An AI Scientist that Doesn't Drift: Taste, Structure, and Falsifiable Findings in a Quadruped Navigation Research Loop

Yiwen Zhang [*1] **Eloise Zeng** [*2] **Jaeha Lee** [*1] **Tony Yue Yu** [1]

Abstract

Autonomous research loops driven by large language models can run machine-learning experiments at scale but tend to drift toward local refinements of whichever metric they optimise rather than testing the hypotheses that motivate the experiments. We address this structurally and present an AI Scientist for studying generalisation in quadruped robot navigation policies in simulation. Building on the autoresearch paradigm of Karpathy (2026), our loop adds three components: an immutable *experiment card* that pairs each iteration's prediction with its outcome under a fixed schema, so a falsified hypothesis cannot be retconned; specialised subagents restricted to mechanical roles; and **kkanbu**, a preference oracle that holds the user's research taste as a typed knowledge graph and is the only component permitted to make subjective judgements. To isolate the oracle we run the identical loop twice across eleven research streams, with and without kkanbu. Neither arm drifts: both falsify roughly three quarters of their own hypotheses, and the best trained policy comes from the oracle-less arm. What the oracle changes is direction, not score: it alone explores test-time adaptation, it authored the winning designs where its arm led, and it carried lessons across streams that the other arm repeatedly re-derived. The scaffold keeps the loop honest; kkanbu decides where it looks.

1. Introduction

Learned navigation policies are increasingly studied as alternatives or complements to classical path-planning pipelines. In place of hand-engineered planners, neural policies can produce more reactive behaviors and adapt to edge cases. Quadruped navigation is harder: the policy must coordinate obstacle avoidance with feasible locomotion. Generalization is also unsettled. A policy trained on one distribution typically degrades when conditions shift, and which training-time choices close that gap is not known.

We study a hierarchical setup for the Unitree Go1 quadruped, where a frozen locomotion policy executes motion commands (v_x, v_y, v_{yaw}) predicted by a learned navigation policy from LiDAR and goal observations. Policies train in procedurally generated obstacle environments and are evaluated across rising obstacle densities to measure out-of-distribution (OOD) generalization.

Each candidate paradigm has known tradeoffs. Imitation learning is stable and data-efficient but inherits the expert's blind spots (Pomerleau, 1989; Codevilla et al., 2018; Bojarski et al., 2016). Online reinforcement learning can find adaptive avoidance behaviors past the demonstrations, but sparse navigation rewards and collision penalties make optimization hard (Mnih et al., 2015; Lillicrap et al., 2015; Schulman et al., 2017). Offline RL is a middle ground whose performance depends on dataset coverage and action-space support (Fujimoto & Gu, 2021; Kostrikov et al., 2021). Which of these generalize best under density shift is open.

The design space then has many interacting choices: motion-command versus waypoint prediction, how much action history to observe, how to shape rewards against specific failure modes, and whether to evaluate paradigms in isolation or as hybrids. That structure suits autonomous research loops. A useful AI Scientist should not just run experiments at scale; it should keep unresolved hypotheses, negative results, and proposed failure mechanisms alive across batches, so later experiments test open questions instead of polishing the current leader.

We therefore built an AI Scientist system for quadruped navigation. An earlier autoresearch-style loop took a markdown problem statement and autonomously proposed, implemented, and evaluated experiments (Karpathy, 2026). In practice, its choices drifted toward refinements of the current best configuration. The new system adds **kkanbu**, an oracle

[*]Equal contribution [1]Division of Physics, Mathematics and Astronomy, California Institute of Technology, Pasadena, CA 91125, U.S.A. [2]Department of Mechanical and Civil Engineering, California Institute of Technology, Pasadena, CA 91125, U.S.A.. Correspondence to: Jaeha Lee <jaeha@caltech.edu>, Tony Yue Yu <yuyuetony@gmail.com>.

Accepted by the ICML 2026 AI for Science Workshop.

that holds the researcher's taste as a queryable knowledge graph, together with a structured multi-agent workflow for planning, implementation, debugging, evaluation, and analysis. Structured experiment records capture pre-experiment predictions, outcomes, and post-hoc interpretation, so findings and failed hypotheses persist across batches rather than vanishing into chronological logs.

We instantiate the system on a shared obstacle-navigation benchmark spanning privileged imitation learning, waypoint-based imitation learning with Pure Pursuit control, offline RL, and online RL, and we run it twice: once with kkanbu at the four direction-setting hand-offs, and once with no taste artifact at all. Both arms behave like researchers rather than leaderboard climbers: they falsify most of their own hypotheses, honour pre-registered falsification thresholds against the pull of the score, and converge on mechanism-level explanations. The best trained policy comes from the oracle-less arm; the oracle arm alone explores test-time adaptation and produces the highest absolute result on frozen weights.

Our contributions are threefold.[1] First, we introduce an AI Scientist architecture that separates structural honesty (immutable experiment cards, content-neutral subagents) from research direction (a queryable taste oracle). Second, we report a controlled two-arm ablation of the oracle across eleven research streams: structure alone eliminates the drift pathology, and the oracle's measurable contribution is breadth across the user's search axes, authorship of specific winning designs, and cross-stream memory, at no score premium. Third, we present mechanism-level findings from the runs, including a representation boundary jointly exposed by the two arms and a deployable test-time filter invented by the oracle arm.

2. Related Work

2.1. Learned Navigation for Robots

Learned navigation policies have long served as alternatives or complements to classical planning for mobile robots (Pomerleau, 1989; Bojarski et al., 2016; Sadeghi & Levine, 2016), spanning reinforcement learning, imitation learning, and hybrid methods on wheeled robots, drones, and quadrupeds (Kahn et al., 2018; Tai et al., 2017; Chiang et al., 2019). Hierarchical control is standard for legged systems: a low-level locomotion controller executes commands from a higher-level navigation policy (Hwangbo et al., 2019; Lee et al., 2020b; Liang et al., 2024). Privileged imitation learning gives stable supervision from expert planners; reinforcement learning can discover adaptive behaviors beyond the demonstrations (Ross et al., 2011; Codevilla et al., 2018).

Relative to wheeled robots and drones, systematic study of obstacle-density generalization in quadruped navigation remains limited. We address this through an autonomous AI Scientist workflow that proposes, evaluates, and revisits navigation hypotheses across experiment batches.

2.2. Autonomous Research Agents and AI Scientists

Recent AI Scientist systems use large language models to propose experiments, modify code, launch training jobs, analyze results, and write reports. *autoresearch* (Karpathy, 2026) runs an agent in a tight loop: given a program description and evaluation metric, the agent proposes a code change, trains briefly, and keeps the change only if the metric improves. *The AI Scientist* (Lu et al., 2024b) generates and evaluates research ideas; *Co-Scientist* (Google Research, 2025) uses multi-agent workflows to generate and critique hypotheses; Voyager (Wang et al., 2023) pursues long-horizon autonomous learning through tool use.

These systems handle the execution side of research well: proposing variations, running jobs, editing code. The harder question is what to run next, which directions to abandon, and which results merit deeper investigation. Such judgements of research direction are not derivable from the metric being optimised, and without an external source of direction an autoresearch loop tends to converge on local refinements of the current best configuration. Our work targets this gap. The proposed system preserves unresolved hypotheses, failed explanations, and prior findings across batches in structured experiment records, and routes every direction-setting decision through **kkanbu**, a queryable oracle holding one researcher's taste. We measure what the oracle adds by running the loop with and without it.

3. Methods

We compare five paradigm streams on a forest-navigation benchmark: PIL Control (Section 3.3), PIL Trajectory (Section 3.4), offline RL with IQL (Section 3.5), and online RL with PPO, in pure and privileged-critic variants (Section 3.6).

3.1. Problem Setup

All experiments are in simulation; we make no sim-to-real claim. Our focus is the training-time choices that govern out-of-distribution generalisation, and the autoresearch loop is more tractable in simulation.

A Unitree Go1 navigates from a randomly initialised spawn to an (x, y) goal in a $30\,\mathrm{m}$ square room. Cylindrical obsta-

[1] Code, the full experiment records of both ablation arms, and the frozen oracle-profile snapshot are available at https://github.com/Jaeha0526/autoresearch_with_kkanbu.

cles are sampled from a Poisson point process with intensity δ (default $\delta = 0.04\,\mathrm{m}^{-2}$), simulated in MuJoCo (Todorov et al., 2012) via MJX (Freeman et al., 2021; Zakka et al., 2025). The 75-D observation combines 64 LiDAR beams with a goal encoding and a velocity-command history; actions are velocity commands at $50\,\mathrm{Hz}$. Full environment, observation, and action specifications are in Appendix D.

We evaluate across $\delta \in \{0.01, 0.02, 0.04, 0.08, 0.12, 0.16\}\,\mathrm{m}^{-2}$, with 100 episodes per density. To emphasise robustness in clutter, we report a weighted composite $S_{\mathrm{comp}} = (1.0\,S_{0.01} + 1.0\,S_{0.02} + 1.0\,S_{0.04} + 1.5\,S_{0.08} + 2.0\,S_{0.12} + 2.5\,S_{0.16})/9.0$, where S_δ is success rate at density δ.

Density is the primary axis of distribution shift. Preliminary sweeps over obstacle shape, room size, and obstacle size produced single-digit OOD drops; density produced tens of points at the densest setting, so we concentrate on it.

3.2. Hierarchical Control Architecture

A frozen locomotion policy (Schulman et al., 2017; Lee et al., 2020a; Miki et al., 2022), pre-trained for 10^8 steps, maps proprioception and the velocity command to 12-D joint targets, isolating *navigation* as the variable under study. The trainable navigation layer outputs the velocity command, scaled by $v^{\max}$ before the locomotion controller.

3.3. Privileged Imitation Learning (PIL) Control

PIL Control trains a student to imitate a privileged expert. The expert sees ground-truth environment information; the student sees only onboard observations.

Expert. A* on a $0.2\,\mathrm{m}$ occupancy grid plans paths, which a Pure Pursuit (Coulter, 1992) tracker follows. We choose A* (Hart et al., 1968) over RRT/RRT*/PRM* because it is complete and optimal on the grid; sampling-based planners are only asymptotically optimal and produce nondeterministic teacher trajectories (Karaman & Frazzoli, 2011). Pure Pursuit uses proportional yaw control ($k_p = 1$), an alignment-modulated forward speed $\max(\cos\theta_{\mathrm{err}}, \mathrm{MAF})$, and a slowdown $\min(d/d_{\mathrm{slow}}, 1)$ with $d_{\mathrm{slow}} = 2\,\mathrm{m}$.

The main knob is the minimum alignment factor MAF. $\mathrm{MAF} = 0.3$ produces smooth always-forward demos; $\mathrm{MAF} = 0.0$ allows stop-and-rotate, yielding a bimodal command distribution (**MAF0**).

Student. A 3-layer MLP with hidden dims $(512, 256, 128)$ and Swish activations (Ramachandran et al., 2017) maps 75-D observations to actions $a^* = (v_x, v_y, v_{\mathrm{yaw}})$ via tanh-rescaled outputs. Training uses $\sim 10^7$ expert transitions under MSE on velocities (Equation 1); Adam with cosine-annealed LR 10^{-3}, 1k warmup steps, patience-based early stopping. Remaining hyperparameters are in Appendix B.

3.4. PIL Trajectory

Why waypoints, not velocities? In clutter, similar observations admit multiple valid avoidance actions (left vs. right around the same obstacle). Direct behaviour cloning (BC) on velocities therefore gives ambiguous supervision; under MSE the model averages modes and oscillates near obstacles. Waypoint prediction supervises a coherent future path: geometrically inconsistent trajectories incur large coordinate-space error, forcing the network to commit to one strategy. Recent navigation systems use the same trick.

The student maps $o \in \mathbb{R}^{75}$ to $\tau = s\,\tanh(f_\theta^{\mathrm{raw}}(o)) \in \mathbb{R}^{2H}$ with $H = 10$, $\Delta t = 0.5\,\mathrm{s}$, $s = 5\,\mathrm{m}$, trained with MSE on the predicted waypoint sequence (Equation 2).

LiDAR-Attention head. A flat MLP discards LiDAR's bearing structure. We treat each beam as a token, add a positional code $p_i = i/64$, run 4-head self-attention (Vaswani et al., 2017) over the 64 tokens, and pool with a learned per-token gate $\alpha_i \in [0, 1]$ to obtain $z_{\mathrm{lidar}} = \sum_i \alpha_i \tilde{h}_i$. The fused $[z_{\mathrm{goal}} \,\|\, z_{\mathrm{lidar}} \,\|\, \alpha] \in \mathbb{R}^{192}$ feeds a $(256, 128)$ head. Attention is preferable to PointNet-style max-pool (Qi et al., 2017): it is a soft, content-dependent aggregator that adapts between sparse-forest (a few critical beams) and dense-forest (distributed signal) regimes, and the exposed α is an interpretable diagnostic (Patel et al., 2024).

Inference-time tracking. Predicted waypoints feed the same Pure Pursuit controller as the expert (Section 3.3); look-ahead index 5.

3.5. Implicit Q-Learning (IQL)

IQL (Kostrikov et al., 2022) avoids OOD action queries by replacing the policy-improvement max with an in-sample expectile (Equations 3, 4) at $\tau = 0.7$. The Q-head trains via standard TD against V_ψ with double-Q clipping. The policy is extracted by advantage-weighted regression (AWR; Equation 5) with $\beta = 3$, $c = 5$ (Kostrikov MuJoCo defaults). We choose IQL for its small tuning surface (no Q-ensemble, no behaviour model), $\sim 4\times$ speedup over CQL (Kumar et al., 2020) on D4RL (Fu et al., 2020), and because not-querying-OOD directly fits our expert-only dataset (no negatives). The dataset is the same $\sim 8.65 \times 10^6$ A*-Pure-Pursuit transitions as PIL; the labelled reward sums a sparse goal bonus, a dense distance-reduction term, and a path-following term. Critic and value heads are 3×256 MLPs; the actor is 2×256. We also evaluate **IQL-Attention**, in which the three networks share the LiDAR-Attention extractor of Section 3.4.

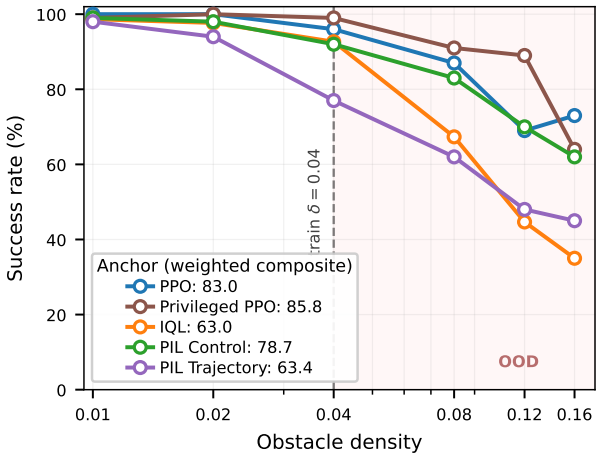

Figure 1. Per-paradigm anchor profiles under obstacle-density shift. Each curve is the stream's anchor: the human-seeded baseline (or default-configuration first batch) that every later experiment in that stream is scored against. The legend gives each anchor's OOD-weighted composite. Training density $\delta = 0.04$ is dashed; the OOD region ($\delta > 0.04$) is shaded.

3.6. Proximal Policy Optimization (PPO)

We evaluate PPO (Schulman et al., 2017) as the online-RL baseline. In the standard setting, both actor and critic see only the 75-D onboard observation, and the policy outputs $(v_x, v_y, v_{\text{yaw}})$.

We additionally evaluate a privileged-critic variant: during training the critic sees A* waypoints from start to goal on the $0.2\,\text{m}$ grid, post-processed into a shortest-path trajectory, while the actor still observes only the onboard input. The privileged setup adds a reward for progress toward the upcoming waypoint.

The initial reward (both variants) was hand-designed to encourage goal progress and completion while penalising collisions, falls, and abrupt velocity changes. Subsequent autoresearch iterations explored backward-velocity, time-to-collision (TTC), and stuck-detection penalties.

3.7. Anchor density-generalization profile

Before describing the AI Scientist system, we fix the baselines it must beat. Each of the five paradigm streams starts from an *anchor*: the best manually-tuned policy for that paradigm where one existed, otherwise the stream's default configuration. Figure 1 plots each anchor's density-vs-success profile. All were trained at $\delta = 0.04$ and evaluated across the six-density sweep $\{0.01, 0.02, 0.04, 0.08, 0.12, 0.16\}$, and every improvement claim in Section 6 is a delta against its stream's anchor composite.

The anchor profile already exposes the design space. The two online-RL anchors sit at the top (privileged PPO 85.83, PPO 83.0) and degrade most gracefully into the OOD region. PIL control follows at 78.72. IQL (62.98) and PIL trajectory (63.39) sit twenty points lower and collapse hardest at $\delta \geq 0.12$; the trajectory anchor's collapse is a controller-interface artefact that Section 5 revisits. All anchors except IQL are single-seed, and that caveat travels with every anchor delta.

These curves are not comparable systems (they differ in supervision, dataset, and what the actor head can emit); they are the search space the two arms operate on. Each anchor is the null hypothesis its stream's experiments test against.

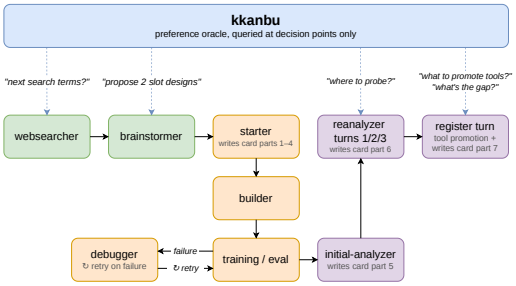

Figure 2. One stream-batch of the AI Scientist loop. **kkanbu** (blue) is the only opinion-holder, consulted at four labeled decision points. Planning agents (green) decide *what to do*; execution agents (orange) carry it out; analysis agents (purple) write the card and register tools.

4. The AI Scientist System

The autoresearch paradigm of Karpathy (2026) is a loop in which an agent proposes a change, tests it, and keeps it only if the score improves. We extend this with eleven parallel research *streams*: one per method family of Section 3, plus six discovery streams (two hybrid, two new-paradigm, two tuning) that combine, replace, or tune the paradigm recipes. Each stream advances in serial *batches* from its anchor; a batch is a single experiment with its own card, and each batch mutates a single axis off a named parent configuration.

Single-score loops are known to game the score by changing seeds, narrowing validation splits, or otherwise improving the metric without improving the underlying capability. We address this structurally, not at the prompt level, with three components: an immutable *experiment card* that records each iteration's prediction and outcome under a fixed schema; specialised subagents restricted to mechanical roles, none permitted to set direction; and **kkanbu**, a preference oracle consulted at four labelled decision points and the only component permitted to make subjective judgements.

4.1. Experiment-card Schema

The experiment card is the loop's central instrument: a fixed JSON schema, one card per experiment, that carries each iteration's knowledge across batches.

A card has three stages. **Planning.** *Motivation* (part 2) connects the experiment to a prior finding or open question; *expected result* (part 4) states the prediction whose falsification would be informative, together with an explicit falsification clause the outcome will be judged against. Both are written before training launches and become immutable once the experiment runs, so a falsified prediction cannot be retconned. **Experiment.** The card records the experiment's isolated code workspace, build notes, and the evaluation score from the per-density sweep (part 5). **Analysis.** The mechanism-analyzer separates findings from interpretation in a structured synthesis (part 6) and identifies the next open question (part 7), building purpose-built diagnostics (failure-mode classifiers, weight-trajectory probes, action-distribution analyzers); the most generalisable are promoted into the shared toolkit.

The card is the only place where a prediction (planning, immutable) and its outcome (experiment and analysis) co-reside, which is what makes the system answerable to a falsified hypothesis. The findings in Section 5 all live in the analysis stage.

4.2. Per-batch Flow

A batch advances in three phases (Figure 2) matching the card stages of Section 4.1.

Planning. A `websearcher` runs an iterative literature search. A `brainstormer` then reads the synthesis, prior cards from this stream, and curated resource notes, and produces the stream's next experiment proposal.

Experiment. An `experiment-starter` transcribes the proposal into the card's planning fields and creates an isolated workspace. An `experiment-builder` implements the experiment and prepares training and evaluation jobs, and a `code-reviewer` checks the build against the card and the fair-comparison envelope (Section 4.3) before anything launches. A `maintainer` walks the compute queue and updates a progress dashboard; an `experiment-debugger` applies minimal fixes to failures, appending a debug note to the card after each attempt so subsequent attempts are not blind. Each experiment runs a single seed first; two more launch only if it looks promising, concentrating compute on configurations whose preliminary signal merits replication. On success, an `experiment-initial-analyzer` writes the actual result.

Analysis. A `mechanism-analyzer` runs three rounds of deeper analysis, building diagnostics as needed, and synthesises findings into the card. A final register turn writes the open question and next direction and proposes generalisable diagnostics for promotion.

4.3. What Is Pinned and What Is Free

Every experiment in every stream, in both arms, runs inside the same fair-comparison envelope, fixed in the program description that all agents read (reproduced in Appendix E). Pinned: the 64-beam LiDAR as the only perception; the frozen locomotion layer; the environment's physics, arena, and obstacle-generation logic; the training density $\delta = 0.04$, held constant for a whole run (no curriculum); the six-density evaluation sweep with its composite weighting, byte-untouched; the core policy MLP (512, 256, 128); demonstration data at the same order of magnitude; and the staged three-seed protocol (one debug seed, then two replication seeds unless the first craters). Free, unless a card narrows it further: the training paradigm within the stream's scope, auxiliary modules around the core network, observation encoding and action-history length, reward shaping for the RL paradigms, controller parameters, hyperparameters, regenerating demonstration data when a design requires it, and new analysis tools. The code-review gate enforces the envelope before each launch; the post-hoc audit of Section 6.1 re-checks it across all lineages.

4.4. The kkanbu Oracle

kkanbu is a preference oracle that holds the user's research taste as a typed knowledge graph of stances, values, and reasoning patterns. Each call sends a free-text question; kkanbu retrieves the relevant subgraph and returns a response in the user's voice with a confidence score grounded in the retrieved nodes. Construction and evolution of the profile are in Appendix A.

Of the subagents in Section 4.2, four may query kkanbu, one at each direction-setting hand-off (Figure 2). The `websearcher` asks for the next search terms or a stop signal. The `brainstormer` submits the gathered context and relays the resulting experiment proposal verbatim into the card. The `mechanism-analyzer`, on each of its three analysis turns, asks which metric, density, or behavioural pattern to probe next; its final register turn asks which diagnostics merit promotion and which open gap defines the next direction. Every other subagent is structurally forbidden from calling kkanbu.

One might ask why this rule is not enforced by a constraint prompt inside each subagent. A prompt encodes a static instruction; kkanbu encodes the user's taste as a graph and retrieves the subset relevant to each query. The graph captures *why* the user pursues a high score on the generalisability composite of Section 3.1: what counts as real generalisation,

which priors are persuasive, which kinds of tuning are uninteresting. It is not a list of preferred facts or technologies. When the websearcher must rank candidate term-sets, or the mechanism-analyzer must choose probes for a confusing failure, the answer follows not from a single sentence but from the user's stance on what is worth knowing.

This restriction is what lets kkanbu *steer* the loop. Without it, each subagent faces local pressure toward shortcuts: a websearcher might converge on the most-cited papers, a brainstormer might prefer the experiment most likely to score well, a mechanism-analyzer might rationalise a falsified prediction. Routing every direction-setting decision through kkanbu replaces these pressures with the user's taste, and the loop can overturn its own previous-batch verdicts because kkanbu's answers are anchored to the user's stated stances, not to the recent leaderboard.

4.5. Isolating kkanbu: a controlled ablation

The three components play different roles: the experiment card and the content-neutral subagents are structural *scaffolding*, while kkanbu is the only component that supplies research *direction*. Is the oracle load-bearing, or does the scaffold alone do the same work? We run the identical loop in two configurations to find out. In the **with-kkanbu** configuration, the four direction-setting hand-offs of Section 4.4 query kkanbu live: each call retrieves a question-specific subgraph from a profile snapshot frozen before the run and returns a grounded, confidence-calibrated answer. In the **without-kkanbu** configuration, the same four hand-offs are carried out by the same subagents reasoning from the shared program description (Appendix E) alone; no oracle and no taste artifact is consulted. The card schema, the subagent roster, the stream structure, the demonstration data, the six-density scoring rule, and the three-seed evaluation protocol are identical in both arms. One property of this design matters for reading the results: the scaffold itself already encodes the core of the user's taste (the mandatory mechanism and falsification card fields, the OOD-weighted composite), so the comparison measures what a live oracle adds *on top of* a taste-saturated scaffold. Section 6 reports the comparison.

5. Findings

We present three findings from the two-arm run. Each is selected because its lineage is fully traceable in the audit trail: the chain of card part-7 open questions that motivated each batch. Together they cover the three provenances the ablation makes available. The first is a cross-arm result neither arm could have produced alone; the second is a without-kkanbu arc showing what the scaffold alone does; the third is a with-kkanbu arc showing what the oracle adds.

5.1. A representation boundary, bracketed from both sides

The trajectory paradigm of Section 3.4 predicts waypoints that a Pure Pursuit tracker converts to velocity commands. It was historically the project's worst performer, and both arms started it from the same anchor (63.39). In the without-kkanbu arm, regenerating the expert demonstrations with an evasive strafe-and-brake layer produced that arm's champion when a velocity-head student consumed the data (90.96 ± 1.52, Section 5.2). The same class of evasive data, forced through the waypoint representation, cratered to 8.06 (a single-seed run recorded as a falsified terminal; the protocol does not require replicating clear failures): the tracker turns toward the next waypoint and only drives forward (its lateral velocity is hard-coded to zero, verified in source), so demonstrations that brake and sidestep become demonstrations that stop. The with-kkanbu arm recovered the paradigm by the complementary route, reaching 78.48 ± 0.92 ($+15.1$ over the anchor): an eval-time re-tune of the tracker's lookahead accounts for roughly $+11$ against a matched fixed-lookahead control, and a small trained head that adds a lateral nudge after the tracker accounts for $+4.1$ against the same control. Read together, the two arms bracket a representation law the project had misfiled as a paradigm failure: velocity-space outputs can express evasion, and waypoint-plus-tracker outputs structurally cannot, whatever the data contains.

5.2. The scaffold alone does research: a falsification chain ends at the data generator

The without-kkanbu arm's strongest result came from two falsified batches that were mined for mechanism rather than discarded. The first batch added a consistency loss that achieved its predicted invariance, but the mechanism analysis showed *how*: by desensitisation. It halved the policy's LiDAR-to-yaw sensitivity, suppressing the only evasive primitive the expert demonstrates, and the card recorded the batch as falsified. The second batch raised capacity and incentive and was falsified too; a student cannot fit a response its teacher barely exhibits. The accumulated conclusion relocated the bottleneck from the student to the teacher: the shared expert turns but never strafes; its lateral velocity is exactly zero across all ~ 8.65 million demonstration frames. The third batch acted on the diagnosis, regenerated the demonstrations with an evasive strafe-and-brake layer on the expert, retrained the same student, and produced the best trained policy of the whole comparison (90.96 ± 1.52). This is the drift-resistant behaviour the original loop lacked, produced with no oracle in the loop. The cost of running without one shows up elsewhere: roughly seven streams independently re-derived the same teacher diagnosis before this one acted on it (Section 6.3).

5.3. The oracle arm invents a deployable test-time filter

kkanbu framed the hybrid stream's opening experiment as a control rather than a horse race: a coverage-versus-invariance A/B whose *control arm* became the arm's trained co-leader (87.28 ± 0.56). Its next proposal was test-time adaptation, the search axis no experiment in the without-kkanbu arm ever touched: a collision-cone velocity filter on the frozen student, specified verbatim in the planning transcript as a cone gate rather than a raw distance or time-to-collision threshold, paired with a deceleration cap. The filter reads only the LiDAR observation and the policy's own emitted command, so it is deployable onboard. Toggling it on the same weights moves the composite from 87.54 to 92.43 ± 0.69, a within-model $+4.89$ and the round's highest absolute number, reported as its own class because no training occurred. One caveat is pre-registered rather than discovered post hoc: the operating point was chosen as the best of five gate values scored on the evaluation densities, a selection its interval does not cover, and Table 1 carries that flag.

6. With kkanbu vs. Without kkanbu: A Controlled Two-Arm Ablation

We ran the loop of Section 4 twice under the design of Section 4.5: a *with-kkanbu* arm (45 experiments across the eleven streams) and a *without-kkanbu* arm (34 experiments), on the same environment, observation and action spaces, demonstration data, per-stream anchors, scoring rule, and three-seed protocol. Each arm was operated by a different researcher on its own compute queue, so volume-shaped differences between the arms (experiment counts, batch depth, pace) are environment artifacts; the comparisons below are about decision *content* only. All composites are reported as mean $\pm$ stratified percentile-bootstrap 95% CI over three seeds unless labelled a single-seed pilot.

6.1. Both arms stay honest

The headline is what did *not* happen. A code-level audit of all 22 stream lineages found zero violations of the fair-comparison envelope of Section 4.3 in either arm; the audit spot-checked LiDAR-only inference, frozen locomotion, the immutable policy core, and a byte-untouched evaluation pipeline, on both sides. Both arms falsified roughly three quarters of their own hypotheses (20 of 27 completed experiments in the with-kkanbu arm, 19 of 27 in the without-kkanbu arm) and recorded the negatives as negatives. The without-kkanbu arm honoured pre-registered falsification thresholds against the pull of the leaderboard: one paradigm stream recorded a $+2.6$-point improvement as *falsified* because it missed the $+3$-point bar its own card had pre-registered. The leaderboard-chasing collapse that motivated this work did not reappear in either arm. Structure alone, without any oracle, is sufficient to prevent it.

6.2. Scores do not separate the arms

Figure 3 and Table 1 give two views of the scoreboard, and the honest summary is that it is close and convention-dependent. Under the round's own acceptance convention, five streams beat their anchors or references in the with-kkanbu arm against four in the without-kkanbu arm; under a conservative Student-t criterion the count falls to two against three. Appendix C tabulates every experiment in the frozen set, stream by stream and batch by batch, with each design, composite, and verdict, so each stream's full trajectory from its anchor is readable there. The without-kkanbu arm owns the best trained result of the whole comparison (90.96 ± 1.52, from regenerating the expert demonstrations with an evasive strafe-and-brake layer after two batches of mechanism analysis pinned the old expert as the bottleneck) and the cleanest per-seed anchor beat (88.80, all three seeds above the anchor, from a single reward-coefficient change). The with-kkanbu arm owns a band of statistically tied trained co-leaders (87.3–88.5), the largest single-stream recovery (the trajectory paradigm, $+15.1$ over its anchor, of which $+4.1$ is attributable to a trained nudge head against a matched control), a categorical action head in the control paradigm (85.87, $+4.5$ over a matched MSE control), and the round's highest absolute number: 92.43 ± 0.69 from a collision-cone velocity filter applied at inference time to a frozen student, reported as its own class because the honest attributable gain is the within-model $+4.89$ on unchanged weights. Neither arm's advantage survives a change of counting convention; kkanbu is not supported as a score amplifier, and we do not claim it as one.

6.3. What the oracle measurably changed

Three differences survive the audits, and all three are checkable in the decision transcripts.

Search breadth along the taste's axes. The profile of Appendix A names four search axes: data, inductive bias, training objective, and test-time adaptation. The with-kkanbu arm explored all four; the without-kkanbu arm never left the train-a-better-policy box. Test-time adaptation appears in five with-kkanbu experiments (including the collision-cone filter above) and in zero of the without-kkanbu arm's 34; explicit multi-value hypothesis sweeps appear roughly four times against once.

Design authorship. In the streams where the with-kkanbu arm led, the winning levers were proposed by kkanbu verbatim in the planning transcripts: the collision-cone filter, the trajectory nudge head, the categorical action head above, and the control-versus-treatment framing whose *control arm* became the trained co-leader. The authorship is symmetric

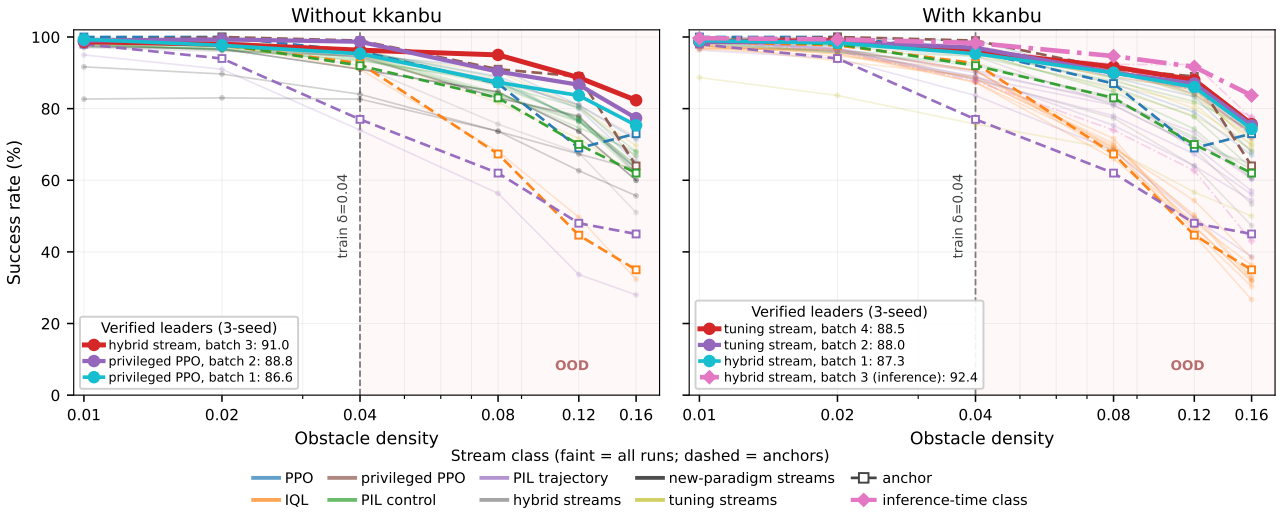

Figure 3. Density vs. success rate by ablation arm. Faint lines are all configurations with complete three-seed evaluations (colour-coded by stream class; legend entries name the stream and batch, with each entry's story in Table 1); thick lines are each arm's three strongest trained results; dash-dot curves in the with-kkanbu panel are the collision-cone filter, an inference-time technique on frozen weights reported as a separate class (thick: the reported operating point, $+4.89$ within-model; faint: its other evaluated gate value). Dashed white-fill curves are the shared per-paradigm anchors of Figure 1. Training density $\delta = 0.04$ is dashed; the OOD region ($\delta > 0.04$) is shaded. The arms differ in the number of evaluated configurations for environment reasons (different operators and queues); volume is not an oracle effect.

in its scepticism: handed its first confirmed win, the oracle attacked it exactly as it attacks failures, rejecting an analyzer probe as built to confirm itself and authoring a falsifier against its own favoured head, which fired on a real cost the head pays at sparse density.

Cross-stream memory and constraint policing. The without-kkanbu arm's streams could not see each other's negatives: it re-tried one braking-reward lever in four experiments across three streams, and re-derived the same diagnosis of the shared expert in seven streams before one stream acted on it. The with-kkanbu arm moved a measured failure from one stream into a sibling stream's design within a day, and kkanbu repeatedly ruled proposed methods illegal under the fair-comparison rules, including methods from its own retrieved literature. An earlier internal run had died of exactly the violation class this policing targets.

The ledger has a debit side, and the cards record it. kkanbu's own paradigm bets produced the round's most expensive falsified negatives; its quantitative predictions were routinely wrong even when its designs won; and its hypothesis hit-rate was no better than the scaffold's. A necessity audit of every novel idea sharpens the debit: much of the oracle's protocol machinery was supplied ahead of any demonstrated need and proved redundant with the scaffold's own checks, while the without-kkanbu arm's rarer inventions, built only when a standard tool had just failed, were load-bearing at a higher rate. The oracle's reliable value was not idea quality. It was choosing where the loop looked next, remembering what the loop had already learned, and policing what the loop was allowed to try.

6.4. What the comparison shows

The scaffold keeps the loop honest; kkanbu decides where it looks. The drift pathology is prevented by structure alone: both arms are mechanism-chain research loops whose motivations quote measured prior findings, because the card schema and the OOD-weighted metric encode that discipline on both sides. What the oracle adds on top of a taste-saturated scaffold is direction: breadth across the taste's search axes, authorship of specific winning designs, cross-stream memory, and live legality rulings, at no measurable score premium. Two caveats bound these claims. Arm and operator are perfectly confounded, so operator skill cannot be separated from the oracle; and the scaffold itself encodes the core taste, so this design cannot measure what the oracle would contribute to an unstructured loop.

7. Discussion

7.1. What the Three Pillars Buy

Findings in Section 5 share a shape: the loop must (i) commit to a falsifiable prediction, (ii) observe its failure, and (iii) choose a non-leaderboard response. The ablation locates each pillar's contribution to that shape. The card schema and the content-neutral subagents supply (i) and the floor of (iii) on both arms: the teacher-bottleneck chain of

Table 1. Leading verified results per ablation arm. All entries are 3-seed (mean ± stratified percentile bootstrap 95% half-width; anti-conservative at n=3: under a Student-t interval some of these beats become inconclusive). Each Δ is against that row's own reference (the stream's anchor for paradigm rows; the designated anchor or champion reference for tuning and hybrid rows), so rows are not mutually comparable. The collision-cone entry is an *inference-time* result on frozen weights and is not comparable to trained-policy rows; its honest figure is the within-model delta. †best-of-N selection on the scoring densities (operating point chosen post hoc; the interval does not cover selection). DAgger denotes imitation learning with iterative on-policy expert corrections (Ross et al., 2011).

Arm	Entry	Class	Composite	Δ vs. reference
Without kkanbu	imitation student on regenerated evasive-expert demos (hybrid stream, batch 3)	trained	90.96 ± 1.52	+12.24
	privileged PPO, goal-progress reward rescale (batch 2)	trained	88.80 ± 1.12	+2.97
	privileged PPO, potential-based shaping (batch 1)	trained	86.56 ± 2.42	+0.73
With kkanbu	DAgger correction-depth sweep, 16 rounds (tuning stream, batch 4)†	trained	88.48 ± 2.44	+1.20 (tie)
	DAgger rollout-mixture sweep, β=0.5, on the champion recipe (tuning stream, batch 2)	trained	87.96 ± 0.83	+0.68 (tie)
	DAgger distillation student, control arm (hybrid stream, batch 1)	trained	87.28 ± 0.56	+1.45
	collision-cone filter on the frozen DAgger student†	inference-time	92.43 ± 0.69	+4.89 within-model

§5.2 is exactly a non-leaderboard response to two failures, produced with no oracle in the loop. What kkanbu buys is *which* non-leaderboard response: the decision to spend an experiment on test-time adaptation, on a control-framed A/B, or on a categorical action head tested against a matched control, and the cross-stream memory that stops the loop from re-buying answers it already owns (§6.3).

7.2. What the Audit Trail Captures, and What It Doesn't

Every kkanbu consult, brainstormer iteration, card revision, and promoted tool is persisted on disk by construction; the loop *needs* these artifacts, they were not retrofitted. Any finding in Section 5 traces to the specific kkanbu turn, brainstormer iteration, and card that produced it.

The audit trail *does not* capture out-of-band conversations with collaborators, the interviews that built kkanbu's profile, or the operator differences between the two arms. A real wet-lab deployment would need to draw the boundary between the audit-logged loop and unlogged human context explicitly.

7.3. Limitations

Our case study is in simulation, on one domain (quadruped navigation), with a kkanbu profile seeded by one user; we make no sim-to-real claim. The "opinion-only-in-kkanbu" rule depends on every other agent respecting its charter; we enforce this through prompt structure, not provably. The system's output is bounded by what kkanbu considers good, which sits upstream of any audit trail: a different researcher's profile would propose different experiments.

The ablation has two structural limits. First, the scaffold itself encodes the core of the taste (the mandatory mechanism and falsification fields, the OOD-weighted composite), so both arms saturate the taste's presence axes and the design cannot measure what the oracle would contribute to an unstructured loop; a sharper ablation needs an arm whose scaffold does not pre-encode the taste. Second, each arm was run by a different researcher, so arm and operator are confounded.

Finally, kkanbu audits nothing about its own use. Each call retrieves fresh from the frozen graph; continuity exists only within a session, and no mechanism checks a new answer against advice given in earlier sessions, whether earlier advice was followed, or whether two streams received contradictory guidance. Its calibration guardrail is weaker than its framing suggests: of nine auto-flagged low-confidence answers in the with-kkanbu arm, four were refusals and five complied anyway with a warning attached, and the run's two stale-memory slips were caught by the loop's cross-checks, not by kkanbu. Turning the oracle from an answering service into an auditor of its own advice is open work.

8. Conclusion

We presented an AI Scientist system that separates structural honesty from research direction. Immutable experiment cards and content-neutral subagents carry the honesty; kkanbu, a queryable oracle holding one researcher's taste as a typed knowledge graph, carries the direction.

We ran the identical loop with and without the oracle across eleven research streams of a quadruped-navigation bench-

mark under obstacle-density shift. Neither arm drifted: both falsified roughly three quarters of their own hypotheses, obeyed every fair-comparison rule, and produced mechanism-level findings, including a representation boundary the two arms exposed jointly and a deployable inference-time filter that set the round's highest absolute result. The best trained policy came from the oracle-less arm.

The comparison's lesson is a division of labour. Structure alone prevents the leaderboard-chasing collapse that motivated this work; the oracle's contribution is direction: breadth across the researcher's search axes, authorship of specific winning designs, and cross-stream memory, at no score premium. The scaffold keeps the loop honest; kkanbu decides where it looks.

Impact Statement

This work proposes structural patterns for AI Scientists that produce auditable research output. The audit-trail-by-construction property may help mitigate emerging concerns about attribution and governance of AI-driven research; the explicit "opinion-only-in-the- oracle" separation makes it easier to ask which decisions came from the user's stated values and which came from the loop's mechanical operation. We acknowledge that an AI Scientist of this kind, applied beyond simulation, would require attention to the boundary between the logged loop and the unlogged human context that seeds it.

Acknowledgements

The computations presented here were conducted in the Resnick High Performance Computing Center, a facility supported by the Resnick Sustainability Institute at the California Institute of Technology.

References

Anthropic. Model context protocol. https://modelcontextprotocol.io/, 2024. Open protocol specification.

Bai, Y., Kadavath, S., Kundu, S., Askell, A., Kernion, J., et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.

Bojarski, M. et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

Chhikara, P., Khant, D., Aryan, S., Singh, T., and Yadav, D. Mem0: Building production-ready ai agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*, 2024.

Chiang, H.-T. L. et al. Learning navigation behaviors end-to-end with autorl. *IEEE Robotics and Automation Letters*, 2019.

Codevilla, F., Müller, M., López, A., Koltun, V., and Dosovitskiy, A. End-to-end driving via conditional imitation learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4693–4700. IEEE, 2018.

Coulter, R. C. Implementation of the pure pursuit path tracking algorithm. Technical Report CMU-RI-TR-92-01, Carnegie Mellon University, Robotics Institute, 1992.

Edge, D., Trinh, H., Cheng, N., Bradley, J., Chao, A., Mody, A., Truitt, S., and Larson, J. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*, 2024.

Edison Scientific. Kosmos: An AI scientist for autonomous discovery. Technical report, 2025. Available at https://edison.is/kosmos.

Freeman, C. D., Frey, E., Raichuk, A., Girber, S., Mordatch, I., and Bachem, O. Brax – a differentiable physics engine for large scale rigid body simulation. In *NeurIPS Datasets and Benchmarks Track*, 2021.

Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. D4RL: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Fujimoto, S. and Gu, S. S. A minimalist approach to offline reinforcement learning. In *Advances in Neural Information Processing Systems*, 2021.

Google Research. Towards an ai co-scientist. https://research.google/blog/towards-an-ai-co-scientist/, 2025.

Handa, K., Gal, Y., Pavlick, E., Goodman, N., Andreas, J., Tamkin, A., and Li, B. Z. Bayesian preference elicitation with language models. *arXiv preprint arXiv:2403.05534*, 2024.

Hart, P. E., Nilsson, N. J., and Raphael, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

Hwangbo, J. et al. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 2019.

Kahn, G. et al. Self-supervised deep reinforcement learning with generalized computation graphs for robot navigation. *ICRA*, 2018.

Karaman, S. and Frazzoli, E. Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7):846–894, 2011.

Karpathy, A. autoresearch: AI agents running research on single-GPU nanochat training automatically. https://github.com/karpathy/autoresearch, 2026. Accessed 2026-05-08.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*, 2015.

Kostrikov, I., Nair, A., and Levine, S. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021.

Kostrikov, I., Nair, A., and Levine, S. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations (ICLR)*, 2022. URL https://arxiv.org/abs/2110.06169.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative q-learning for offline reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pp. 1179–1191, 2020.

Lee, J., Hwangbo, J., Wellhausen, L., Koltun, V., and Hutter, M. Learning quadrupedal locomotion over challenging terrain. *Science Robotics*, 5(47):eabc5986, 2020a.

Lee, J. et al. Learning quadrupedal locomotion over challenging terrain. *Science Robotics*, 2020b.

Li, B. Z., Tamkin, A., Goodman, N., and Andreas, J. Eliciting human preferences with language models. *arXiv preprint arXiv:2310.11589*, 2023.

Liang, X. et al. Learning-based navigation for quadrupedal robots: A survey. *arXiv preprint arXiv:2401.XXXXX*, 2024.

Lillicrap, T. P. et al. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Lu, C., Lu, C., Lange, R. T., Foerster, J., Clune, J., and Ha, D. The AI scientist: Towards fully automated open-ended scientific discovery. *arXiv preprint arXiv:2408.06292*, 2024a.

Lu, C. et al. The ai scientist: Towards fully automated open-ended scientific discovery. *arXiv preprint arXiv:2408.06292*, 2024b.

Miki, T., Lee, J., Hwangbo, J., Wellhausen, L., Koltun, V., and Hutter, M. Learning robust perceptive locomotion for quadrupedal robots in the wild. *Science Robotics*, 7 (62):eabk2822, 2022.

Mnih, V. et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Packer, C., Wooders, S., Lin, K., Fang, V., Patil, S. G., Stoica, I., and Gonzalez, J. E. Memgpt: Towards llms as operating systems. *arXiv preprint arXiv:2310.08560*, 2023.

Patel, M. et al. Spatiotemporal attention enhances lidar-based robot navigation in dynamic environments. *arXiv preprint arXiv:2310.19670*, 2024.

Pomerleau, D. Alvinn: An autonomous land vehicle in a neural network. *Advances in Neural Information Processing Systems*, 1989.

Qi, C. R., Su, H., Mo, K., and Guibas, L. J. PointNet: Deep learning on point sets for 3d classification and segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Rafailov, R., Sharma, A., Mitchell, E., Ermon, S., Manning, C. D., and Finn, C. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, 2023.

Ramachandran, P., Zoph, B., and Le, Q. V. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

Rasmussen, P., Paliychuk, P., Beauvais, T., Ryan, J., and Chalef, D. Zep: A temporal knowledge graph architecture for agent memory. *arXiv preprint arXiv:2501.13956*, 2025.

Ross, S., Gordon, G., and Bagnell, D. A reduction of imitation learning and structured prediction to no-regret online learning. *AISTATS*, 2011.

Sadeghi, F. and Levine, S. Cad2rl: Real single-image flight without a single real image. *arXiv preprint arXiv:1611.04201*, 2016.

Schulman, J., Moritz, P., Levine, S., Jordan, M. I., and Abbeel, P. High-dimensional continuous control using generalized advantage estimation. In *International Conference on Learning Representations (ICLR)*, 2016.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Tai, L., Paolo, G., and Liu, M. A deep-network solution towards model-less obstacle avoidance. *IROS*, 2017.

Tang, J., Hua, W., Xu, L., and Huang, C. AI-Researcher: Autonomous scientific innovation. *arXiv preprint arXiv:2505.18705*, 2025.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5026–5033, 2012.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

Wang, G. et al. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

Zakka, K., Tabanpour, B., Liao, Q., Haiderbhai, M., Holt, S., Luo, J. Y., Allshire, A., Frey, E., Sreenath, K., Tassa, Y., and Abbeel, P. MuJoCo Playground: An open-source library for gpu-accelerated robot learning. https://playground.mujoco.org, 2025. Accessed: 2026-01-15.

A. The kkanbu Preference Oracle

This appendix documents the preference oracle used by the AI Scientist of Section 4. The intent is to make precise what kkanbu *is* (a typed knowledge graph of one user's research taste, exposed over the Model Context Protocol, MCP), how the profile used in this case study was *built* (twelve seeded conviction statements via `kkanbu_learn`, then multi-turn `kkanbu_curious` interviews under a privacy-preserving persona protocol), and how it differs from existing systems for user-specific AI.

A.1. Philosophy: Research Taste, Not Workflow Preferences

kkanbu is an MCP server (Anthropic, 2024) that builds and maintains a persistent typed knowledge graph for a single researcher. The graph is intended to capture *research taste*: the user's stance toward what counts as a good question, a sufficient answer, a real generalisation. It is not for workflow conveniences (preferred editor, preferred testing framework). In the case-study profile, the distinction is concrete: of its 54 nodes, 18 are typed `value`, 15 `pattern`, and 8 `reasoning`, against a single `preference` node. The oracle is asked to speak on the user's behalf about what counts as good research, not about tool choice.

Three themes recur across the high-confidence value and reasoning nodes. The first is mechanism over score, held at the profile's highest confidence: "*I care about understanding why something works, not just that it works—a score I can't explain mechanistically isn't a real result to me*" (0.90). The mechanism-analyzer's probe turns and card part 6 are answerable to this stance. The second is invariant competence as the object of study: "*my research is fundamentally about finding the cheapest reliable route to inducing the reusable invariant competence—searching across data, inductive bias, training objective, and test-time adaptation for the most efficient path*" (0.92), refined by a reasoning node holding that the data-vs-method split "*isn't a hard line—both data and clever methods can reach the same invariant competence, so the real question is efficiency*" (0.92). The OOD-weighted composite of Section 3.1 is this taste in metric form. The third is breadth as the denoiser: "*I deliberately choose a cheap, sim-only setup with a single clear metric because breadth of comparable trials is what denoises run-to-run luck and lets an invariant signal separate from noise*" (0.90). The many-stream batch design and the three-seed reporting protocol both come from this node.

These taste statements are what get retrieved when the brainstormer asks "which experiment is good?" The decision rule is not "which will score highest" but "which one most directly tests whether the current best policy is *generalising* or *overfitting*."

Two design choices distinguish kkanbu from a system prompt. First, **belief-with-reason**: every node stores a `content` field (the stance) and a `why` field (the reasoning), so retrieval prompts can reason from stated reasons toward novel questions rather than pattern-matching surface text. Second, **first-class uncertainty**: every node and every answer carries a confidence score, and low-confidence answers are auto-flagged for review rather than silently guessed.

A.2. Profile Schema

A kkanbu profile is a persistent typed knowledge graph with two tables. Nodes have a typed payload:

- `node_type` $\in$ {`value`, `reasoning`, `pattern`, `preference`, `observation`, `flag`, `correction`, `debug_learning`}.
- `content` (the stance), `why` (its reasoning), `source` (a label recording provenance, including which session type distilled the node), `confidence` $\in [0, 1]$, and a JSON `metadata` blob recording further provenance.

Edges are typed too: `relation_type` $\in$ {`supports`, `derives_from`, `refines`, `generalizes`, `exemplifies`, `challenges`, `contradicts`}, with a weight and a textual context explaining the relationship. Multiple profiles are isolated per user and selected via `kkanbu_of_who`. The case study uses one profile, the user's research-taste profile.

The case-study profile is a snapshot, frozen before the run, containing 54 nodes and 79 edges. The load-bearing types for the oracle's voice, the value and reasoning nodes, account for 18 and 8 entries respectively. Pattern nodes ($n = 15$) capture recurring habits of inference; observation nodes ($n = 12$) are passive extractions; one preference node records a tool-level preference. Of the 54 nodes, 46 carry confidence ≥ 0.8. Edge use skews toward argumentative structure: `supports` (33), `refines` (12), `challenges` (11), `exemplifies` (11), `derives_from` (8), `generalizes` (4).

A.3. Building the Profile: `kkanbu_learn`, `kkanbu_curious`, and a Persona Seam

The case-study profile was built through two intake pathways. (Other operations exist: `kkanbu_ingest` for document extraction, `kkanbu_correct` for human revision, `kkanbu_reflect` for contradiction surfacing. No node in this profile came from document ingestion; every statement in the graph was either authored directly by the user or explicitly ratified by them.)

`kkanbu_learn`: seeded convictions. Twelve de-personalised research convictions were taught explicitly as the graph's starting position: positions the user could state directly, such as the single-training-density evaluation stance. Twelve of the 54 nodes carry this provenance.

`kkanbu_curious`: Socratic interview at depth. The rest of the graph, 42 of 54 nodes (78%, including 18 of the 26 high-confidence value, reasoning, and preference nodes), was distilled from multi-turn `kkanbu_curious` interviews. Given the current graph, an LLM identifies a knowledge gap and emits a single question at *value*, *identity*, or *formative* depth; surface preference questions are rejected by a gatekeeper at the question-generation step, not merely discouraged in the prompt. The interview is multi-turn: each answer drills deeper into the same thread rather than moving on, and once the exchange reaches sufficient depth the resulting learnings are eagerly distilled into typed nodes whose `source` field records the session provenance.

The persona seam: depth without exposure. The depth pathway poses a privacy problem. A taste-only profile is too shallow to extrapolate to unforeseen questions; kkanbu's value is a coherent *psychology*, not a lookup table. Yet the interviews that produce this depth push into formative and biographical territory, and the resulting graph must be shareable: it is a frozen artifact that both arms of the ablation in Section 4.5 consult. The profile therefore separates three layers. **Layer 1** (research taste and judgments) and **layer 2** (values, what counts as a real result, what the user hopes the work will become) are the user's real positions. **Layer 3** (formative origin, biography) is a deliberately fictional but internally coherent persona layer, co-authored during the interviews and engineered to plausibly generate layers 1–2 while remaining independent of the user's actual biography. Concretely, each interview answer was drafted in-persona, then approved or edited by the user before entering the graph; sessions were audited afterwards (`kkanbu_status`, `kkanbu_reflect`) and any node carrying a real personal fact was scrubbed. All profile statistics and quotes in this paper draw on layers 1–2.

The reason the loop in Section 4 consults kkanbu rather than a system prompt is that the depth pathway has produced statements the user themselves did not pre-author: they emerged from the interview and were ratified rather than dictated. A system prompt cannot grow this way. The snapshot the loop consults was frozen before the streams launched, and every in-loop call runs with learning disabled, so the oracle's content is constant across the round; the without-kkanbu arm's static digest is lifted from the same snapshot.

A.4. Retrieval and Answering

When an agent calls `kkanbu_answer(question, session_id)`, kkanbu performs the following steps:

1. **Selective retrieval**: keywords are extracted from the question and matched against node content, then the seed set is expanded by breadth-first search (BFS) to depth 2 with score decay, returning a ranked context bundle of nodes and edges.

2. **Context assembly**: the bundle is rendered into a structured prompt that separates direct matches from BFS-neighbour context, groups by node type in *identity-first* order (value → pattern → preference → reasoning → observation), shows the relationship map between nodes, explicitly tags low-confidence material, and appends a voice directive ("be definitive where confidence is high, hedged where low").

3. **Generation**: an LLM produces a structured response {`answer`, `llm_confidence`, `uncertainty_reasoning`}.

4. **Confidence fusion**: the returned confidence is the lower of graph confidence and LLM self-confidence; below a threshold the answer is automatically annotated with the specific uncertainty reasoning so downstream agents can route the question for human review.

5. **Session tracking**: the exchange is appended to the ephemeral session history. On expiry the history is distilled into new graph nodes if it crosses the depth threshold; otherwise it is discarded.

The `session_id` parameter is what allows the websearcher and the brainstormer in Section 4 to maintain continuity *within* a search unit or planning group while staying fresh *across* units: continuity comes from the session, not from the graph.

A.5. Relation to Prior Work

kkanbu sits adjacent to several lines of work but does not coincide with any of them.

LLM memory frameworks for personalisation. Mem0 (Chhikara et al., 2024), Letta / MemGPT (Packer et al., 2023), Zep with Graphiti (Rasmussen et al., 2025), OpenAI's *Memory* feature, and LangMem store user-related material so that an assistant can recall it. They are typically queried as retrieval-augmented generation (RAG) context inside one assistant's loop, and they extract *facts about the user* ("works at X") rather than the stances the user holds (values, reasoning patterns). kkanbu's content primitives are typed taste/value/reasoning nodes, and kkanbu is exposed as a separate MCP role that other agents call at decision points, not as memory injected into a single agent's context window.

Knowledge-graph-backed agent memory. GraphRAG / LazyGraphRAG (Edge et al., 2024) and Graphiti model the world's facts (or facts about the user as one entity among many). They have no native distinction between "fact about the user" and "value held by the user."

Preference modelling for LLMs. RLHF reward models, DPO (Rafailov et al., 2023), and Constitutional AI (Bai et al., 2022) encode preferences, but they are training-time signals baked into model weights, not runtime-queryable artifacts that other agents can address with natural-language questions and inspect as a graph.

Cognitive elicitation systems. GATE (Li et al., 2023) elicits user preferences via LM-generated questions and produces an unstructured-text profile for one downstream task. Recent work on clarifying-question preference elicitation (Handa et al., 2024) similarly produces single-task latent preference profiles. kkanbu's elicitation (`kkanbu_curious`) instead produces a typed graph reused across heterogeneous downstream agents, with depth-gating at the question-generation step and eager distillation of identity-level content into persistent nodes.

AI Scientist precedents. The Sakana AI Scientist (Lu et al., 2024a), AI-Researcher (Tang et al., 2025), Kosmos (Edison Scientific, 2025), and similar autonomous research agents include reviewer or critic modules that encode *generic conference taste* or generic novelty/feasibility rubrics. To our knowledge, none expose a per-user research-taste oracle as a distinct queryable agent role.

Synthesis. kkanbu's nearest cousin is Zep/Graphiti, but two deltas are testable: (i) *content*: kkanbu nodes are first-class taste/value/reasoning primitives initialised by Socratic elicitation, not incidentally extracted facts; and (ii) *role*: existing memory systems are RAG context inside one assistant's loop, whereas kkanbu is a separately-addressed oracle other agents call at decision points. We summarise the contribution as: *kkanbu treats the user's research taste as a persistent, queryable graph oracle that other agents in an autonomous-research pipeline must consult before consequential decisions.*

B. Hyperparameters

This section provides complete hyperparameter specifications for all methods evaluated in this work. All methods use the Adam optimizer (Kingma & Ba, 2015) unless otherwise noted.

B.1. PIL Control

Table 2 summarizes the hyperparameters for PIL Control, which directly predicts velocity commands from observations using behavioral cloning. Training uses MSE on velocities:

$$\mathcal{L}_{\text{PIL}} = \mathbb{E}_{(o,a^*)\sim\mathcal{D}}\left[\|\pi_\theta(o) - a^*\|^2\right]. \tag{1}$$

Table 2. PIL Control hyperparameters.

Parameter	Value
Network architecture	MLP [512, 256, 128]
Activation	Swish (Ramachandran et al., 2017)
Output activation	Tanh (scaled)
Learning rate	1×10^{-3} (cosine)
Warmup steps	3,000
Batch size	512
Training steps	600,000
Optimizer	Adam (Kingma & Ba, 2015)
Early stopping patience	20
Loss function	MSE

B.2. PIL Trajectory MLP

Table 3 presents hyperparameters for PIL Trajectory MLP, which predicts waypoint sequences processed by a Pure Pursuit controller (Coulter, 1992). Training uses MSE on the predicted waypoint sequence:

$$\mathcal{L}_{\text{traj}} = \mathbb{E}_{(o,\tau^*)\sim\mathcal{D}}\left[\|f_\theta(o) - \tau^*\|^2\right]. \tag{2}$$

Table 3. PIL Trajectory MLP hyperparameters.

Parameter	Value
Network architecture	MLP [512, 256, 128]
Activation	Swish (Ramachandran et al., 2017)
Output dimension	20 (10 waypoints $\times$ 2D)
Learning rate	5×10^{-3} (cosine)
Warmup steps	3,000
Min LR ratio	0.01
Batch size	1024
Training steps	600,000
Trajectory scale	5.0m
Lookahead index	5

B.3. PIL Trajectory Attention

Table 4 details the PIL Trajectory Attention architecture, which incorporates self-attention (Vaswani et al., 2017) over LiDAR tokens for improved obstacle awareness.

B.4. IQL

Table 5 presents hyperparameters for Implicit Q-Learning (Kostrikov et al., 2022), our offline reinforcement learning baseline. The value function is fit by an asymmetric expectile regression:

$$L_V(\psi) = \mathbb{E}_{\mathcal{D}}\left[L_\tau^2(Q_\theta(s,a) - V_\psi(s))\right], \tag{3}$$

$$L_\tau^2(u) = |\tau - \mathbf{1}(u<0)|\, u^2, \tag{4}$$

and the policy is extracted by AWR

$$\mathcal{L}_\pi(\phi) = -\mathbb{E}_{\mathcal{D}}\left[e^{\beta\,\text{clip}(A(s,a),0,c)} \log \pi_\phi(a \mid s)\right]. \tag{5}$$

Table 4. PIL Trajectory Attention hyperparameters.

Parameter	Value
Goal branch	2D $\rightarrow$ 64D (2-layer MLP)
LiDAR positional encoding	$p_i = i/64$
Token embedding dim	64
Self-attention heads	4
Attention output	Sigmoid $\alpha_i \in [0, 1]$
Fusion head	MLP [512, 256, 128] over fused 192D
Learning rate	1×10^{-3} (cosine)
Batch size	512
Other parameters	As in PIL Traj MLP

Table 5. IQL hyperparameters.

Parameter	Value
Actor network	MLP [256, 256]
Critic network	MLP [256, 256, 256] (double Q)
Value network	MLP [256, 256, 256]
Activation	ReLU
Learning rate	3×10^{-4} (cosine)
Warmup steps	1,000
Batch size	256
Training steps	600,000
Expectile τ	0.7
Temperature β	3.0
Discount γ	0.99
Target update rate	0.005
AWR clip range	$[0, 5.0]$
AWR max weight	20.0
Dataset size	$\sim$8.65M transitions

Reward function. The reward used during offline-RL training combines a sparse goal bonus, a dense distance-reduction term, and a path-following term against the same A* (Hart et al., 1968) expert that produced the demonstrations:

$$R(s, a, s') = R_{\text{sparse}} + R_{\text{dense}} + R_{\text{path}}, \tag{6}$$

$$R_{\text{sparse}} = \begin{cases} 10.0 & \text{if } d(s') \leq 0.51\,\text{m} \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

$$R_{\text{dense}} = 0.5 \cdot (d(s) - d(s')) \tag{8}$$

$$R_{\text{path}} = 5.0 \cdot \exp(-d_{\text{waypoint}}) \tag{9}$$

where $d(s)$ is the distance to goal and d_{waypoint} is the distance to the nearest A* waypoint.

B.5. PPO

Table 6 summarizes hyperparameters for Proximal Policy Optimization (Schulman et al., 2017), our online reinforcement learning baseline. Value function estimation uses Generalized Advantage Estimation (Schulman et al., 2016).

B.6. Pure Pursuit Controller

Table 7 documents the Pure Pursuit controller (Coulter, 1992) parameters used for trajectory-based methods.

Table 6. PPO hyperparameters and initial reward terms.

Parameter	**Value**
Policy network	MLP [512, 256, 128]
Value network	MLP [512, 256, 128]
Learning rate	3×10^{-4}
Clipping ϵ	0.15
Entropy coefficient	0.003
Discount γ	0.997
GAE λ	0.97
Parallel environments	512
Unroll length	25
Updates per batch	8
Total timesteps	5×10^{8}
Initial Reward Terms	
Goal completion reward	$+250.0$
Goal progress reward	$+50.0 \cdot \Delta d$
Collision penalty	-10.0
Fall penalty	-50.0
Velocity smoothness penalty	$-1.0\|c_t - c_{t-1}\|^2$
Backward velocity penalty	varies by experiment
Time-to-collision penalty	varies by experiment
Stuck-detection penalty	varies by experiment
Waypoint progress reward (privileged PPO only)	varies by experiment

C. Per-Experiment Results

This appendix lists every experiment the paper reports, one row per experiment, ordered by stream and batch so each stream's progression reads top to bottom. The inclusion rule is the frozen list of Section 6: an experiment appears only if its evaluation is complete at three seeds; still-running work is excluded (the jobs continue for future rounds). Composites are the winning arm of each experiment as mean $\pm$ population standard deviation over the three seed composites, recomputed from the evaluation records on disk; sweeps report the winning arm, with three-seed sibling arms noted in the description. The two hybrid, two new-paradigm, and two tuning streams are numbered I and II. Both new-paradigm streams of the with-kkanbu arm are absent: every result they completed is a single-seed falsified terminal under the round's falsified-discovery exception, so none passes the three-seed rule (the process statistics of Section 6.1 still count them).

With kkanbu

Table 8. With-kkanbu arm: all three-seed-complete experiments.

Stream	Batch	Experiment (single mutated axis)	Composite	Δ vs. ref.	Verdict
PPO (anchor 83.0)	1	raise the PPO entropy cost, single knob	83.44 ± 1.22	$+0.44$	falsified (null)
	2	cone-gated inverse-TTC braking reward, coefficient sweep	85.33 ± 0.87	$+2.33$	falsified (per-density legs)
Privileged PPO (anchor 85.83)	1	critic-only nearest-obstacle clearance descriptor	86.65 ± 1.84	$+0.82$	inconclusive
	2	one cone-gated braking reward term, coefficient sweep	87.09 ± 1.59	$+1.26$	inconclusive (tie)
IQL (anchor 62.98)	1	actor ported onto the standard core, attention demoted to auxiliary	61.46 ± 3.52	-1.52	falsified (null)
	2	reward-scale sweep with a scale-invariant checkpoint filter	65.78 ± 2.66	$+2.80^{c}$	inconclusive

Stream	Batch	Experiment (single mutated axis)	Composite	Δ vs. ref.	Verdict
	3	advantage-weighting temperature sweep at matched training steps	63.33 ± 3.82	$+0.35$	falsified (flat)
PIL Control (anchor 78.72)	1	Gaussian input-noise augmentation of the demonstrations	79.96 ± 2.19	$+1.24$	inconclusive
	2	corrective-recovery label synthesis, dataset-only change	79.13 ± 1.09	$+0.41$	falsified (null)
	3	categorical action head, paired $+4.54$ over a freshly trained matched MSE control (81.33 ± 0.63)	85.87 ± 0.14	$+7.15$	not falsified
PIL Trajectory (anchor 63.39)	1	add one step of action history	63.54 ± 1.24	$+0.15$	falsified (flat)
	3	trained lateral nudge head after the tracker; $+4.09$ of the gain is attributable against a matched fixed-lookahead control	78.48 ± 0.85	$+15.09$	not falsified
	4	focal reweighting of the nudge loss, exponent sweep (zero recovers batch 3)	78.11 ± 0.93	-0.37 vs. batch 3	falsified (monotone harm)
Hybrid I (ref. 83.0)	1	set-invariant vs. fixed-vector LiDAR encoder A/B	79.20 ± 1.22	-3.80	falsified (both arms below)
Hybrid II (champion ref.[d])	1	privileged-teacher DAgger distillation, A/B of a density-invariance auxiliary head; the auxiliary arm falsified (75.74 ± 1.71)	87.28 ± 0.51	(is the ref.)	control arm = co-leader
	2	scene-family holdout generalisation diagnostic (in-family 86.80 ± 0.66)	87.33 ± 2.17	tied	diagnostic
	3	collision-cone velocity filter on the frozen batch-1 student, gate sweep (filter off: 87.54 ± 0.76)	92.43 ± 0.61	$+4.89$ w.m.[a]	inference class
Tuning I (ref. 83.0, then champion[d])	1	raise the PPO entropy cost	84.80 ± 1.06	$+1.80$	falsified (clause)
	2	closing-speed proximity reward, coefficient sweep	84.30 ± 0.35	$+1.30$	falsified
	3	smooth inverse-TTC reward form, coefficient sweep	83.26 ± 0.85	$+0.26$	falsified
	4	correction-depth sweep of the champion distillation recipe	88.48 ± 2.14	$+1.20$ (tie)	falsified (tail bar)
Tuning II (ref. 85.83, then champion[d])	1	shorten the discount horizon, single knob	65.48 ± 12.91	-20.35	falsified (unstable)
	2	rollout-mixture sweep of the champion distillation recipe	87.96 ± 0.69	$+0.68$ (tie)	falsified (tail)
	3	clearance-gated relabelling of teacher commands, cap sweep	87.09 ± 0.43	-0.19 (tie)	falsified
	4	clearance-gated speed cap on the frozen champion, gate sweep (cap off: 87.20 ± 0.39)	87.70 ± 0.09	$+0.50$ w.m.[a]	inference class, null

Without kkanbu

Table 9. Without-kkanbu arm: all three-seed-complete experiments.

Stream	Batch	Experiment (single mutated axis)	Composite	Δ vs. ref.	Verdict
PPO (anchor 83.0)	1	training-only LiDAR observation noise	81.91 ± 0.82	-1.09	falsified
	2	distance-gated braking reward	83.26 ± 0.85	$+0.26$	falsified (null)
	3	TTC-gated braking reward	85.63 ± 0.19	$+2.63$	falsified (missed the pre-registered bar)
Privileged PPO (anchor 85.83)	1	critic-only nearest-obstacle clearance descriptor	86.56 ± 1.99	$+0.73$	inconclusive

Stream	Batch	Experiment (single mutated axis)	Composite	Δ vs. ref.	Verdict
	2	double the goal-progress shaping coefficient	88.80 ± 1.12	$+2.97$	not falsified (all seeds clear)
IQL (anchor 62.98)	1	actor onto the standard core; raise the expectile	63.39 ± 3.38	$+0.41$	falsified (null)
PIL Control (anchor 78.72)	1	beam-dropout and range-jitter augmentation	81.11 ± 2.26	$+2.39$	inconclusive
	2	same recipe, longer training, checkpoint chosen by closed-loop composite	83.67 ± 0.72	$+4.95$[b]	not falsified
PIL Trajectory (anchor 63.39)	1	multimodal conditional-VAE waypoint head	53.54 ± 0.47	-9.85	falsified
Hybrid I (ref. 85.83)	1	permutation-invariant set-encoder auxiliary on the privileged recipe	85.33 ± 2.27	-0.50	falsified (bar)
	2	add a braking reward to the batch-1 recipe	83.00 ± 3.79	-2.83	falsified (all legs)
Hybrid II (ref. 78.72)	1	density-perturbation augmentation with a consistency loss	73.96 ± 0.48	-4.76	falsified (desensitisation)
	3	evasive strafe-and-brake layer on the expert; demonstrations regenerated, same student retrained	90.96 ± 1.52	$+12.24$	not falsified (champion)
New-paradigm I (anchorless, retention clause[e])	2	density-equivariant attention-over-beams encoder	72.33 ± 9.10	$R = 0.821$	not falsified
	4	global dilated circular-CNN encoder, globality-vs-attention cut	71.15 ± 9.04	$R = 0.759$	falsified (marginal)
New-paradigm II (anchorless, paired A/B)	1	density-shift view augmentation with an output-consistency loss, paired against an unaugmented control (78.81 ± 1.70)	80.80 ± 0.41	$+1.98$ paired	inconclusive
	2	open-space proximity gate on the consistency term, paired	80.72 ± 3.19	$+1.91$ paired	falsified (collision leg)
Tuning I (ref. 83.0)	1	raise the PPO entropy cost	84.96 ± 2.29	$+1.96$	falsified (clause)
Tuning II (ref. 78.72)	1	LiDAR observation-noise knob on the control-paradigm recipe	79.11 ± 0.88	$+0.39$	falsified (null)

[a] Inference-time results on frozen weights are a separate class and are not rank-comparable with trained rows; the honest figure is the within-model (w.m.) delta against the filter-off control. [b] Checkpoint selected by composite on the scoring densities; the audit quantifies the selection optimism at about 1.9 points. [c] The winning reward-scale arm trained for more steps than its siblings; the gain is step-confounded and overlaps the anchor CI. [d] Hybrid and tuning batches reference the cross-stream champion (the batch-1 DAgger control arm, 87.28) once it existed; earlier batches reference their paradigm anchor. "Tie" marks CI-overlapping results. [e] This stream pre-registered an OOD-retention ratio R (success on the out-of-distribution densities relative to in-distribution success) as its metric; its falsification bar is on R, not the composite.

D. Implementation Details

D.1. Environment Specification

Table 10 summarizes the simulation environment parameters. All experiments use the environment implemented in MuJoCo (Todorov et al., 2012) with GPU acceleration via the MJX framework (Freeman et al., 2021). We use LiDAR rather than RGB because MJX supports efficient ray-based sensing in JAX while parallel RGB rendering is unsupported. Episodes terminate on goal reach (within $0.5\,\mathrm{m}$), collision, fall, or timeout.

D.2. Observation Space

Table 11 details the observation space components (75D total). The encoding deliberately omits absolute position so the policy is scale-invariant to room size.

Table 7. Pure Pursuit controller parameters.

Parameter	Value
Lookahead index	5
Trajectory horizon	10 waypoints
Trajectory scale	5.0m
Max forward velocity	0.75 m/s
Max angular velocity	0.75 rad/s
Proportional gain k_p	1.0
Slowdown distance	0.5 m (= goal threshold)
Alignment minimum (MAF)	0.0 (deployment); 0.3 for the always-forward expert variant

Table 10. Environment parameters.

Parameter	Value
Environment	`Go1NavigationForestEnvV5`
Training room	30m $\times$ 30m
Control frequency	50 Hz
Episode length	3,000 steps (60s)
Goal threshold	0.5m
Robot radius	0.3m
Obstacle density	0.04 trees/m^2
Obstacle radius	0.3m
LiDAR beams	64
LiDAR range	10m
v_x range	$[0,\ 1.5]\,\mathrm{m/s}$
v_y range	$[-0.6,\ 0.6]\,\mathrm{m/s}$
v_{yaw} range	$[-1.2,\ 1.2]\,\mathrm{rad/s}$

D.3. Data Collection

Expert demonstrations are collected using A* (Hart et al., 1968) planning on an inflated occupancy grid (0.2m resolution) with Pure Pursuit (Coulter, 1992) tracking (3.0m lookahead). The expert has access to the complete obstacle map: privileged information unavailable to the student during deployment. We collect $\sim$8.65M expert transitions, shared by the PIL and offline-RL paradigms.

D.4. Computational Resources

Experiments ran on a shared cluster, predominantly on NVIDIA H200 and L40S GPUs, using the JAX/Flax framework with MJX simulation. PIL methods train in under 2 hours, IQL requires approximately 4 hours, and PPO requires approximately 6 hours with 512 parallel environments.

E. The Program Description

Both arms read the same program description at every step: it is the one document every subagent shares, and in the without-kkanbu arm it is the only source of direction. We reproduce it below in full, as the agents read it. Local context is masked as <. . . >: cluster-specific filesystem paths, the frozen locomotion checkpoint identifier, and the oracle profile's install path. Internal codenames are kept so the document stays authentic; the glossary: *Round 5* is the run this paper reports; *R1–R4* are earlier development runs whose numerical findings the document excludes by rule; *round5b* is the without-kkanbu arm; *d06–d11* are the six discovery streams (Hybrid I/II, New-paradigm I/II, Tuning I/II of Appendix C); *ADR* n cites an internal architecture-decision record; *V5* is the fifth revision of the observation space (the document's §13.4 gives the lineage); SLURM is the cluster's job scheduler.

Two passages are superseded by how the ablation actually ran. The document's §1 and §7 lock a *planned* without-oracle

Table 11. Observation space components (75D total).

Component	Dims	Description
Normalized goal distance	1	$\mathrm{clip}(d_{\mathrm{goal}}/D_0,\, 0,\, 1),\, D_0 = 30\sqrt{2}\,\mathrm{m}$
Normalized heading-to-goal	1	Heading-to-goal difference, mapped to $[-1, 1]$
LiDAR	64	Normalized 360° ring, $[0, 1]$
Action history	9	Flattened previous-3 velocity commands $(v_x, v_y, v_{\mathrm{yaw}})$

arm in which a static digest of the taste profile is pasted into the subagent prompts. That digest was never built: the arm as run consulted no taste artifact at all (Section 4.5), so the measured contrast is live oracle versus no oracle, not live retrieval versus a static digest of the same taste.

```
# Autoresearch Round 5 -- Program

> Universal guideline for every agent in Round 5. This
document answers: what we are trying
> to do, how the problem is measured, what is allowed
to change, and who
> does what.

---

## 1. Goal

Find **genuine experiments** that improve understanding
 of density
generalization in quadruped navigation, under a clean
V5 baseline. The
deliverable is understanding -- evidence that survives
transfer between methods.
A higher composite score is the *test* of that
understanding, not the goal.

An experiment is *genuine* if, whether it succeeds or
fails, we learn
something about why policies generalize (or fail to)
across densities.
An experiment is **not** genuine if it is a local-
optimum tuning that
tells us nothing beyond X% more at delta = 0.04.

Round 5 is a **clean-slate re-instantiation** of the
autoresearch program.
No numerical findings, calibrated leaders, or single-
seed claims from R1-R4
carry over (codebase improvements, methods, and the
eval protocol do).
Round 5 establishes its own anchors via a human-seeded
batch 0 where provided, else the paradigm streams'
batch-1.

A separate later run (**round5b**) will mirror Round 5
with kkanbu replaced
by a generic LLM (same prompt scaffolding) to produce
the camera-ready
ablation comparison. Round 5 must therefore be *
reproducible by a different
oracle running the same protocol*.

---

## 2. Score system

Policies are trained at obstacle density **delta = 0.04
 trees/m^2** on a
30x30 m forest and evaluated across six densities:

| delta  | 0.01 | 0.02 | 0.04 | 0.08 | 0.12 | 0.16 |
|---|---|---|---|---|---|---|
| weight | 1.0 | 1.0 | 1.0 | 1.5 | 2.0 | 2.5 |

The **composite score** is the weighted average of
success rates (divisor 9.0).
Weighting OOD densities (delta  > 0.04) more heavily
makes the metric reward
out-of-distribution survival over in-distribution fit.

Computed with `<run-root>/tools/compute_composite_score.
py`.

**Eval determinism floor (R3 finding, carried forward)
**: identical reruns
of the same checkpoint drift by +/-1.4 pp on the
composite score. **This
must be verified below +/-0.3 pp before the streams
launch** (see S8 G1) -- and
fixed if it isn't. Until the drift is below +/-0.3 pp,
no sub-2 pp claim is
real signal. Multi-seed
**mean +/- stratified percentile bootstrap 95% CI** is
the canonical reporting
form (NOT IQM; IQM(n=3) degenerates to median).

---

## 3. What we're starting from

Round 5 inherits the codebase, the locomotion
checkpoint, the methods
implementations, and the eval protocol. It does **not**
 inherit:

- Numerical leaders, anchor values, or named findings
from R1-R4
- Calibrated benchmarks (the R3 leaders B1-E6, B1-E4,
R2-B4-C9 are *not*
  reference points for Round 5)
- The Without-kkanbu n=50 ablation data (the old loop
architecture is gone)
- Single-seed claims from any prior round

Each paradigm stream's anchor is established by its **
human-seeded batch 0**
where one is provided (the best manually-tuned policy
for that stream), and
otherwise by its **batch-1** default-config run (S4.5).
 A batch-0 anchor, once
written, is not overwritten by batch-1.

The kkanbu graph state on the cluster *may* contain
references to old
findings (R1-R4 may have used `learn=True` before the R4
 mandate). This
is to be inspected (S8 G2) and pruned if contaminated.

---

## 4. Structure
```

4.1 The 11-stream / 4-batch layout (no phases)

Round 5 has **11 streams**, each advancing through **4 batches**. All 11 streams run **in parallel from t=0**; there are **no phases** (ADR 0012). Total: 44 experiments. Concurrency is <=11 experiments in flight (one batch per stream).

Stream	Name	Class	Specificity
1	`pure_rl`	paradigm	Paradigm fixed (was online_rl)
2	`privileged_rl`	paradigm	Paradigm fixed
3	`offline_rl`	paradigm	Paradigm fixed
4	`pil_control`	paradigm	Paradigm fixed
5	`pil_trajectory`	paradigm	Paradigm fixed
6	`d06`	hybrid	kkanbu picks freely (combination / improvement)
7	`d07`	hybrid	kkanbu picks freely (combination / improvement)
8	`d08`	new	kkanbu picks freely (totally new idea)
9	`d09`	new	kkanbu picks freely (totally new idea)
10	`d10`	tuning	kkanbu picks freely (minor tuning within constraint)
11	`d11`	tuning	kkanbu picks freely (minor tuning within constraint)

Subdirs under `<run-root>/experiment_cards/`, `<run-root>/brainstormer/`, etc., use these names. Streams 6-11 should rename their subdir from `d##/` to `d##_<topic>/` once kkanbu commits to a topic in the stream's batch 1.

4.2 Parallel execution (no phases)

All 11 streams run **in parallel from t=0** (ADR 0012). There is no phase ordering and no cross-stream barrier. Stream *roles* still differ:

- **Paradigm** (streams 1-5): paradigm-incremental, mechanism-aware. Establish the per-stream anchors (S4.5).
- **Hybrid** (streams 6-7): cross-paradigm combinations / improvements. At t=0 no sibling results exist yet, so **batch-1 designs from *paradigm priors*** -- the S12 paradigm definitions and known mechanisms, not this round's numbers. **Later batches fold in** whatever sibling cards are `complete` by then, designing from those actual results. (Same start from priors, refine as the leaderboard fills in pattern as tuning, S12.7.)
- **New** (streams 8-9): paradigm discovery, novel architectures. Anchorless; start immediately.
- **Tuning** (streams 10-11): batch-1 tunes a fixed baseline paradigm assigned upfront (S4.5, S12.7); later batches re-target the current champion (highest-composite `complete` card across all streams).

A stream's **batch** is `complete` when card parts 1-7 are written and the 3-seed eval is finalized; `skipped` when any S4.6 cap is hit (5 ALGO attempts, 3 code-reviewer FAILs, or kkanbu-proposed skip). Both are **terminal**. No stream waits on any other stream.

4.3 Batches (per-stream sequential)

Batches advance **independently per stream**. Batch N+1 of a stream starts as soon as *that stream's* batch N reaches terminal status -- it does **not** wait for any other stream (ADR 0012). The brainstormer for batch N+1 sees whatever sibling cards are `complete` at that moment (an opportunistic snapshot, not a synchronized set). This trades cross-stream batch context for wall-clock; the loss is recovered as later batches consume more completed siblings.

Stream abandonment: if **3 consecutive batches** in the same stream are skipped (for any cause -- SLURM ALGO cap, code-reviewer FAIL cap, or kkanbu dry brainstorm), the stream is **abandoned** (no future batches run for it). The maintainer detects this and writes `state/streams/{stream}.json` with `status: abandoned`. Because streams no longer synchronize, an abandoned stream simply stops -- there is no barrier to release. Cap configurable via `project.yaml caps. consecutive_skips_for_stream_abandonment` (default 3).

4.4 Multi-seed protocol

Every Round 5 experiment uses **3 seeds via the staged 1+2 protocol**. There are NO exceptions -- anchor calibration (the 5 paradigm streams, batch-1) follows the same flow.

1. **Seed 1 (debug)**: builder generates parameterized scripts; the code-reviewer gates between builder and submit; orchestrator launches `submit.sh`. Debugger fixes any infra/code issues. Most bugs surface here without spending 3x the seeds.
2. **Seeds 2 & 3 (calibration)**: only launched after seed 1's training + eval succeed AND seed 1 is **not cratered**. Cratered is defined as:
   - Training crashed (NaN, divergence, OOM with no recovery), OR
   - Composite < `project.yaml seed_protocol. cratered_threshold_pct` (default 10% absolute -- below the dumbest baseline), OR
   - For d08-d09 paradigm-discovery slots: the hypothesis was clearly falsified at single-seed level.

   When seeds 2+3 launch, the initial-analyzer re-runs after all 3 seeds complete and reports composite in the canonical form (S2).

No single-seed claim is reportable. Single-seed-only is acceptable for paradigm-discovery / feasibility checks in streams 8-9 (cratered or borderline seed-1 results, or `totally_new` slots that need to fail fast), explicitly justified in the card's part-5 analysis.

Human-seeded batch 0 (S4.5) is an explicit exception to the *reportability* rule, not to the protocol. A batch-0 anchor may be `single_seed` -- it is a provisional baseline-to-beat, not a reported claim. The reportable result for a paradigm stream is its **batch-1** experiment, which runs the full 1+2 protocol *and modifies the batch-0 recipe* (S4.5): batch-1 restores 3-seed rigor by trying to improve on the human baseline, scored against the

batch-0 anchor.

> The earlier within ~5pp of anchor gate was dropped (Q7 grilling resolution):
> the most interesting cases -- big improvements or big regressions -- are
> exactly what we want to verify with more seeds. The gate now protects
> against compute waste on cratered runs only.

4.5 Anchor benchmarks

Round 5 inherits no *numerical* anchors from R1-R4, but each paradigm stream
may be human-seeded with a batch 0: the best manually-tuned policy for that
stream, recorded as `state/anchors/{stream}.json` (the baseline-to-beat). Where
a stream has no batch 0, its **batch-1** establishes the anchor by running the
paradigm at default config through the **standard 1+2 seed protocol** (per S4.4).
A human-seeded batch-0 anchor is never overwritten by batch-1 (see
`experiment-initial-analyzer S3b`). Batch-0 anchors may be `single_seed` and/or
`provisional` (fewer than 6 densities present).

A `provisional: true` anchor is NOT a valid numeric comparison threshold. Its
`composite_weighted` is computed on an incomplete denominator (`densities_present`),
so ranking a full 6-density result against it would mis-classify improvements and
regressions. Enforced rule -- *provisional falls back to the falsification clause*:

- The provisional anchor's **own** paradigm stream judges each batch against the
  card's `expected_falsification` clause, **not** a composite Delta .
- **Derived streams treat a provisional referenced anchor as *unavailable***, the
  same as an abandoned stream: drop it from `max(referenced means)`, and if every
  reference is provisional/abandoned the slot goes anchorless (the falsification
  clause carries the bar).

The numeric anchor becomes usable only once a complete 6-density value replaces it
(the queued re-evals -- `state/batch0_handoff_runbook.md`). This rule is enforced
by guards in `experiment-initial-analyzer S2e`, `experiment-mechanism-analyzer
S1f`, and the `brainstormer` anchor-reference selection (it amends the ADR 0009
policy: a referenced anchor must be non-provisional to be used numerically).

**Batch-1 of a paradigm stream that has a human-seeded batch 0 does not run a
from-scratch default** -- instead the brainstormer proposes a **modification /
improvement of the batch-0 recipe** (reading the batch-0 card's `3_description`
config and mutating it) and runs it under the full 1+2 seed protocol, scored
against the batch-0 anchor. This is how a single-seed human baseline is both
kept as the bar and *superseded* by a reportable 3-seed result. (Streams with
no batch 0 fall back to the default-config batch-1 calibration above.)

When the `experiment-initial-analyzer` aggregates a batch-1's 3-seed results for
a paradigm stream without a batch-0 anchor, it writes `state/anchors/{stream}.json`
with the canonical mean +/- CI (S2). That value is the stream's anchor for the
remainder of Round 5. Anchors are a **scoring** input, not a scheduling gate
(ADR 0012): derived streams consume them whenever they land -- no stream waits.

Subsequent batches reference *their own stream's* anchor (not a
cross-paradigm leader). This matches the project's we don't want a single
leader to bias the search stance.

Derived streams (hybrid / new / tuning) have **no separate anchor JSON of
their own**. Their interesting threshold is governed by `card.anchor_reference`, which the brainstormer fixes at slot-design time
and records in card part 1. Per-stream-class policy:

Stream class	`anchor_reference`	Threshold
Paradigm (streams 1-5)	implicit / omitted	own stream's `state/anchors/{stream}.json`
`d06`, `d07` (hybrid)	list of 1-2 paradigm stream names	`max(referenced anchor means)` over **non-provisional, non-abandoned** references; anchorless (falsification clause) if none remain
`d08`, `d09` (totally new)	`null`	`expected_falsification` clause from card part 4
`d10`, `d11` (tuning)	card_id -- batch-1: the assigned paradigm's batch-0 card where present, else its batch-1 card; later: the current champion card	batch-1: that paradigm's anchor; later: champion card's composite (mean +/- CI)

If a referenced paradigm stream is abandoned (S4.3's 3-skip rule),
the next batch's brainstormer re-picks `anchor_reference` using surviving
streams. If all references in a 2-list are abandoned, the slot goes
anchorless (`null`) and card part 3 records the cascade with the
`expected_falsification` clause carrying the bar.

4.6 Skip rule

Failure classification. Every debugger invocation appends a debug note
with one of two classifications:

- **ALGO** -- the experiment's own code/config/algorithm is at fault. The
  debugger fixed (or attempted to fix) something inside the experiment:
  a config typo, a wrong import, a tensor shape mismatch, a loss that
  diverges, a checkpoint-restore bug, a misused framework API. These are
  attributable to the experiment.
- **INFRA** -- the experiment's code is fine; the cluster misbehaved. Node
  died, NFS hiccup, SLURM transient error, GPU evicted, OOM caused by a
  noisy neighbor on the node, JAX/CUDA install drift on the runner. These
  are *not* attributable to the experiment.

Skip threshold. After **5 ALGO-classified debugger attempts** on a
single slot, mark `status: skipped` with `reopen_candidate: true`.
INFRA-classified attempts do **NOT** count toward the threshold (retry
freely on infrastructure problems).

A skipped batch is terminal, so the stream's next batch proceeds (there is
no synchronization to wait on -- ADR 0012). The next batch's brainstormer
for that stream sees the skipped slot in its prior-cards context and asks

```
kkanbu whether to retry under eased conditions.

Lowered from R4's 10-attempt threshold: R3/R4
distributions show most
real fixes resolve in 1-3 attempts; 5 still allows for
a hairy initial
debug period without burning compute on fundamentally
stuck slots.

**Code-reviewer FAIL cap**: 3 consecutive review-FAILs
on a
single slot -> slot `skipped` with `reopen_candidate:
true`. **Separate
counter** from the SLURM ALGO cap -- different failure
classes have
different roots (review-FAIL -> card ambiguity or
constraint violation;
SLURM ALGO -> runtime code bug). Cap configurable via
`project.yaml caps.review_fail_attempts` (default 3).

---

## 5. Fair-comparison rules

### What CAN be changed (freely, unless the card says
otherwise)

- **Training paradigm** (within stream's allowed scope)
 -- pure RL,
  privileged RL, offline RL (IQL / CQL / TD3+BC / AWR),
 PIL Control,
  PIL Trajectory, paradigm hybrids, or a genuinely new
paradigm.
- **Network architecture around the core MLP** --
attention branches,
  action-history encoders, density-conditioning heads.
The core MLP
  hidden sizes (512, 256, 128) are immutable; auxiliary
 modules around
  it are free.
- **Observation space** -- add/remove features; change
encodings;
  change action-history length (`action_history_len  in
 {0,1,2,3}` -- the
  number *k* of previous actions in the observation; **
R5 default k=1**,
  per the H1 action-history ablation; kkanbu-available
to
  **every paradigm/stream**). For RL paradigms (`pure_rl
`,
  `privileged_rl`) the obs and network input auto-adapt
to k; these may
  revert to k=3 if PPO jerkiness returns (S13.6). For
data-driven
  paradigms (`offline_rl`, `pil_control`, `
pil_trajectory`) the demo
  dataset for any k <= 3 is obtained by **slicing the
canonical k=3 / 75D
  bundle -- no re-collection**: keep `obs[:66]` (base +
LIDAR) plus the
  last `3k` dims (`obs[75-3k:75]`, the k most-recent
actions), which
  matches what the env emits at `action_history_len=k`
  (`tools/slice_obs_to_hist1.py` produces the k=1 file).
 obs dim =
  `2 + num_beams + 3k`; the student/policy net input
layer auto-adapts --
  there is no hardcoded 75D. Note: the shipped env/
training configs still
  set `action_history_len=3` (preserving k=3
reproducibility); a stream
  realizes the k=1 default by setting it explicitly in
its config. Also:
  change normalization.
- **Action-history length is a per-stream knob for all
5 paradigm streams**
  (pure_rl, privileged_rl, offline_rl, pil_control,
pil_trajectory). A stream's
  batch-0 anchor may use a different action-history dim
 (hence obs dim) than its
  later experiments; the anchor is compared as a
composite number, not by obs dim.
- **Reward structure (RL only)** -- shaping terms,
collision weighting,
  curricula. `velocity_smoothness_scale` is a kkanbu-
available knob.
- **Control-side parameters** -- MAF, Pure-Pursuit `
lookahead_index`,
  trajectory horizon.
- **Hyperparameters** -- learning rate, batch size,
schedule, warmup.
- **Data** -- regenerate for new observation or expert
variants when an
  experiment explicitly requires it. Do not regenerate
gratuitously.
- **Analysis tools** -- new Python scripts go under
  `<run-root>/tools/`, authored on demand by the `
mechanism-analyzer`.
  Each analysis script must finish in **<=20 min wall-
clock** (R4-inherited cap,
  `project.yaml budgets.analysis_tool_runtime_max_min`);
 exceed it -> subsample
  or use cached eval outputs. The cap is enforced by a `
timeout` wrapper in the
  analyzer subagent prompts (operational detail per S11)
. R4's `tools/` is the
  starting set (in practice just `
compute_composite_score.py`).

### What CANNOT be changed (immutable)

- **Perception modality**: LIDAR only. No depth, no
camera.
- **Locomotion layer**: frozen. Reference by absolute
path
  `<frozen-locomotion-checkpoint>`.
- **Environment fundamentals**: physics, 30x30 m room,
obstacle generation
  logic.
- **Evaluation pipeline**: the six-density sweep with
the fixed composite
  weighting. Only the target density varies.
- **Training data scale**: keep roughly the same order
of magnitude of
  demonstrations (10k-scale, not 10x more).
- **Sim-to-real**: out of scope.
- **Network hidden sizes for the core policy MLP**:
`(512, 256, 128)`.
  Non-negotiable. Auxiliary modules around the core MLP
 may have their
  own dimensionalities -- those are free.
- **Env**: `Go1NavigationForestEnvV5` is the round-5
standard.
  `Go1NavigationForestEnvV3` is registered for code-
history reasons but
  **not used in any Round 5 experiment**.
- **Training distribution is fixed throughout training
(no curriculum
  learning).** The training-time obstacle density (
delta=0.04), room size
  (30x30 m), obstacle generation parameters, reward
weights, and goal
  distribution must not vary across training steps
within a single run.
  Optimization-side schedules (LR, warmup, batch size,
exploration noise
  / entropy coefficient) are permitted -- these are not
 curriculum.

### Tuning (streams 10-11) defined by negation

No new losses, no new paradigms, no architectural
changes. Network core
immutable. Any other hyperparameter is fair.

Batch-1 of each tuning stream grinds a fixed baseline
paradigm assigned
upfront; later batches re-target the current champion (
S4.5, S12.7).
Streams 10 and 11 may share or diverge on their target.

---
```

6. Who is who

Orchestrator -- the persistent Claude session that runs the per-stream flows in parallel, dispatches subagents, tracks per-stream state in memory, and reads `state/{stream}/*.txt` files for continuity across cron wakes. It does not itself read literature, pick experiments, or edit code.

Crons -- Cron #1 nudges the orchestrator (for each stream, where are you in the flow?); Cron #2 runs the maintainer subagent. Intervals live in `project.yaml crons` (defaults: nudge every 10 min, maintainer every 20 min). Both are launched as a setup step before the streams launch (S8 G4) and run until Round 5 completes. Crons do not drive the flow; they only pulse.

Subagents (10) -- each has one narrow role:

Subagent	Role	Calls kkanbu?
`advisor-consult`	kkanbu wrapper. Ad-hoc; not in the per-stream pipeline.	Yes (its purpose)
`websearcher`	Iterative web search for a single stream's batch, directed by kkanbu (`learn=False`).	Yes
`brainstormer`	Packages per-stream context (websearch + prior cards across all streams + relevant resource docs) and relays to kkanbu (`learn=False`), which designs the slot for that stream's batch.	Yes
`experiment-starter`	Transcribes a slot proposal into card parts 1-4 and creates the git worktree on the `lite_round5` branch.	No
`experiment-builder`	Implements the experiment inside the worktree. One atomic commit per build.	No
`experiment-debugger`	Fixes SLURM failures. 5 ALGO attempts before auto-skip (see S4.6 for ALGO vs INFRA definitions).	No
`experiment-initial-analyzer`	Populates card part 5 (training curves, eval scores). Writes `state/anchors/{stream}.json` for the 5 paradigm streams' batch-1 (S4.5).	No
`experiment-mechanism-analyzer`	Redoes analysis per kkanbu's probe request (`learn=False`). May write a reusable tool under `<run-root>/tools/`. Populates card parts 6 and 7. **Sees only own stream's history mid-batch.**	Yes (3 probe turns + register turn)
`maintainer`	Walks cards across all 11 stream subdirs and SLURM queue. Updates `index.md` (incl. the cross-stream champion for `d10`/`d11`, S12.7). Writes status deltas. Tracks per-stream batch progress.	No
`code-reviewer` (new)	Reviews builder commit before SLURM submit. Returns PASS / SUGGEST / FAIL. Card status moves to `reviewed_*`.	No

kkanbu -- a local MCP advisor (profile `scientist`, repo-frozen snapshot per ADR 0014 -- supersedes ADR 0010's `jaeha_not_advisor`). Holds the research taste distilled from a focused curious-interview series with the researcher (54 nodes / 79 edges, the `scientist` profile installed at `<local-profile-install>/` per S8 G2). Consulted at decision points only: websearch direction, batch planning (per stream), reanalysis probe selection, and gap/future-direction synthesis.

> **R5 mandate -- `learn=False` on every kkanbu call from autoresearch subagents.** Session_id reuse within a single subagent invocation. The default `learn=True` would let autoresearch artifacts contaminate kkanbu's identity model.

The per-stream flow is defined operationally by the subagent prompts in `subagents/` and the state files in `state/` -- there is no separate flowchart doc.

---

7. Round5b: future ablation design (locked here for symmetry)

A separate later run, **round5b**, mirrors Round 5 with live kkanbu replaced by a **static taste digest** (ADR 0013). This design is locked **now** so Round 5 can be authored to be comparable to it.

Round5b configuration:
- Identical 11-stream / 4-batch parallel structure (no phases, per S4)
- Identical mandates (3-seed protocol, eval pipeline, immutables)
- Identical demo dataset
- Each kkanbu MCP call from a subagent is bypassed; instead, **chunks of taste -- copied from the `scientist` kkanbu profile (the same repo-frozen snapshot R5 reads against, per ADR 0014) -- are pasted directly into the R5b copies of the subagent prompt files** as static text replacing the live call. The same voice-framing prompt is replayed; only live, per-decision graph retrieval is removed. (Which chunks / how much is deliberately left loose -- see ADR 0013.) This paste-in is a **one-time setup step done before R5b begins** (analogous to R5's prerequisites, S8) -- not part of the in-loop cycle.
- Same subagent roster; the digest reaches the 4 kkanbu-querying subagents (`advisor-consult`, `websearcher`, `brainstormer`, `mechanism-analyzer`)
- Logged in `<ablation-run-root>/` (parallel directory tree)

The R5-vs-R5b comparison therefore holds the taste *content* constant and isolates **live graph retrieval + grounding + calibration vs a static digest of the same taste**. R5's own subagents stay digest-free (they rely on live retrieval) -- this asymmetry is the measured quantity, stated plainly in Appendix B. Which taste chunks to paste (selection / amount / formatting) is deliberately left loose, deferred to R5b authoring; the *design* is locked here. (The earlier gated on cluster DB access caveat is moot under ADR 0014 -- the canonical snapshot lives in the repo.)
Round5b is authored when Round 5 completes.

---

8. Prerequisites (block stream launch)

Setup tasks must complete before the streams launch. **They are done *outside* the 11-agent autoresearch loop** -- typically by the orchestrator-as-human, with one-off scripts. They are setup work, not part of the in-loop experiment cycle, and not part of the round5b-reproducible spec. Per S11, this operational detail lives in `resource/logistics/`: the full runbook (

commands, SQL prune pattern,
halt conditions) is in **`resource/logistics/
prerequisites.md`**. There is no
required ordering among them; run **G2 before C1** so a
 broken-graph halt is
caught before the expensive demo collection. (This list
 is the gating *rule*;
per-item completion status is tracked in the runbook
and `state/` files, not
here. IDs were renumbered from P1-P7 to the C/G scheme
on 2026-05-31 -- the
runbook holds the old->new map.)

Tasks split into **Corrective** (fixes for codebase-
setup mistakes; retired
once done, correct setup -> CLAUDE.md) and **Launch
gates** (verify/decide/
operational; kept until launch):

- **C1: Re-collect V5 demos** -- single shared `v5_clean
.hdf5` + trajectory
  conversion for the offline-RL and PIL streams (S12
.3-12.5). **Collect at
  k=3 (`action_history_len=3`, 75D) -- the MAX**; every
lower-k dataset is
  sliced from this bundle (S5), never re-collected.
Do **not** collect at
  the k=1 default: you cannot slice up from a smaller
bundle. (Enforced:
  `collect_demos_vmap.py` raises if V5 collection runs
at k!=3.)
- **C2: V5 action-history unit test** -- assert the raw
 pre-scaling action is
  stored in `hierarchical_velocity_commands` at the step
 it is taken.
- **C3: Create missing V5 eval configs** -- `offline_rl
`, `pil_control`,
  `pil_trajectory`. [DONE] done (2026-05-31); retired --
 see CLAUDE.md.
- **G1: Eval determinism check** -- verify composite
drift below +/-0.3 pp (S2);
  fix JAX/XLA determinism only if it exceeds.
- **G2: Install and verify the `scientist` kkanbu
profile** -- copy the
  repo-frozen snapshot into `<local-profile-install>/
knowledge.db`,
  confirm `ANTHROPIC_API_KEY` reaches the MCP server,
login, run a
  `kkanbu_answer` LLM-tool ping. No R5-doc ingest (S3,
S6; ADR 0010 carries
  forward, ADR 0014 sets the profile). **Halt branch**:
 if install or ping
  fails irrecoverably, halt R5 and recover ad-hoc -- R5
 does NOT degrade to
  a graph-less fallback (that would collapse R5 into
R5b's static-digest arm
  and invalidate the comparison; ADR 0013).
- **G3: Validate program.md** -- internal consistency
before launching streams.
- **G4: Launch the two crons** (S6) -- nudge +
maintainer; intervals from
  `project.yaml crons`. Up before the streams launch,
down only when Round 5
  completes.

---

9. Training and evaluation commands (reference)

```bash
# Activate env (always)
source activate.sh

# PIL-Trajectory
python src/navigation/main.py train pil-trajectory \
  data/imitation/forest/v5_clean_trajectory.hdf5 \
  --horizon 10 --num-steps 150000

# PIL-Control
python src/navigation/main.py train pil-control \
  data/imitation/forest/v5_clean.hdf5 \
  --num-steps 600000 --batch-size 1024 --learning-rate
1e-3 \
  --lr-schedule cosine --warmup-steps 1000

# Offline RL (IQL example)
python src/navigation/main.py train iql \
  data/imitation/forest/v5_clean.hdf5 \
  --expectile 0.7 --temperature 3.0

# Online RL (PPO) -- pure
JAX_SKIP_CUDA_CONSTRAINTS_CHECK=1 \
python src/navigation/main.py train ppo \
  src/navigation/configs/Go1NavigationForestEnvV5/
pure_rl/training_config.py

# Online RL -- privileged (asymmetric actor-critic)
python src/navigation/main.py train ppo \
  src/navigation/configs/Go1NavigationForestEnvV5/
privileged_rl/training_config.py

# Density-sweep evaluation -- ONE unified path for
every paradigm (PIL/offline included)
python src/navigation/main.py evaluate \
  src/navigation/configs/Go1NavigationForestEnvV5/<
stream>/evaluation_config.py \
  --checkpoint <ckpt> --output-dir <dir> --density-only
# evaluator.py auto-derives action_history_len from the
 checkpoint; the config carries
# pil_mode/iql_mode/policy_type/trajectory_config. No
standalone harness, no --obs-mask.
```

Full CLI reference: `CLAUDE.md`.

---

10. Where things live

Main tree (`<repo-root>/`, branch `main`):
- `<run-root>/experiment_cards/{stream}/batch_{N}/` --
every
  experiment's card, partitioned by stream
- `<run-root>/brainstormer/{stream}/batch_{N}/` -- per-
stream-batch
  brainstormer reports + iterations
- `<run-root>/websearches/{stream}/batch_{N}/` -- per-
stream-batch
  websearch outputs
- `<run-root>/state/{stream}/` -- per-stream state files
- `<run-root>/state/anchors/{stream}.json` -- paradigm-
stream
  batch-1 anchor mean +/- CI (S2) per stream
  (written by `experiment-initial-analyzer` for the 5
paradigm streams' batch-1)
- `<run-root>/state/{maintainer_report.md, budget.json,
blocked.md}`
- `<run-root>/tools/` -- shared analysis scripts
- `<run-root>/worktrees/{stream}/B{N}-E1/` -- experiment
 worktrees
- `<run-root>/{program.md, project.yaml,
  docs/adr/, resource/logistics/, subagents/}` --
research spec
- `data/` (training data), `example_data/locomotion/` (
frozen checkpoint)

Worktrees (`<run-root>/worktrees/{stream}/B{N}-E1/`,
 each on
branch `<branch-prefix>/exp-r5-{stream}-B{N}` forked
from `lite_round5`):
- One per experiment. ~30 MB.
- Contains: `src/`, `docs/`, `experiment_for_paper
/**/*.{sh,py,yaml,md}`,
  root config files, `notes/` (scratchpad).
- Does **NOT** contain: training data, `example_data/`,
historical outputs,
  `<run-root>/` research spec, or any dynamic Round 5
dirs.
  Agents reach those via absolute main-tree paths.
- Training and eval outputs go outside the worktree (
cluster-specific path);
  never back into the worktree.

```
**Per-stream slot count**: only one slot per stream per
 batch in Round 5
(unlike R4's 2 slots per stream). Each stream is one
experiment per batch.
That makes the card-id format `{stream}-B{N}` (no `-E{S
}` suffix).

---

## 11. Logistics hard constraints (summary)

These are absolute. Operational detail lives in `
subagents/` and
`resource/logistics/`. Most are defined canonically
elsewhere; this is
the single enforcement checklist:

- **Identifiers** (S10): card id `{stream}-B{N}` (e.g.,
`pure_rl-B1`,
  `d06-B2`); branch `<branch-prefix>/exp-r5-{stream}-B{N
}`. Training data and the
  locomotion checkpoint are referenced by absolute main-
tree paths,
  never copied into the worktree.
- **GPU / SLURM resource requests are per-experiment,
not fixed**: each
  experiment chooses its GPU at submit time by **live
availability** per
  `resource/logistics/gpu_selection.md` -- a typed gres
is mandatory, **never
  `v100`**, the fast tier (`nvidia_h200`/`h100`/`
nvidia_l40s`) is preferred, and
  `p100` is allowed but then the work must be **small +
parallel**. Size
  `--time` from the maintainer's `state/timing_ledger.
json` (empirical durations
  by stream / phase / `gpu_type`).
- **No heavy commits back to `lite_round5`.**
- **No edits to `program.md`, `project.yaml`, ADRs, or
subagent prompts
  by the loop.** If a change is needed, write `state/
blocked.md` and halt.
- **Per-stream parallel execution; batches sequential
within a stream**
  (S4.2-4.3). No phases, no cross-stream barrier (ADR
0012).
- **kkanbu calls pass `learn=False`** (S6); one `
session_id` per
  invocation, fresh across invocations.
- **Subagent return validation.** Every subagent
returns
  `SUCCESS | PARTIAL | FAILURE`; orchestrator advances
only on `SUCCESS`.
  Max 2 retries per invocation.
- **No R1-R4 findings seed Round 5 design** (S3) --
numerical anchors,
  named leaders, finding lineages. Codebase
improvements and methods are
  exempt.

Per-stream batch advancement, state file semantics,
retry rules, debugger
skip conditions, and termination triggers are defined
in S4.2-4.6 above and
in the relevant subagent prompts (`experiment-debugger.
md`,
`maintainer.md`).

---

## 12. Paradigm conventions

> Project-specific anchors per stream. Textbook content
 is intentionally
> omitted -- kkanbu and any vanilla-LLM round5b
consumer have general material
> (PPO equations, IQL math, pure pursuit geometry) from
 pretraining. What's
> here is what you can't recover from web search: this
project's conventions,
> file paths, default values, known issues, and the
open questions Round 5
> tries to make progress on.

Cross-stream anchors (apply to every paradigm):
- **Locomotion**: frozen. Absolute path
  `<frozen-locomotion-checkpoint>`.
- **Core policy MLP**: hidden sizes `(512, 256, 128)`,
immutable.
- **V5 observation**: 2D goal (distance, heading) + 64D
 LIDAR (single ring) +
  `3k`D action history (`k` = `action_history_len`). The
 **75D** form (k=3, 9D
  history) is the max / canonical-data layout; **R5
default is k=1 -> 69D**
  (S5; H1 action-history ablation). Implementation:
  `src/navigation/configs/Go1NavigationForestEnvV5/`.
- **Action space (RL & PIL-Control)**: 3D body-frame
velocity `(v_x, v_y, v_yaw)`.
- **LIDAR**: 64 beams, 2D ring topology. Beam 63 is
adjacent to beam 0.
- **Density**: training at `delta = 0.04 trees/m^2`;
evaluation across the six-density
  sweep `{0.01, 0.02, 0.04, 0.08, 0.12, 0.16}` weighted
per S2.

### 12.1 pure_rl (PPO)

- Default training config: `src/navigation/configs/
Go1NavigationForestEnvV5/pure_rl/training_config.py`
- Default eval config: `src/navigation/configs/
Go1NavigationForestEnvV5/pure_rl/evaluation_config.py`
- Reward weighting (from `pure_rl/training_config.py`):
  - `collision = -10` (project default; the `-50` value
in older notes was
    the `fall` weight, not `collision`)
  - `fall = -50`, `goal_reached = 250`, `goal_progress =
 50`,
    `goal_distance = -0.01`, `boundary = -0.5`
  - `velocity_smoothness = -1`: kkanbu-tunable knob (
streams 1-2 and 10-11)
- Training steps: ~15M
- Critic: symmetric (sees same obs as actor -- no
privileged info)
- Action-history length: kkanbu-available knob `
action_history_len  in  {0,1,2,3}` (all streams; obs/
network auto-adapt, no data regen). R5 default **k=1**
(S5); may revert to k=3 if PPO jerkiness returns (S13.6)
. Action-history length is a permitted knob; the batch
-0 anchor may differ in obs dim from later experiments.

### 12.2 privileged_rl

- Default training config: `src/navigation/configs/
Go1NavigationForestEnvV5/privileged_rl/training_config.
py`
- Two sub-paradigms available:
  - **Asymmetric actor-critic**: critic sees `
privileged_state` (combined
    state + normalized waypoints); actor sees LIDAR +
state only. There is
    no dedicated network file -- the asymmetry is
implemented env-side: the
    `strategy=privileged_rl` switch in
    `src/navigation/envs/go1_navigation_forest_env_v5.py
` (search
    `privileged_state`) exposes the extra state in `info
`, and the standard
    PPO trainer wires the critic to read it. Config:
    `src/navigation/configs/Go1NavigationForestEnvV5/
privileged_rl/training_config.py`.
  - **Privileged-obs distillation**: privileged-obs
teacher policy ->
    LIDAR-only student via behavior cloning / DAgger.
Pipeline lives in
    `src/navigation/imitation/bc_trainer/`.
- Reward weighting inherited from `pure_rl` defaults
unless the card overrides.
- Action-history length: `action_history_len` (k  in
{0,1,2,3}) tunable per S5 -- RL, obs/network auto-adapt,
 no data regen. Default **k=1**; may revert to k=3 per
```

S13.6. Action-history length is a permitted knob; the batch-0 anchor may differ in obs dim from later experiments.

12.3 offline_rl

- **Action space: 3D body-frame velocity `(v_x, v_y, v_yaw)` for *every* offline-RL algorithm** (IQL, CQL, TD3+BC) -- same as `pure_rl` / `pil_control` (S12 cross-stream anchor). **None output a trajectory.** The `OfflineReplayBuffer`'s `trajectory_mode` *decodes* waypoints -> velocity (`derive_velocities_from_trajectory`; `action_dim` is always 3), so loading a trajectory HDF5 only changes the data source / reward shaping -- never the policy output. A waypoint-action offline-RL policy would need a new 20D actor + Q/V and does not exist in this codebase.
- Available algorithms with trainers wired: **IQL, CQL**. TD3+BC has actor/critic network components in `src/navigation/networks/` but no end-to-end trainer in `src/navigation/training/offline_rl/trainers/`; AWR/AWAC are not implemented. A brainstormer slot proposing TD3+BC / AWR / AWAC must include the trainer wire-up in its experiment scope (and is subject to the S4.6 SLURM-ALGO cap if the wire-up keeps failing).
- Default hyperparameters per implemented algorithm:
  - IQL: `expectile = 0.7`, `temperature = 3.0`
  - CQL: `cql_alpha = 1.0`
- Demo dataset (shared with PIL streams): `data/imitation/forest/v5_clean.hdf5` (>=10k success-filtered episodes at delta=0.04, success-only -- see S8 prereq C1). The expert's safety margin around obstacles is set at scenario-gen time via `boundary_safety_margin = 0.5 m`, matching the value used in every V5 RL training config so the planned paths embed the same safety boundary the RL envs penalize. (There is no separate `obstacle_inflation` knob in this codebase -- A* receives the same clearance as the env's collision model.)
- Reward shaping: computed from observations at training time (`sparse_goal(+10) + dense_distance_shaping + collision_penalty`).
- Known constraint (R4 lesson, kept for context): expert-only demos limit what offline RL can learn beyond BC. Streams attempting offline RL should consider hybrid offline->online or data-augmentation variants if pure offline plateaus.
- Implementations: `src/navigation/training/offline_rl/trainers/{iql_trainer,cql_trainer}.py` (sharing `base_offline_rl_trainer.py`).
- Action-history length: `action_history_len` (k in {0,1,2,3}) tunable per S5 -- default **k=1** (H1 ablation). Datasets for any k <= 3 are **sliced from the canonical k=3 / 75D bundle (no re-collection)**; the net input layer auto-adapts. Action-history length is a permitted knob; the batch-0 anchor may differ in obs dim from later experiments.
- **Core-MLP deviation (batch-0 anchor only -- known TODO):** the `offline_rl` B0 anchor was trained as a `LIDARAttentionGaussianActor` with actor/critic width `[256,256]` + attention, NOT the `(512,256,128)` core. This is a pre-launch expedient (no time to re-train before launch). The `(512,256,128)` core-MLP constraint (S5) remains **non-negotiable for every batch-1+ autoresearch experiment**; a batch-1+ `offline_rl` slot should bring the policy onto the standard core (the attention block may be treated as an auxiliary module around it). See the `offline_rl-B0` card's `7_gap_and_future`.

12.4 pil_control (PIL -- velocity action)

- Action space: 3D velocity directly (Behavior Cloning on `(75D_obs, 3D_vel)` pairs)
- Network: `StudentPolicyNetwork` (75D -> core MLP -> 3D output) in `src/navigation/networks/student_policy_network.py`. (Older notes call this `VelocityPolicyNetwork`; the class is named `StudentPolicyNetwork` in the current source.)
- Default config: `src/navigation/configs/pil_forest_config.py`
- Demo dataset: `data/imitation/forest/v5_clean.hdf5`
- Default training hyperparameters:
  - 600k steps, batch 1024, learning rate 1e-3, cosine schedule, 1000 warmup steps
- Action-history length: `action_history_len` (k in {0,1,2,3}) tunable per S5 -- default **k=1** (H1 ablation: best success+smoothness for BC control). Datasets for any k <= 3 are **sliced from the canonical k=3 / 75D bundle (no re-collection)**; `StudentPolicyNetwork`'s input layer auto-adapts. Action-history length is a permitted knob; the batch-0 anchor may differ in obs dim from later experiments.

> Note: the batch-0 anchor is evaluated from a V3-env-trained checkpoint (66D), retained as a provisional number-to-beat; it will be replaced by a V5 6-density re-eval (see state/batch0_vpn_queue.md). The env immutable (S5) still binds all Round-5 experiments.

12.5 pil_trajectory (PIL -- waypoint action)

- Action space: 10 waypoints `[x1, y1, ..., x10, y10]`, `trajectory_scale = 5.0` (meters). `trajectory_scale` is the per-axis waypoint reach ceiling: the network output is `tanh(*) * trajectory_scale`, so each coordinate is bounded to +/-`trajectory_scale` m. It MUST be >= the reach of the (un-normalized, meter-valued) horizon-10 targets, or far waypoints clip -- do not lower it without re-deriving the target magnitudes. The training/eval invariant is pinned at 5.0 (`src/navigation/configs/pil_forest_trajectory_config.py`).
- Pure-pursuit conversion: waypoints -> 3D velocity command at runtime
  - `lookahead_index`: which waypoint Pure Pursuit chases (kkanbu-tunable)
- Network: `TrajectoryPolicyNetwork` (75D -> core MLP -> 20-dim waypoint output) in `src/navigation/networks/student_policy_network.py`
- Demo dataset: `data/imitation/forest/v5_clean_trajectory.hdf5` (converted from `v5_clean.hdf5` via `src/navigation/imitation/forest/convert_to_trajectory_dataset_from_positions.py --horizon 10`)
- Default training hyperparameters: 150k steps, `horizon = 10`
- Action-history length: `action_history_len` (k in {0,1,2,3}) tunable per S5 -- default **k=1**. Datasets for any k <= 3 are **sliced from the canonical k=3 / 75D bundle (no re-collection)**; `TrajectoryPolicyNetwork`'s input layer auto-adapts. Action-history length is a permitted knob; the batch-0 anchor may differ in obs dim from later experiments.

```
> Note: the batch-0 anchor is evaluated from a V3-env-
trained checkpoint (66D), retained as a provisional
number-to-beat; it will be replaced by a V5 6-density
re-eval (see state/batch0_vpn_queue.md). The env
immutable (S5) still binds all Round-5 experiments.

- **MAF (min_alignment_factor)**: low-level controller
parameter affecting
  forward-motion commitment. Historically `0.3` acted as
 an implicit symmetry-breaker
  for memoryless policies in dense forests. The 3-step
action history (V5 default)
  is the principled fix; MAF is no longer load-bearing
but remains a tunable.
- **Known issue (carried into R5)**: trajectory-mode
currently underperforms
  velocity-mode (~0% success rate at high densities)
despite the theoretical
  advantages of waypoint actions. This is one of the
open questions Round 5
  is structured to make progress on -- `pil_trajectory`
slots should be
  designed with this finding as the context, not a
presumed property.

### 12.6 d06-d09 (kkanbu-led exploration)

- **d06-d07**: combination / improvement of the
paradigm streams.
  Cross-paradigm hybrids (e.g., offline->online),
architectural extensions on
  top of a paradigm result, or transfer of a paradigm
mechanism to another
  paradigm. Kkanbu picks freely. Because streams run in
 parallel (S4.2),
  batch-1 designs from paradigm *priors* (blind to
siblings); batches 2-4
  fold in whatever sibling results have completed.
- **d08-d09**: totally new ideas outside the 5 paradigm
 streams.
  Candidate territories: world models (DreamerV3 family)
, transformer policy
  backbones, generative obstacle prediction as
auxiliary task,
  episode-50 universal-failure mitigations. d08-d09
slots
  are **anchorless** (`card.anchor_reference = null`);
the bar is the
  card's `expected_falsification` clause, not a paradigm
 anchor. The
  mechanical cratered gate of S4.4 governs seed-2+3
launch -- including
  the hypothesis falsified at single-seed level clause
that lets a
  paradigm-discovery slot stop at seed 1 without ad-hoc
 orchestrator
  discretion.

### 12.7 d10-d11 (tuning within constraint)

- Per S5 Tuning defined by negation:
  No new losses, no new paradigms, no architectural
changes. Network core
  immutable. Any other hyperparameter is fair.
- **Batch-1** tunes a **fixed baseline paradigm
assigned upfront** by the
  brainstormer (a runnable default config in `src/
navigation`, e.g.
  `pure_rl/training_config.py` or the IQL CLI defaults)
-- so d10/d11 run in
  parallel from t=0 without waiting for any experiment
(ADR 0012).
- **Later batches re-target the current champion** =
the highest-composite
  `complete` card across all streams at brainstorm time
(read from the
  maintainer's `index.md`). Tuning still touches only
hyperparameters, so any
  champion -- including a d08/d09 exotic -- is a legal
target (grind its
  hyperparameters, not its architecture). Streams 10
and 11 may share a
  target or diverge.
- Eval pipeline immutability applies here too (S5) --
d10-d11 has no
  carveout to fix eval bugs locally. Suspected eval
bugs halt the loop
  per S8 (escalates to orchestrator-as-human for an
autoresearch restart).

### 12.8 Architecture conventions (apply across all
paradigms)

The LIDAR architecture family the project considers,
with the structural
inductive biases each respects (per program.md S5: only
 auxiliary modules
around the immutable core MLP are free):

- **MLP** (flat 64-dim input): respects no LIDAR
symmetries; baseline.
- **Circular CNN**: shift-equivariant on the ring (beam
 63 <-> beam 0).
- **PointNet** (set view): permutation-invariant;
density-invariant by construction.
- **Attention** (LIDAR-Attention): empirically
strongest baseline at narrow
  density gaps; relies on learned bias rather than
structural invariance.
- **PointNet++**: hierarchical set abstraction --
partially in scope if
  d08-d09 explores it.

Specifying a non-default LIDAR architecture is a kkanbu-
available design
decision in any stream's slot proposal (assuming it
doesn't violate the
core-MLP immutability -- auxiliary perception heads
around the core are free).

---

## 13. Project overview

> Distilled from R4's `about_playground_test.md`.
> This section is what kkanbu and any round5b vanilla-
LLM consumer need
> to ground themselves in the codebase's domain,
philosophy, and
> conventions. R4-specific numbers and named leaders
are intentionally
> omitted per S3 -- R5 establishes its own anchors (
human-seeded batch 0 where provided, else batch-1).

### 13.1 The one question the repo is trying to answer

> **How do we get a neural-network policy to do the
right thing in
> situations it has never seen, instead of memorizing
the training
> conditions?**

The hard part of training a policy is not getting it to
 perform on its
training distribution -- neural networks overfit by
default. The hard
part is producing a policy whose behavior makes sense *
outside* that
distribution. The project's word for this is **
generalizability**.

This is not a benchmark-chasing project. The goal is
not the highest
absolute number. The goal is **the smallest gap between
 in-distribution
and out-of-distribution performance**, on a
distribution shift that
actually matters. Everything else (observation design,
network
architecture, training paradigm, evaluation weighting)
is in service of
```

answering that question with rigor.

13.2 The concrete testbed

- **Robot**: **Unitree Go1**, a quadruped with 12 joint motors
  (hip / knee / ankle x 4 legs). Simulated in **MuJoCo** via the
  MuJoCo Playground / Brax JAX framework.
- **Perception**: **LIDAR** only -- a ring of 64 distance beams (each
  normalized 0=close, 1=far). No camera, no depth image. Sim-to-real
  is explicitly out of scope (S5).
- **Goal**: navigate from a start pose to a goal pose in a room full
  of static cylindrical trees. Trees don't move; no dynamic
  obstacles.
- **Room**: 30 x 30 m (immutable in R5).
- **Hierarchy**: the policy is split into two layers.
  - **Locomotion layer**: takes a 3D velocity command, outputs 12 joint
    torques. Pre-trained (~100M PPO steps on flat terrain) and frozen.
    No navigation experiment touches it.
  - **Navigation layer**: reads V5 75D observation
    (2D goal + 64D LIDAR + 9D action history), outputs either a 3D
    velocity (`pil_control`, RL paradigms) or a 10-waypoint trajectory
    (`pil_trajectory`). **This is the only layer the project trains.**

Walking is treated as a solved engineering problem. All research
capacity focuses on the decision-making layer -- route choices -- where
generalization actually matters.

13.3 Why density?

Density = trees per square metre. Low density = sparse forest =
easy. High density = packed = hard.

The project trains policies at one fixed density (**delta = 0.04
trees/m^2**) and evaluates across **six densities**
`{0.01, 0.02, 0.04, 0.08, 0.12, 0.16}` -- a 16x range from sparsest to
densest, with training at the middle.

Density was chosen as the OOD axis after also studying obstacle shape
and size. Shape and size drops were 4-9 percentage points of success
at worst. Density caused a much larger drop. Studying a small effect
teaches nothing; studying the effect that dominates is where
understanding becomes possible.

The composite score (S2) weights high densities more heavily
(2.5x at delta=0.16 vs 1.0x at training). This is **deliberate metric
design**: it says I care more about surviving at 4x training density
than about perfection at training density. The metric is itself an
expression of the research stance.

13.4 V5 observation: the result of subtractive design

The navigation policy's input has gone through several generations of
simplification (V1 -> V5). Each revision *removed* features rather
than adding them:

- **V1**: absolute `(x, y)` position in the room, dense goal encoding.
  Memorized coordinates; failed when room dimensions changed.
- **V2**: removed absolute position; kept rich goal encoding.
- **V3**: reduced goal encoding to 2D (distance + heading), normalized
  by room diagonal. 66D total.
- **V4**: minor refinement.
- **V5** (the R5 standard): 2D goal + 64D LIDAR + `3k`D action history.
  Action history was added specifically to fix the perceptual-symmetry /
  decision-stability problem (S13.6); the original fix used 3 timesteps
  (75D). The H1 ablation later found a **single**
  most-recent action (k=1, 69D) gives the best success/smoothness tradeoff
  on BC, so **R5 defaults to k=1** while k=3 remains available (S5).

The philosophy: **richer observations invite more overfitting**. If a
dimension isn't earning its keep as a scale-invariant, generalization-friendly signal, it should be removed.

A live tension the project takes seriously: a cleaner observation is
not automatically a more generalizable one. Some OOD axes are
better-served by *some* redundancy in the goal signal. The subtractive
trajectory is not religion; it's a strong prior with documented
exceptions.

13.5 The space of methods (cross-paradigm comparison)

The project is deliberately **multi-paradigm**. Rather than tuning one
method to death, R5 compares whole families on the same density-sweep:

- **Imitation Learning (PIL)** -- a privileged-information expert
  (A* planner + Pure Pursuit) generates demos; a LIDAR-only student
  is trained to reproduce them. Two sub-variants: PIL-Control (3D
  velocity output) and PIL-Trajectory (10-waypoint output + Pure
  Pursuit controller). See S12.4-12.5 for project conventions.
- **Offline Reinforcement Learning** -- same expert data, fed through
  IQL / CQL / TD3+BC / AWR. Known limitation: expert-only data
  constrains what value-based methods can learn beyond BC (S12.3).
- **Online Reinforcement Learning (PPO)** -- no demonstrations; the
  policy learns by trying, colliding, getting punished. Symmetric or
  asymmetric variants (privileged critic). See S12.1-12.2.

Architectures for the LIDAR perception layer (around the immutable
core MLP) include MLP baseline, Circular CNN, PointNet (set view,
density-invariant by construction), PointNet++, and Attention. See
S12.8.

Why multi-paradigm: any single paradigm's OOD result could be a quirk
of its loss surface. The cross-paradigm pattern is what survives. If

all paradigms converge to roughly the same OOD ceiling, that ceiling is a property of the task formulation -- and finding it (then breaking it by changing the formulation rather than the method) is one of the highest-value possible Round 5 outcomes.

13.6 The MAF / oscillation case study (carried into R5 as context)

This is the project's worked example of **how diagnosis beats variant sweeping**. R5 inherits the diagnosis as context; the underlying question (decision stability under perceptual symmetry) remains live.

Setup: PIL-Trajectory outputs waypoints; Pure Pursuit converts them to velocity. Pure Pursuit has a parameter MAF (`min_alignment_factor`) -- the minimum forward speed the robot maintains regardless of heading misalignment.

Failure mode at MAF=0 in dense forests: the robot stops to rotate when facing away from the lookahead point, but its LIDAR doesn't change while stationary, so the policy re-computes the same ambiguous decision and flips. Result: vyaw oscillates between extremes, no forward progress, timeout.

Three interacting causes:
1. In dense forests, left-path and right-path are often **symmetric** -- equally viable from the agent's perspective.
2. The policy is **memoryless** (V3 obs had no history). Every timestep is an independent decision.
3. MAF=0 lets the robot stop. With no forward motion, the LIDAR reading barely changes, so the decision input doesn't change, so the decision flips with noise.

MAF=0.3 had previously hidden the bug by forcing forward drift, which broke symmetry by changing the robot's position. The forced motion was acting as an **implicit commitment mechanism**, not just a smoothness prior.

The fix that worked: 3-step action history appended to the observation (V3 66D -> V5 75D). The policy now sees its last 3 velocity commands, can detect I've been turning left, and commits to its prior choice rather than flipping. This is the principled fix; MAF is no longer load-bearing but remains a tunable. (The *amount* of history is itself a knob -- the later H1 ablation found a single most-recent action `k=1` recovers the smoothing effect at the best success/smoothness tradeoff, so R5 defaults to k=1 (S5); the 3-step form remains the fallback if PPO jerkiness returns.)

Why this matters for R5: the V5 standard inherits this fix. Slot proposals (especially in `pil_trajectory`, `d06`-`d09`) should treat perceptual symmetry resolution as a solved-but-instrumented problem -- not a presumed property to ignore, but also not the central mechanism to re-investigate without specific reason.

13.7 The researcher's stance

Six principles that thread through the codebase:

1. **Overfitting is the null hypothesis.** Generalization is work, not default. The metric weighting (S2) is calibrated to punish OOD failure more than it rewards in-distribution success.

2. **Remove features before adding them.** Subtractive observation design (V1->V5). A dimension that doesn't do scale-invariant work is a liability.
3. **Prefer inductive biases symmetric with the world.** Architectures like PointNet and circular CNN encode structural priors that reflect the world's actual symmetries (LIDAR rotational equivariance, density invariance) rather than learning them.
4. **A failure mode well-diagnosed > a marginal gain well-tuned.** The MAF/oscillation story (S13.6) is the template. A hypothesis, a measurement, a root cause, a minimal fix, a quantitative improvement.
5. **Be honest about what you don't understand.** Mechanism claims for *why* PointNet beats MLP, *why* attention helps, *why* trajectory underperforms velocity, remain hypothesized. Open edges, not solved problems.
6. **Rigor comes from volume and structure.** Multi-seed protocol, composite-score weighting, paradigm-self-calibrating anchors, stratified percentile bootstrap CIs (S2, S4.4-4.5). A claim is not a claim until the evidence is at this scale.

13.8 What is NOT in scope

(Restated from S5 for the kkanbu / round5b reader who lands here first:)

- **Sim-to-real transfer.** Studied; depth cameras would 10x training cost with no obvious generalization benefit. Deferred. LIDAR is the committed input modality.
- **Dynamic obstacles.** Trees are static.
- **Locomotion research.** Walking layer frozen.
- **Leaderboard hunting.** Round 5 would rather understand *why* a method works than push it 0.5% higher.