of this curated corpus, and a median of 9.5 findings per dataset with no dataset ever clean is a usability defect we measure in our own output. We tested one hypothesis for the worst offender, `dynamics.inverse_residual` (100% fire rate, HIGH on 50%): an ill-conditioned ridge solve. Fixing it eliminated the numerical fault but left the fire and HIGH rates unchanged, falsifying the hypothesis. The two surviving explanations, control rate and format-conversion provenance, are perfectly confounded in this corpus. We state plainly what this does not establish: without ground-truth labels a fire rate is a rate of complaint, not of correctness, and no policy was trained, so the premise that these defects degrade policies remains untested here.

1 Motivation

A data-quality detector encodes a claim: that a specific pattern in the data predicts a specific training failure. Testing that claim by fault injection establishes recall and nothing else. A detector hard-coded to return HIGH passes every such test.

Precision determines whether such a tool is usable in practice. A HIGH severity that fires on most curated public datasets carries no information, and it breaks a `--fail-on HIGH` continuous-integration gate for every user. Measuring precision requires real data and a count of complaints. This report is that measurement.

2 Corpus and Method

Twenty LeRobot datasets were selected for breadth of embodiment and provenance rather than size. Fourteen are Open X-Embodiment datasets [1] that reached LeRobot through a conversion from RLDS, the Reinforcement Learning Datasets format; six were published natively for LeRobot. Four are simulated. Action dimensionality spans 2 to 40 and control rates span 5 Hz to 50 Hz. Adduct does not decode video, so the corpus occupies roughly 50 MB.

Embodiments were confirmed from published sources where possible: `asu_table_top` is a UR5 commanding joint velocities, `austin_buds_dataset` and `nyu_franka_play_dataset` are Franka arms [2], `cmu_stretch` is a Hello Robot Stretch [3], `d1r_edan_shared_control` is a wheelchair-mounted 8-DoF DLR arm [4], and `utokyo_saytap` is a Unitree Go1 quadruped [5], the one locomotion dataset in an otherwise manipulation-heavy corpus. Platforms not verified against a primary source are not claimed.

Table 1: Corpus summary. All twenty datasets resolved as `lerobot_v3`. OXE denotes an Open X-Embodiment dataset reaching LeRobot via RLDS conversion. H/M/L are counts of HIGH/MEDIUM/LOW findings from the full pass.

Dataset	Source	Hz	act	state	Episodes	Frames	Scan (s)	H	M/L
<code>ucsd_pick_and_place</code>	real, OXE	5	4	7	1,355	67,750	4.2	2	7/3
<code>xarm_lift_medium</code>	sim	15	4	4	800	20,000	1.6	1	6/2
<code>nyu_franka_play</code>	real, OXE	5	15	13	456	44,875	2.4	1	8/3
<code>utokyo_pr2_tabletop</code>	real, OXE	5	8	7	240	32,708	1.1	2	10/3
<code>pusht</code>	sim	10	2	2	206	25,650	0.7	0	3/4
<code>pusht_keypoints</code>	sim	10	2	2	206	25,650	0.9	0	3/4
<code>ucsd_kitchen</code>	real, OXE	5	8	21	150	3,970	0.6	1	7/2
<code>cmu_stretch</code>	real, OXE	5	8	4	135	25,016	0.8	3	8/1
<code>asu_table_top</code>	real, OXE	5	7	7	110	26,113	0.8	1	6/1
<code>dlr_sara_grid_clamp</code>	real, OXE	5	7	12	107	7,622	0.6	1	4/4
<code>dlr_edan_shared_control</code>	real, OXE	5	7	7	104	8,928	0.6	1	9/2
<code>dlr_sara_pour</code>	real, OXE	5	7	6	100	12,971	0.7	1	8/4
<code>aloha_mobile_cabinet</code>	real	50	14	14	85	127,500	2.1	1	5/1
<code>utokyo_pr2_fridge</code>	real, OXE	5	8	7	80	11,522	0.6	1	7/2
<code>tokyo_u_lsmo</code>	real, OXE	5	7	13	50	11,925	0.7	1	4/2
<code>aloha_sim_insertion</code>	sim	50	14	14	50	25,000	0.8	1	3/1
<code>austin_buds</code>	real, OXE	5	7	24	50	34,112	1.1	2	6/2
<code>unitreeh1_warehouse</code>	real	50	40	19	24	11,275	0.6	0	6/2
<code>utokyo_saytap</code>	real, OXE	5	12	30	20	22,937	0.8	1	5/0
<code>nyu_rot</code>	real, OXE	5	7	7	14	440	0.4	2	6/2
Total					4,342	545,964	22.3	23	121/46

Table 2: The four detectors that ever reported HIGH.

Detector	Fires	HIGH	Reading
<code>dynamics.inverse_residual</code>	95%	50%	Suspect (§3.5)
<code>stats.dead_dimension</code>	35%	35%	Plausible
<code>label.trajecory_label_mismatch</code>	20%	20%	Plausible
<code>multimodality.contradictory_actions</code>	10%	10%	Plausible

Each dataset was scanned twice. The **full** pass reads every episode and is the basis for every number reported here. The **default** pass is what a user gets from **adduct scan**: a triage scan capped at 300 episodes, which subsamples 3 of the 20 datasets. Section 3.8 compares them. Both passes used no policy checkpoint, no target family, no calibration corpus, and no vision.

3 Results

3.1 Coverage and cost

All 20 datasets parsed, profiled, and reported with no crash and no adapter failure. Total wall-clock time was 22.5s on a laptop CPU with no GPU, median 0.8s per dataset. Seventeen of twenty produced at least one HIGH; none came back clean.

Wall time correlates with episode count at $r = 0.87$ and with frame count at only $r = 0.68$ (Figure 1), so per-episode fixed work dominates rather than raw data volume: `aloha_mobile_cabinet` reads 127,500 frames in 2.1s while `ucsd_pick_and_place` reads about half that in 4.2s with $16\times$ more episodes. Growth is sub-linear, with $97\times$ more episodes costing $10.5\times$ more time.

3.2 Selectivity

Of the 46 detectors that ran by default at the time of this run (48 registered, 2 held back; Section 4 explains why that is now 45), **27 fired at least once** and **19 never fired**. Four ever reached HIGH (Table 2, Figure 3).

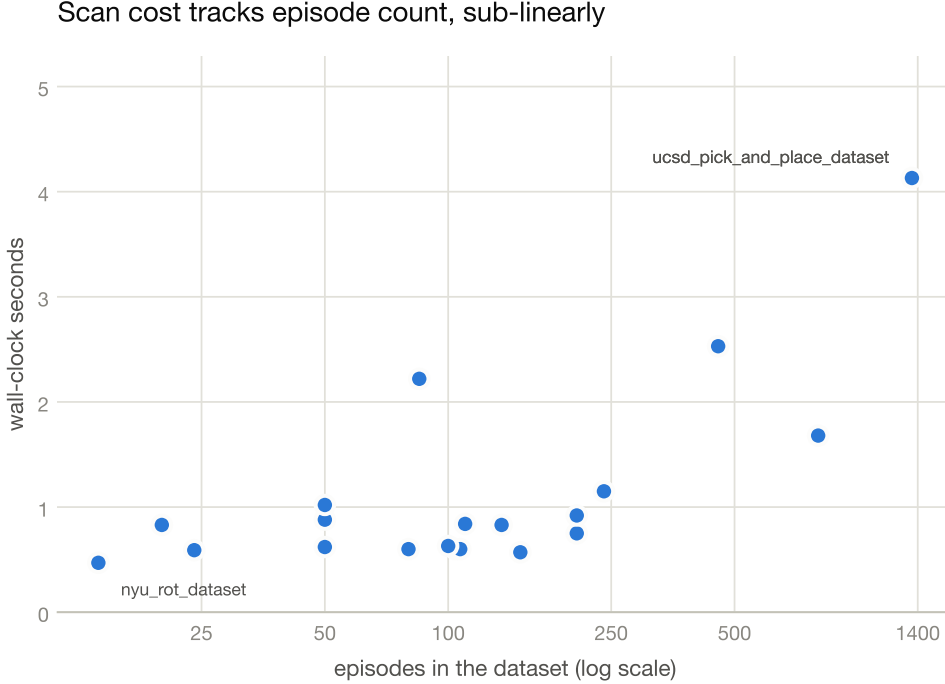


Figure 1: Scan wall-time against episode count (log x). Cost tracks episodes ($r = 0.87$) more than frames ($r = 0.68$): adduct reads Parquet columns and never decodes video, so per-episode fixed work dominates.

Table 3: Detectors firing on 70% or more of the corpus.

Detector	Fires on	Reaches HIGH
<code>dynamics.inverse_residual</code>	100%	50%
<code>dynamics.forward_residual</code>	90%	0%
<code>smoothness.jerk_outlier</code>	85%	0%
<code>smoothness.curvature</code>	75%	0%
<code>smoothness.path_efficiency</code>	70%	0%
<code>temporal.non_markovian_pause</code>	70%	0%

3.3 Finding volume is a usability defect

190 findings across 20 datasets is a median of **9.5 per dataset**. No dataset came back clean and 17 of 20 produced at least one HIGH; `utokyo_pr2_tabletop` returns 15 findings and `nyu_rot` returns 10 on 440 frames. A report that flags every dataset and returns ten complaints is not a triage tool: the two findings that mattered are lost among the eight that did not. We measure this in our own output and treat it as a defect in the product rather than a property of the data. The terminal now shows the top five clusters by severity $\times$ blast radius and names the flag carrying the rest; machine-readable outputs stay complete. That is presentation. The substantive fix is Section 3.4, and it is not done.

3.4 Six detectors are effectively always-on

Ranking by severity, as Table 2 does, hides the real problem. Ranked by fire rate, six detectors fire on 70% or more of a curated, published corpus (Table 3). **A detector that fires on 85% of curated public data carries almost no information, whatever severity it attaches.** These six escaped scrutiny in an earlier draft by being MEDIUM, but they are the bulk of the 121 MEDIUM findings and therefore the bulk of the volume problem above.

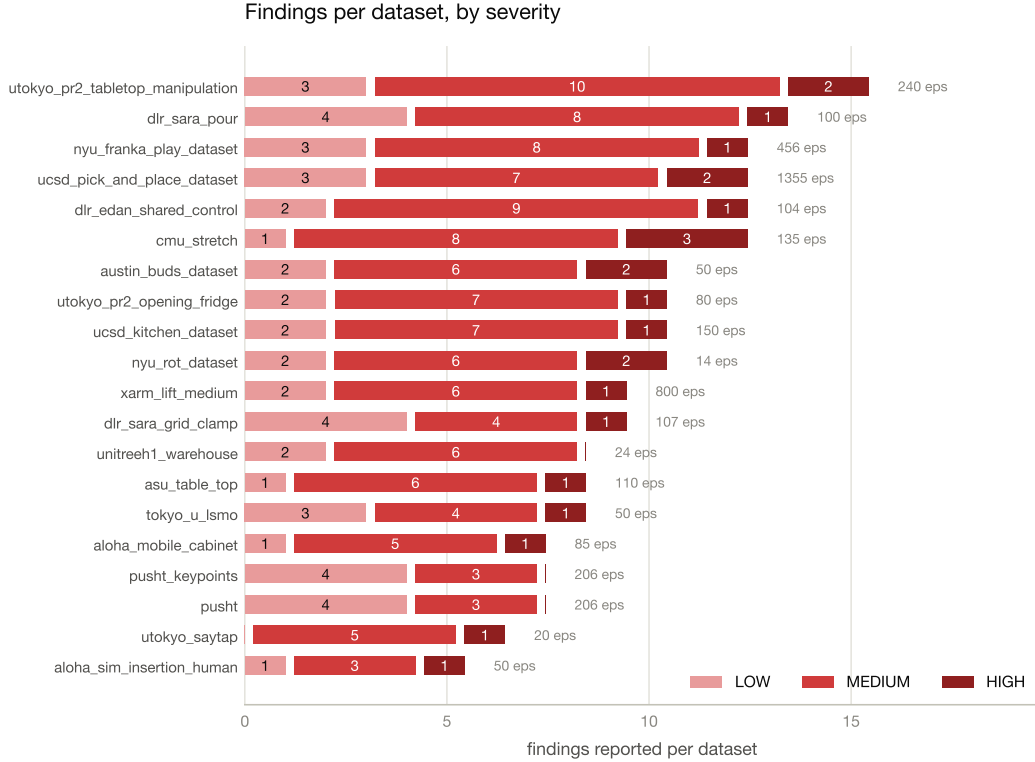


Figure 2: Findings per dataset, split by severity. Every dataset produced findings; 17 of 20 produced at least one HIGH.

3.5 `dynamics.inverse_residual`: one hypothesis tested and rejected

The detector fires on 20 of 20 datasets and returns HIGH on 10 of 20. An earlier draft proposed three explanations; one has since been tested.

Rejected: the ridge solve was ill-conditioned. Adduct builds the fit from consecutive states, which in a smooth trajectory are nearly identical, so the columns it solves over are close to duplicates of one another. The ridge penalty meant to counteract that was 10^{-6} , small enough to do nothing. The linear-algebra library underneath reported `Ill-conditioned matrix` ($\text{rcond} = 7.05 \times 10^{-17}$) on real datasets. We standardized features per fold, raised the penalty to 1.0, and made the solve abstain rather than return a residual from a fit the solver itself flags as unreliable. The numerical fault was real and is fixed. **It was not the cause of the over-firing:** the fire rate moved from 19/20 to 20/20 and the HIGH rate stayed at exactly 10/20.

Still open: the threshold ignores control rate. Fourteen of twenty datasets run at 5 Hz. At 200 ms per step, consecutive states are only weakly related by the action between them, so an inverse-dynamics residual is legitimately large without misalignment.

Still open: the finding is real. Action/observation misalignment is a known hazard of format conversion, and 14 of 20 datasets reached LeRobot through an RLDS conversion of an already-converted Open X-Embodiment dataset.

Both surviving hypotheses predict the observed split: 9/14 (64%) HIGH on 5 Hz datasets against 1/6 (17%) above 5 Hz, and 9/14 (64%) on OXE conversions against 1/6 (17%) on native datasets. **These are the same partition.** Every OXE dataset here runs at 5 Hz and every native one runs faster, so control rate and provenance are perfectly confounded and this corpus cannot separate them. Doing so needs data that breaks the confound: native recordings at 5 Hz, or converted data at higher rates.

3.6 What this run changed in adduct

Adduct’s rule for a detector it cannot yet trust is to measure it, record the number, and gate it, never to delete it. `dynamics.inverse_residual` now meets that bar and has moved to adduct’s `DEFAULT_EXCLUDED`

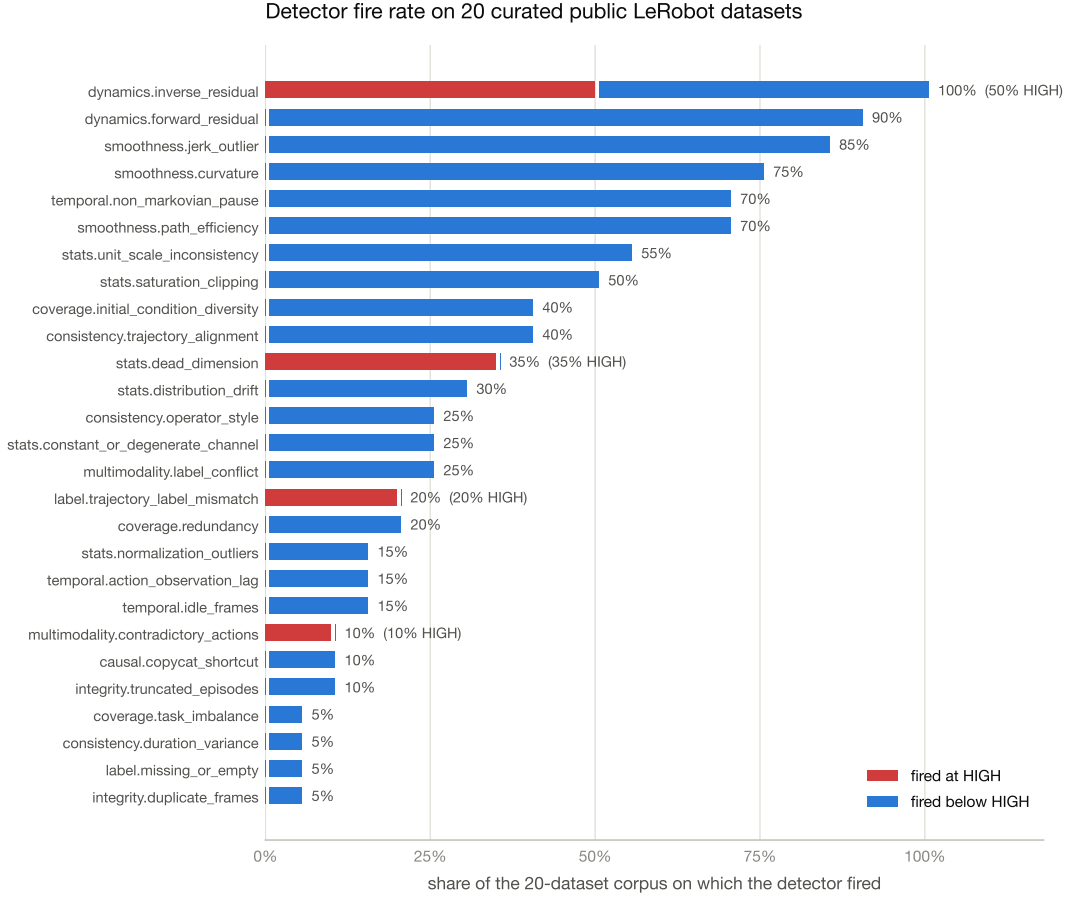


Figure 3: Per-detector fire rate across the corpus, with the HIGH share in red. Twenty-seven detectors appear; the other nineteen produced no finding on any dataset.

Table 4: The held-back set, measured on the same 20 datasets. Visibility is the share of datasets on which the detector reached the report’s visible top five.

Detector	Fires on	Reaches HIGH	In top 5
<code>dynamics.inverse_residual</code>	100%	50%	60%
<code>smoothness.discontinuity_jump</code>	70%	70%	50%
<code>integrity.declared_mismatch</code>	60%	60%	5%

set, so it no longer runs in a default scan. It remains fully implemented and reachable with `--all`. Table 4 compares it against the two detectors adduct excluded before it, measured on this corpus.

Its HIGH rate is the *lowest* of the three. The 100% fire rate decides it: a detector that fires on every curated public dataset cannot discriminate, whatever severity it attaches. Removing it from adduct’s default set takes this corpus from 23 HIGH findings to 13, and the datasets carrying at least one HIGH from 17 of 20 to 11 of 20, which is what makes a HIGH worth reading again.

`dynamics.forward_residual` is the obvious next candidate and is deliberately left in. It fires on 90% and reaches the visible top five on 75%, the highest of any detector in adduct, but never reports HIGH, so it does not break a `--fail-on HIGH` gate. Excluding two undiagnosed detectors at once would be over-correction, and this set is meant to be evidence-gated rather than a place for detectors we are merely unsure about.

3.7 The nineteen silent detectors

Nineteen of the 46 default detectors produced no finding across 4,342 episodes. This is not evidence they are broken: several target defects this corpus should not have (`integrity.nan_inf`), the vision family

had decoding disabled, and the entire POLICY↔DATA family stays silent without a checkpoint. It does mean those detectors carry no real-data evidence in this run, and their default severities rest on synthetic fixtures alone. It would be incorrect to describe this run as having validated 46 detectors.

3.8 Triage sampling does not change the conclusions

The default scan caps at 300 episodes, subsampling `ucsd_pick_and_place` (300 of 1,355), `xarm_lift_medium` (300 of 800), and `nyu_franka_play` (300 of 456). It reads 61% of episodes and 85% of frames in 21.5 s against 22.5 s.

Every HIGH finding is identical between the two passes, as are the counts of datasets scanned, datasets with at least one HIGH, detectors that fired, and detectors reaching HIGH. Two MEDIUM findings out of 190 differ (119 versus 121), and four detectors shift by one dataset each, all on the three subsampled datasets. Triage is a reasonable default on this corpus; that is a statement about these 20 datasets, not a general guarantee.

4 The Held-Back Set, and Why It Grew

Adduct’s registry defines `DEFAULT_EXCLUDED`: detectors that are implemented and tested but withheld from a default scan pending recalibration, reachable with `--all`. Before this run it held `smoothness.discontinuity_jump` and `integrity.declared_mismatch`, excluded after an earlier sweep measured them reporting HIGH on 70% and 60% of the corpus. Neither appears in any default finding here, confirming the mechanism works. This run added `dynamics.inverse_residual` (Section 3.6).

Deleting a noisy detector destroys the evidence needed to fix it and quietly narrows what adduct can see. Measuring it, recording the number, and gating it keeps both.

5 Limitations

These bound every number above.

Fire rate is not precision. Nothing here establishes that a single finding is correct. A 95% fire rate is equally consistent with “95% of public robot datasets have this defect” and with “the threshold is too tight.” No ground-truth labels exist for this corpus and none were constructed. Every rate reported here is a rate of complaint. The next step is to hand-adjudicate a stratified sample and publish a precision estimate with an explicit N .

No link to downstream policy performance. This is the most important gap. The premise behind every detector is that the defect it finds degrades a trained policy. This report does not test that premise: no policy was trained, no defect was ablated, no success rate was measured. The mechanisms are drawn from the literature, not from an experiment run here. A defensible causal claim requires training policies on data with and without a given defect and measuring the difference in rollout success. Until then, severity is a triage ordering rather than a prediction of harm.

This corpus is not the population the tool is for. Fourteen of twenty datasets are Open X-Embodiment conversions recorded at 5 Hz. The intended user teleoperates an SO-101 or similar and records natively at 30 Hz. Those are different populations, and Section 3.5 shows the difference is load-bearing: the HIGH rate is 64% on the 5 Hz conversions and 17% on everything else, so a substantial share of these findings may be measuring *this dataset survived two format conversions* rather than *this teleoperation session was poor*. This run therefore does not tell us how the tool behaves for its intended user, and because rate and provenance are confounded here, this corpus cannot diagnose its own most suspicious detector. Both are fixed by the same thing: a second corpus drawn from community-published LeRobot datasets, natively recorded and far more numerous. The bias also runs the other way and is worth keeping: these datasets are curated and widely used, so a high fire rate on them is *more* suspicious, not less.

Single machine, single run. One laptop CPU with no GPU, one pass per configuration, warm cache. Timings are indicative rather than benchmarked: no repeats, no confidence intervals.

Default configuration only. No checkpoint, target, calibration corpus, or vision. The five POLICY $\leftrightarrow$ DATA detectors were inert by construction.

The conformal gate never activated, and public metadata may prevent fixing that. Adduct's `--fpr` becomes a false-discovery bound only when a calibration corpus covers the dataset's embodiment, keeping a separate reference distribution per embodiment, which is what conformal prediction calls a Mondrian taxonomy [6]. No corpus ships with the package, so every scan fell back to a robust- z heuristic. Filling that gap meets an obstacle this run surfaced incidentally: **18 of the 20 datasets declare robot_type: "unknown"**, with only the two ALOHA datasets naming an embodiment. A Mondrian taxonomy keyed on a field that 90% of this corpus leaves blank collapses to a single wildcard bucket and forfeits the group-conditional validity the design was chosen for.

Numerical warnings are resolved, and one root cause is fixed. An earlier run leaked `LinAlgWarning` and numpy divide-by-zero, overflow and invalid-value warnings from scikit-learn to `stderr` during ordinary scans. The ill-conditioning had a real cause and is fixed at source (Section 3.5). The rest are floating-point flags raised deep inside the numerical libraries during clustering, on input adduct has already checked is finite; they are suppressed at one documented boundary, with an environment variable to restore them for debugging. This sweep produced zero warning lines on `stderr`.

Dataset revisions are not pinned. A Hub-side update will change these numbers.

6 Next Steps

In priority order.

1. **Build a second corpus from community LeRobot datasets.** Natively recorded, higher control rate, numerous. It breaks the confound in Section 3.5, measures the tool on its intended population, and is the prerequisite for recalibrating anything.
2. **Recalibrate the six always-on detectors** (Section 3.4) against that corpus. A detector firing on 85% of healthy data needs a threshold that reflects it, or it joins `DEFAULT_EXCLUDED` with the number recorded.
3. **Resolve `dynamics.inverse_residual`** by making the residual threshold control-rate aware and testing against native data at matched rates.
4. Publish a precision estimate with an explicit N from hand-adjudicated findings.
5. Run a policy-outcome experiment on at least one detector.
6. Resolve the embodiment-metadata gap before shipping a default calibration corpus.

Reproducibility

```
git clone https://github.com/prabhu-gopal/adduct && cd adduct
git checkout c8265c0
python -m venv .venv && .venv/bin/pip install -e '[dev]'
cd benchmarks/2026-08-26-lerobot-20-v0.1.0
../../.venv/bin/python scripts/run_sweep.py --out results/
```

Every measurement here was taken at `c8265c0`, when 46 detectors ran by default. Section 3.6 records the decision this run produced: `dynamics.inverse_residual` moved to `DEFAULT_EXCLUDED` in the next commit, so a default scan on adduct 0.2.0 runs 45 detectors and returns 13 HIGH findings across this corpus rather than 23. Reproducing the tables above therefore needs `c8265c0` rather than the release tag.

Raw results, figure-generation code, and this document's source are committed under `benchmarks/2026-08-26-lerobot-20-v0.1.0/`. Every figure derives from `results/sweep_full.json` alone; the committed JSON is authoritative if a figure and the text disagree.

References

- [1] Open X-Embodiment Collaboration. *Open X-Embodiment: Robotic Learning Datasets and RT-X Models*. arXiv:2310.08864, 2023.
- [2] TensorFlow Datasets catalog, Open X-Embodiment entries.
<https://www.tensorflow.org/datasets/catalog/overview>
- [3] S. Bahl et al. *Affordances from Human Videos as a Versatile Representation for Robotics*. CVPR, 2023.
- [4] J. Vogel et al. *EDAN: An EMG-controlled Daily Assistant to Help People with Physical Disabilities*. DLR Institute of Robotics and Mechatronics.
- [5] Y. Tang et al. *SayTap: Language to Quadrupedal Locomotion*. 2023.
- [6] V. Vovk, A. Gammerman, G. Shafer. *Algorithmic Learning in a Random World*. Springer, 2005.

Environment: Python 3.10.21, numpy 2.2.6, polars 1.44.0, scikit-learn 1.7.2, huggingface-hub 1.28.0, pydantic 2.13.4, rich 15.0.0. Figures: matplotlib 3.11.1, generated in a separate environment; matplotlib is a reporting tool and deliberately not a adduct dependency.