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, 2021a; 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

[*]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.

best configuration. The new system adds **kkanbu**, an oracle 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. 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 loop that falsifies its own champion's operating point.

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 paradigms on a forest-navigation benchmark: PIL Control (Section 3.3), PIL Trajectory (Section 3.4), IQL (Section 3.5), TD3+BC (Section 3.6), and PPO (Section 3.7).

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 obstacles 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; ac-

tions are velocity commands at $50\,\mathrm{Hz}$. Full environment, observation, and action specifications are in Appendix C.

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. 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.

3.6. TD3+BC

TD3+BC (Fujimoto & Gu, 2021b) adds one weighted BC term to the TD3 (Fujimoto et al., 2018) actor objective (Equation 10) with $\lambda = \alpha/\mathbb{E}_{\mathcal{D}}|Q|$, $\alpha = 2.5$, plus per-state

observation normalisation. One extra hyperparameter, no ensemble, no behaviour model, and it matches CQL on D4RL at roughly half the wall-clock (Tarasov et al., 2023). Pairing it with IQL places the comparison at opposite poles of the offline-RL design space (*no policy improvement at OOD points* vs. *constrained improvement everywhere*), so a gap localises whether a problem is OOD-action evaluation or general policy degradation. We omit CQL (sensitive to its conservatism coefficient on narrow expert data), BCQ/AWAC (extra behaviour model), and Decision Transformer (suited to sparse-reward / human demos (Bhargava et al., 2024)). Architecture, dataset, reward, and 75-D observation are shared with IQL.

3.7. Proximal Policy Optimization (PPO)

We evaluate PPO (Schulman et al., 2017) as the online-RL baseline. In the standard ("pure RL") 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, and stuck-detection penalties.

3.8. 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 RL 85.83, pure RL 83.0) and degrade most gracefully into the OOD region. PIL control follows at 78.72. Offline RL (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 offline RL are single-seed, and that caveat travels with every anchor delta.

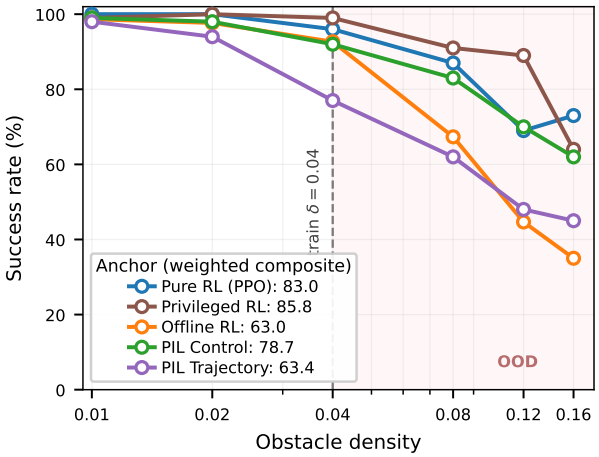

Figure 1. Per-paradigm anchor profiles under obstacle-density shift. Each curve is the stream's anchor: the human-seeded baseline (or default-configuration first batch) that every later experiment in that stream is scored against. The legend gives each anchor's OOD-weighted composite. Training density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

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.

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.

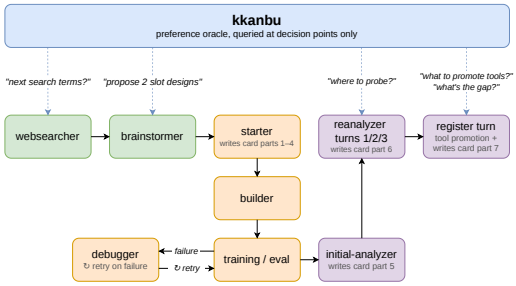

Figure 2. One stream-batch of the AI Scientist loop. **kkanbu** (blue) is the only opinion-holder, consulted at four labeled decision points. Planning agents (green) decide *what to do*; execution agents (orange) carry it out; analysis agents (purple) write the card and register tools.

4.1. 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 rules 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. 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.

4.4. 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 4.3 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 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: 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 falsifies its own champion's operating point

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. The loop then declined to trust its own peak. The operating point had been chosen as the best of five arms scored on the evaluation densities, so a follow-up grid varied the gate and the cap jointly; its first seed reproduced the original cell bit-for-bit and falsified the chosen point as an artifact of the unswept cap, with the optimal gate drifting as the cap grows. The falsification machinery of Section 4.1 here operates on the system's own best number.

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

We ran the loop of Section 4 twice under the design of Section 4.4: 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 rules in either arm: LiDAR-only inference, frozen locomotion, 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 in the with-kkanbu arm against four in the without-kkanbu arm; under a conservative Student-t criterion the count reverses to one against three. 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), 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, a categorical action head for the control paradigm, and the control-versus-treatment framing whose *control arm* became the trained co-leader.

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

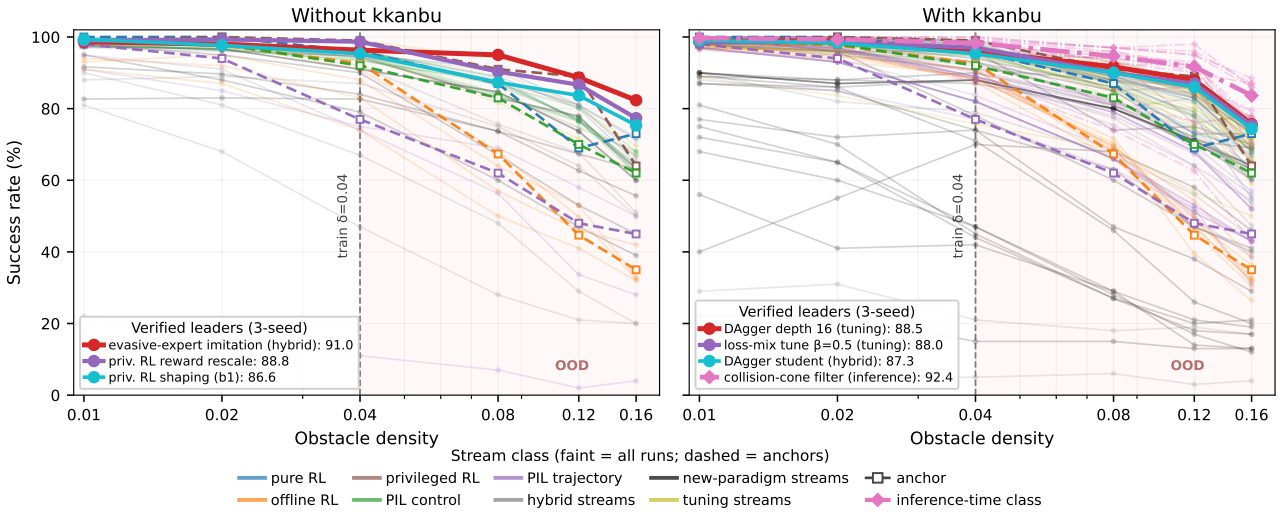

Figure 3. Density vs. success rate by ablation arm. Faint lines are all evaluated configurations (colour-coded by stream class); thick lines are each arm's three strongest verified 3-seed trained results; dash-dot curves in the with-kkanbu panel are the collision-cone filter family, an inference-time technique on frozen weights reported as a separate class (thick: its verified 3-seed operating point, $+4.89$ within-model; faint: single-seed cells of its operating-point grid). 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.

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. 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 §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 falsifier grid aimed at its own champion (§5.3), and the cross-stream memory that stops the loop from rebuying 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

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).

Arm	Entry	Class	Composite	Δ
Without kkanbu	imitation student on regenerated evasive-expert demos (hybrid stream, batch 3)	trained	90.96 ± 1.52	
	privileged RL, goal-progress reward rescale (batch 2)	trained	88.80 (seeds 87.2/89.7/89.4)	
	privileged RL, potential-based shaping (batch 1)	trained	86.56 ± 2.42	
With kkanbu	DAgger correction-depth sweep, 16 rounds (tuning stream, batch 4)†	trained	88.48 ± 2.44	
	privileged-RL loss-mix tune, β=0.5 (tuning stream, batch 2)	trained	87.96 ± 0.83	
	DAgger distillation student, control arm (hybrid stream, batch 1)	trained	87.28 ± 0.56	
	collision-cone filter on the frozen DAgger student†	inference-time	92.43 ± 0.69	$+4.89$

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 benchmark 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 champion whose own operating point the loop went on to falsify. 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.

Bhargava, P., Chitnis, R., Geramifard, A., Sodhani, S., and Zhang, A. When should we prefer decision transformers for offline reinforcement learning? In *International Conference on Learning Representations (ICLR)*, 2024.

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*, 2021a.

Fujimoto, S. and Gu, S. S. A minimalist approach to offline reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pp. 20132–20145, 2021b.

Fujimoto, S., Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning (ICML)*, 2018.

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.

Hong, J., Dragan, A., and Levine, S. Offline rl with observation histories: Analyzing and improving sample complexity. *arXiv preprint arXiv:2310.20663*, 2023. URL https://arxiv.org/abs/2310.20663.

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.

Tarasov, D., Kurenkov, V., Nikulin, A., and Kolesnikov, S. Revisiting the minimalist approach to offline reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

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 reanalyzer'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 4.4 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$9.4M transitions

Reward function. The reward used during offline-RL training (shared by IQL and TD3+BC) 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. TD3+BC

Table 6 presents hyperparameters for TD3+BC (Fujimoto & Gu, 2021b), the second offline-RL algorithm we evaluate. The actor objective adds a single weighted BC term to the TD3 (Fujimoto et al., 2018) objective:

$$\mathcal{L}_\pi(\phi) = -\mathbb{E}_{\mathcal{D}}[\lambda Q_\theta(s, \pi_\phi(s))] + \mathbb{E}_{\mathcal{D}}\left[\|\pi_\phi(s) - a\|^2\right]. \tag{10}$$

The same hand-shaped reward as IQL (Section B.4) is used; the actor head and dataset vary by experiment.

Table 6. TD3+BC hyperparameters. The BC weight α is annealed linearly from 2.5 to 0.5 over the first 100 K training steps and held at 0.5 thereafter, following the standard TD3+BC recipe. Critic target update uses Polyak averaging.

Parameter	Value
Actor encoder	LiDAR-Attention (same as PIL)
Actor head	MLP [512, 256, 128]
Critic encoder	LiDAR-Attention (twin Q)
Critic head	MLP [256, 256, 256]
Activation	Swish (encoder) / ReLU (heads)
BC weight α (start)	2.5
BC weight α (end)	0.5
α anneal schedule	linear over first 100 K steps
Actor learning rate	3×10^{-4}
Critic learning rate	3×10^{-4}
Optimizer	Adam (Kingma & Ba, 2015)
Discount γ	0.99
Target update rate τ	0.005
Policy noise	0.2
Noise clip	0.5
Batch size	256
Training steps	1,000,000
Action history	none ((Hong et al., 2023))

Configuration that varies by experiment. The actor output head and the training dataset are the only variables across the TD3+BC experiments; everything else in Table 6 is held constant.

B.6. PPO

Table 7 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.7. Pure Pursuit Controller

Table 8 documents the Pure Pursuit controller (Coulter, 1992) parameters used for trajectory-based methods.

C. Implementation Details

C.1. Environment Specification

Table 9 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.

C.2. Observation Space

Table 10 details the observation space components (75D total). The encoding deliberately omits absolute position so the policy is scale-invariant to room size.

C.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.

Table 7. 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 RL only)	varies by experiment

C.4. Computational Resources

All experiments are conducted on NVIDIA A100 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.

Table 8. 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 9. 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	$[-1.2,\ 1.2]\,\mathrm{rad/s}$

Table 10. 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 $(v_x, v_y, v_{\mathrm{yaw}})$