An AI Scientist Guided by Encoded Human Research Preferences: A Case Study in Quadruped Neural Navigation

Anonymous Authors[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. Across multiple experiment batches, the system produced mechanism-level findings that redirected the project, including identifying a previously-overlooked bottleneck, recovering an earlier-retired paradigm, and falsifying one of its own hypotheses.

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.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

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 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 best configuration. The new system adds **kkanbu**, a persistent advisor that stores evolving hypotheses and open questions across batches, 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 and compare privileged imitation learning, waypoint-based imitation learning with Pure Pursuit control, offline RL, and online RL. Across batches, the system produced both strong policies and mechanism-level findings. It revisited offline RL after identifying that earlier failures came from the experimental setup rather than the paradigm, and it rejected one of its own mechanistic hypotheses through targeted follow-ups. Under multi-density evaluation, the strongest policies came from online RL.

Our contributions are threefold. First, we present a comparative study of quadruped navigation paradigms under obstacle-density shift, spanning imitation learning, offline RL, and online RL within a shared hierarchical control framework. Second, we introduce an AI Scientist workflow that combines persistent cross-batch hypothesis tracking through **kkanbu** with a structured multi-agent experimentation loop. Third, we show this changes the character of autonomous experimentation: compared with an earlier version of our own loop, the new system explores broader algorithmic directions and surfaces mechanism-level explanations rather than only optimizing benchmark scores.

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 via **kkanbu**, structured experiment records, and a persistent multi-agent workflow. We test whether this structure broadens algorithmic exploration and yields more mechanism-level findings.

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

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 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 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 $\sim 8.5 \times 10^6$ 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.

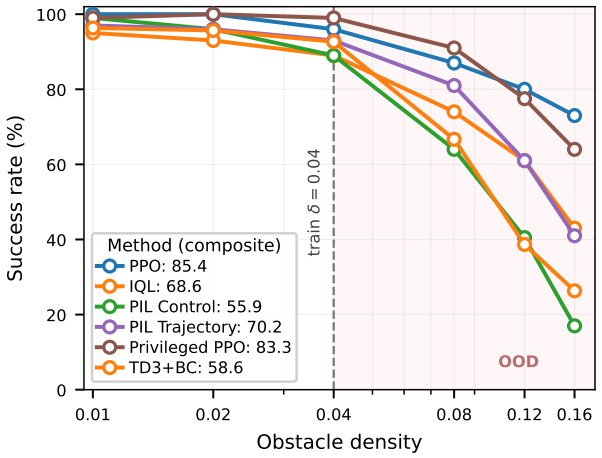

Figure 1. Per-paradigm baselines under obstacle-density shift. Each curve is the cleanest default-configuration run for one paradigm in §3; legend shows composite scores. Training density $\delta = 0.04$ is dashed; OOD region ($\delta > 0.04$) is shaded.

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. Baseline density-generalization profile

Before describing the AI Scientist system, we summarise how the method families behave at default configuration. Figure 1 plots density-vs-success-rate for the cleanest default run of each paradigm: PPO, Privileged PPO, IQL, TD3+BC, PIL Control, and PIL Trajectory. Each was trained at $\delta = 0.04$ and evaluated across the six-density sweep $\{0.01, 0.02, 0.04, 0.08, 0.12, 0.16\}$.

The default profile already exposes much of the design space. PPO and Privileged PPO are the only paradigms holding above 70% at every density and above 60% at the hardest OOD point $\delta = 0.16$; their composites of 85.4 and 83.3 set the online-RL floor. The offline-RL baselines cluster well below (IQL 68.6, TD3+BC 58.6) and collapse hardest at $\delta \geq 0.12$, a gap that §5.2 will show is structural (forward-only actor head plus forward-only demos), not paradigm-fundamental. The two PIL baselines (PIL Trajectory 70.2, PIL Control 55.9) saturate near 95% at training density but lose at least half their success rate by $\delta = 0.16$; PIL Trajectory's collapse will turn out to be a Pure-Pursuit interface artefact rather than a representation failure.

These curves are not meant as comparable systems (they differ in supervision, dataset, and what the actor head can emit) but as the search space the loop operates on. A density-generalisation result is interesting only against this profile: each paradigm's default trace is the null hypothesis that the findings in §5 test against.

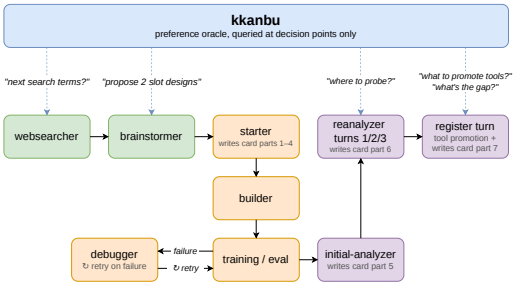

Figure 2. One stream-batch of the AI Scientist loop. **kkanbu** (blue) is the only opinion-holder, consulted at four labeled decision points. Planning agents (green) decide *what to do*; execution agents (orange) carry it out; analysis agents (purple) write the card and register tools.

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 parallel research *streams*, one per method family of Section 3, each advancing independently from its baseline. From batch 1 onward, each batch produces two parallel *slots*; a slot is a single experiment with its own card.

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 slot, that carries each iteration's knowledge across batches.

A card has three stages. **Planning.** *Motivation* (part 2) connects the slot to a prior finding or open question; *expected result* (part 4) states the prediction whose falsification would be informative. 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 slot's git worktree, build notes, and the evaluation score from the per-density sweep (part 5). **Analysis.** The reanalyzer 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) coreside, 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 two distinct slot proposals.

Experiment. The two slots run in parallel. For each, 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. 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 slot 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. An `experiment-reanalyzer` 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, reasoning patterns, and priors drawn from the literature. 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 two slot proposals verbatim into the cards. The `experiment-reanalyzer`, 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 reanalyzer 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 slot most likely to score well, a reanalyzer 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.

5. Findings

We present three findings the loop produced over its experiment batches. Each is selected because its lineage (the chain of earlier experiments whose card part 7 constrained the next batch) made the finding legible. Each subsection states the headline, the structural argument, and why a leaderboard-driven loop could not have produced it; per-batch operational detail (formulas, intermediate diagnostics, exact statistics) is in Appendix G.

5.1. Online-RL: a four-batch arc with subtractive wins and a falsified self-diagnosis

The online-RL stream went from a catastrophic **17.1** composite at batch 1 to a calibrated **96.28 IQM,** $\sigma = 1.62$ (3 seeds) at batch 4. Three features matter for the argument. First, three of four positive interventions across batches 2 and 3 were *subtractive*: the loop removed a proximity-penalty reward term that crashed batch 1, fixed a checkpoint-restoration bug, applied a bounded one-sided penalty against a residual backward-spinning attractor, and pruned a dead multi-hundred-million-step hold phase.

Second, batch 4 *falsified its own preceding mechanism diagnosis*. Batch 3's card part 7 had prescribed a TTC-based reward intervention against a hypothesised "late hard swerve" kinematic ceiling. 3-seed reanalysis from batch 4 found the actual failure was *commit-and-freeze* 3–5 m before impact, structurally invisible to a TTC reward defined over the last 2 seconds. Card part 7 retired the entire approach-phase-reward family and named action-space expansion as the next paradigm pivot.

Third, a sibling batch-4 slot re-ran the batch-3 recipe on three seeds with *no code changes*, producing IQM 96.28 (CI $[95.19, 97.28]$) and demoting the apparent batch-3 leader of 97.30 (single seed) to a high-side draw at the upper CI boundary. The leaderboard reading was now noise-bounded.

Why a leaderboard-driven loop could not have produced this. A score-maximiser would have discarded "density curriculum" at batch 1's 17.1, held at batch 2's 92.7 (no metric surfaces backward-spin), stopped at batch 3's 97.30, and never run the batch-4 TTC slot (-0.32 pp "regression") or the calibration slot (zero-code, equivalent score, wider error bars). The two most informative interventions of the arc are the two a score-maximiser would skip.

5.2. Offline-RL resurrection: retirement reversed in one batch

An offline-RL slot *formally retired* the paradigm at the end of batch 3 with a clean mechanistic explanation. The actor head $v_x = 0.75 \tanh(\cdot) + 0.75$ structurally clipped v_x to forward-only, and the A^* expert dataset was 100% forward-only. The constraints *compound*: the architecture cannot represent what the data lacks, and the data lacks what the architecture cannot represent. Card part 7 logged the retirement reason as "forward-only demos $\times$ forward-only head" rather than "offline-RL paradigm failure," preserving the resolving experiment for the next batch.

The next batch lifted both constraints at once: a symmetric actor head and a swap to a success-filtered rollout of the online-RL leader, originally collected one batch earlier by a PIL-Trajectory distillation slot for a *different* purpose. Composite **77.13** (single seed, $+18.48$ pp over parent; $\delta = 0.16$ alone gained $+37.67$ pp).

Reanalysis surfaced a finding orthogonal to the score. The teacher PPO is bang-bang bimodal at every density, including 51% backward at $\delta = 0.01$; the student rejects those samples at deployment and goes forward-only at $\delta = 0.01$ while emitting bidirectional bang-bang at $\delta \geq 0.02$. The same critic delivers $+37.67$ pp at $\delta = 0.16$ and -7.66 pp at $\delta = 0.01$: one mechanism, two opposite outcomes, density-conditional—LIDAR- thresholded mode selection that exists nowhere in the teacher's behaviour.

Why a leaderboard-driven loop could not have produced this. The retired score (58.65) was below the IQL reference, and the *retirement was scientifically correct* given the data; a score-maximiser would have closed the slot. Resurrection required three things a leaderboard cannot consume: the part-7 mechanism statement naming compounding constraints; a tool-output correlation between Pure-Pursuit zero-collapse waypoints and collision rate (computed during a sibling PIL-Trajectory slot); and *cross-paradigm dataset reuse*, a teacher-rollout built for PIL-Trajectory and routed into offline-RL.

Caveat. The slot is single-seed; a sibling 4-seed characterisation measured $\sigma_{\text{offline}} \approx 4.76$ pp, so the rank of 77.13 is uncertain at ± 9 pp (2σ). The *direction* of the $+18.48$ pp jump is structural; the rank is not. Per-batch detail in Appendices E–F.

5.3. A previously-overlooked bottleneck: single-seed scores were a lottery

The loop discovered its own previously-overlooked bottleneck. The single-seed composite estimates used as the comparison metric for the prior three batches were an unreliable lottery; every ranking decision the loop had made was unwittingly noise-bounded. A batch-3 calibration probe measured per-density variance large enough that two-thirds of the loop's prior comparisons sat within the noise band. kkanbu computed an explicit minimum-detectable effect, mandated a 3-seed replication of the perceived online-RL leader (which became the calibrated 96.28 IQM of §5.1), and the resulting argument propagated into the program description as a *1+2 staged-seed protocol*: seed 1 as debug; seeds 2 and 3 promoted only if seed 1 lands within $\sim$5 pp of the calibrated benchmark.

The protocol caught the same lottery prospectively. A later batch added a post-network action cap. Seed 1 produced an apparent $+1.4$ pp gain with a confident "dual-action" mechanism diagnosis from reanalysis turn 2; seeds 2 and 3, promoted under the threshold, yielded a 3-seed IQM *below* the benchmark. The seed-1 leader was an outlier; the diagnosis was built on a single sample.

A zero-cost ablation revealed bidirectionality. The next batch toggled the cap at eval time on existing checkpoints (no retraining). The cap is *bidirectional across seeds*: it precipitates heading-lock on one (a cap-firing-vs-lock-onset alignment diagnostic finds the cap fires before lock onset, causal) and suppresses lock on another. Same failure mode, opposite cap effect—a per-step state-based gate cannot distinguish "this policy will lock without intervention" from "this policy will lock if I intervene."

Why a leaderboard-driven loop could not have produced this. The seed-1 "leader" *was* a higher number; a leaderboard loop would have promoted it. The 1+2 protocol exists only because the calibration slot forced the argument onto kkanbu's reasoning, which propagated into the program description the brainstormer reads at every batch. The runtime-toggle slot is structurally unscoreable: the output is a bidirectionality finding, not a ranked policy.

6. Without kkanbu vs. With kkanbu: A Same-Codebase Ablation

We ran an earlier version of the loop on the same problem and codebase. That loop lacked kkanbu and the dedicated subagents: its coordinator was a single Claude Code session reading and writing Python state files, with parallelism only via SLURM dependencies. We then ran the system of §4 for 61 more experiments. The substrate (environment, observation, action set, scoring rule, locomotion controller) was identical across all 111 runs; the only difference is whether kkanbu and the subagent flow were present. Call the conditions *Without kkanbu* ($n = 50$) and *With kkanbu* ($n = 61$).

Without-kkanbu success reasons. All three top entries are PPO variants, and all three are reward- or training-distribution tweaks rather than mechanism-targeted interventions. The leader (91.9) wins by a single change—collision penalty $-10 \rightarrow -50$. The mechanism is recoverable retrospectively: -50 makes the policy decisive rather than cautious, eliminating *timeouts* (TO@$\delta = 0.12$: $11\% \rightarrow 0\%$) without reducing *collisions* (CR@$\delta = 0.12$: $8\% \rightarrow 7\%$), a counter-intuitive finding that no Without-kkanbu card recorded at the time. The second-place run stacks -50 with training density 0.08; the third uses higher training density alone. None has a contemporaneous reanalysis or mechanism diagnosis: the loop's next-experiment signal was leaderboard rank, so once the -50 penalty won, the remaining budget went to variants of the leader rather than to asking what mechanism the leader had identified. The OOD endpoint $\delta = 0.16$ peaks at $\approx 67\%$.

With-kkanbu success reasons. The leader at composite 97.50 adds a forward-KL distillation loss against an earlier calibrated PPO leader, *spatially gated* to fire only when the student's nearest LIDAR return is in $[1.5, 5.0]$ m—the early-commit window an earlier batch's reanalysis had identified. The gate produces a geometry-conditional policy that matches the teacher in-gate and develops novel avoid-flip behaviour out-of-gate; the mechanism prediction held, and this is the only With-kkanbu result clearly above the calibrated benchmark of §5.3. The second-place run (96.98) adds a TTC-force reward and a reverse-KL anchor against the same teacher; the TTC term itself produced *zero* behavioural change at $\delta = 0.16$, but the reverse-KL anchor reduced cross-seed spread without moving the mean, and reanalysis falsified the parent's failure-mode story ("late-swerve at saturated v_x") in favour of *commit-and-freeze* 3–5 m before impact. The third-place run (96.78) extends the density curriculum two phases further ($\delta = 0.10 \rightarrow 0.14$) under a "training-distribution gap" hypothesis; the *hypothesis was falsified* ($\delta = 0.16$ stays at 93%, same as the parent) but the run remains a tight statistical tie with the leader at significantly tighter spread than the calibrated benchmark. The OOD endpoint $\delta = 0.16$ peaks at $\approx 93\%$, a $\approx +26$ pp absolute lift over the Without-kkanbu condition.

6.1. Where the score gains came from

Figure 3 and Table 1 give two views of the same data. The score arc is large ($91.9 \rightarrow 97.50$ IQM), but *the kind of win* differs more than the numbers do. We summarise each condition's top three by attributable mechanism, not absolute number.

Without kkanbu: leaderboard increments, no reanalysis hook. All three leaders are PPO variants. The leader at composite 91.9 is one change: collision penalty $-10 \rightarrow -50$. The mechanism is recoverable retrospectively: -50 makes the policy decisive rather than cautious, eliminating *timeouts* (TO@$\delta = 0.12$: $11\% \rightarrow 0\%$) without reducing *collisions* (CR@$\delta = 0.12$: $8\% \rightarrow 7\%$). No card recorded that at the time. The second-place run stacks the same penalty change with training density 0.08; the third uses the higher density alone. After a PointNet variant led an earlier slot, the loop's next-action signal was "ranked second on the board," so the remaining budget went to PointNet capacity and regularisation tweaks (LIDAR augmentation, density branches, a higher-dimensional variant, mixup), none of which tested *why* the leader generalised better. A post-hoc retrospective names the failure mode plainly: *"the loop defaulted to running another variant of the current leader because the next-action signal came from the leaderboard, not from open questions."*

With kkanbu: every leader has a mechanism statement, and falsifications are recorded next to confirmations. The top run, at composite 97.50 IQM, adds a forward-KL distillation loss against the online-RL stream's

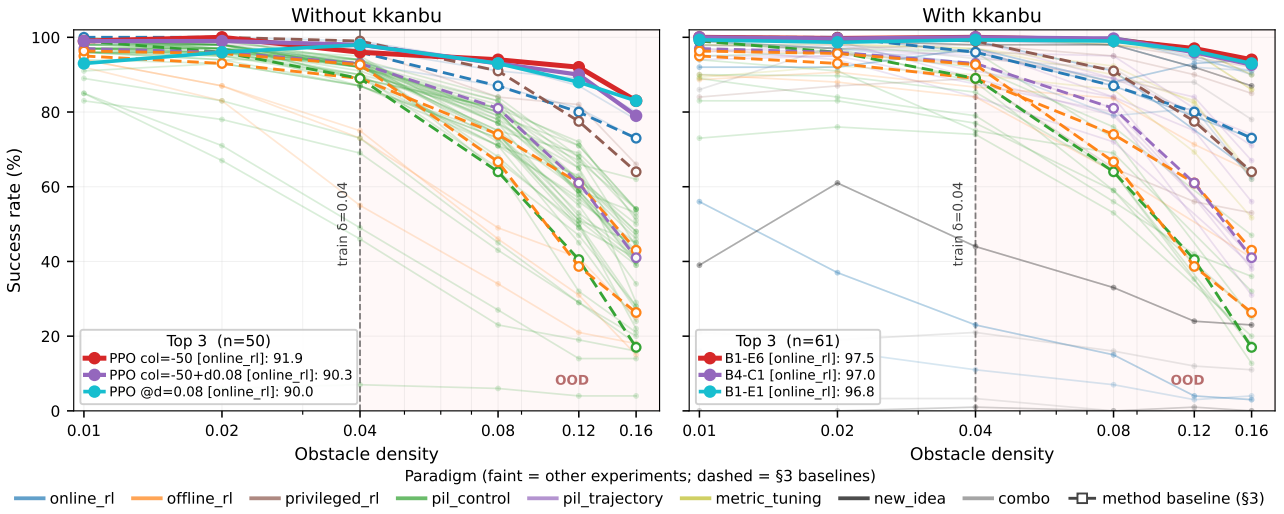

Figure 3. Density vs. success rate by condition. Faint lines are every completed experiment (paradigm-coded); thick lines are the top three by composite. Dashed white-fill curves are the per-paradigm baselines from Fig. 1. Training density $\delta = 0.04$ is dashed; OOD region ($\delta > 0.04$) is shaded. Without-kkanbu tops out at 91.9; With-kkanbu at 97.50 IQM.

Table 1. Top-3 experiments per condition, ranked by composite. Bracketed intervals are 95% bootstrap CIs; entries without intervals are single runs.

Condition	Recipe	Composite
Without kkanbu	PPO, collision penalty -50	91.9
	PPO, collision -50 + train $\delta = 0.08$	90.3
	PPO, train $\delta = 0.08$	90.0
With kkanbu	PPO + spatially-gated forward-KL anchor	97.50 [96.44, 98.56]
	PPO + TTC-force reward + reverse-KL anchor	96.98
	PPO + density curriculum extension to 0.14	96.78 [96.39, 97.00]

calibrated leader, *spatially gated* to fire only when the student's $d_{\min} \in [1.5, 5.0]$ m, the early-commit window flagged by an earlier batch's reanalysis. The gate yields a geometry-conditional policy that matches the teacher in-gate and develops novel avoid-flip behaviour out-of-gate; the pre-registered mechanism held. The second-place run (96.98 IQM, $\sigma = 0.49$) adds a TTC-force reward and a reverse-KL anchor against the same teacher. The TTC term gave *zero* behavioural change at $\delta = 0.16$, but the reverse-KL anchor cut cross-seed σ from 1.62 pp (parent) to 0.49 pp without moving the mean, and reanalysis falsified the parent's failure-mode story ("late-swerve at saturated v_x"), replacing it with *commit-and-freeze* 3–5 m before impact (§5.1). Neither is a mere tuning result; each carries a card part 7 mechanism statement that constrains the next batch.

6.2. What the comparison shows

The scoreboard says With-kkanbu beats Without-kkanbu by $+5.6$ pp on composite and $\approx +26$ pp at the OOD endpoint $\delta = 0.16$ ($\approx 67\% \rightarrow \approx 93\%$). The audit trail says more. Without-kkanbu wins were *leaderboard-incremental*: a reward weight, a higher training density, a higher-capacity encoder, none paired with a contemporaneous mechanism diagnosis. With-kkanbu wins are *mechanism-diagnosed and variance-controlled*: every top-3 entry carries a card part 7 statement constraining the next batch, every top-3 entry is 3-seed (the condition mandated 3-seed evaluation for leaders after the calibration finding of §5.3), and falsifications of the pre-registered mechanism are recorded alongside confirmations rather than overwritten by score gains. The shape of the work changes when kkanbu is in the loop, even on slots where the numerical lift is small.

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 card schema gives (i), SLURM jobs (ii), kkanbu (iii). Leaderboard-driven loops lack (iii): the next-action signal is whichever exper-

iment scored best, so a confident negative result like the offline-RL retirement of §5.2 collapses into "stop trying offline RL" rather than "which constraint inside that retirement was binding."

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, casual paper reading that seeded kkanbu's graph, or the decision to switch from the Without-kkanbu coordinator to the With-kkanbu loop. 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. 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. We make no sim-to-real claim; the substrate is JAX/Brax/MuJoCo, not a physical Go1.

8. Conclusion

We presented an AI Scientist system for quadruped navigation research that targets mechanism-level findings rather than benchmark scores. The system couples persistent cross-batch hypothesis tracking via **kkanbu** with structured experiment records of predictions and outcomes, driven by a multi-agent workflow for planning, implementation, execution, and analysis.

We instantiated it on a shared obstacle-navigation benchmark spanning privileged imitation learning, offline RL, and online RL under obstacle-density shift. Across batches, the system produced findings that redirected the project: it identified the Pure Pursuit conversion layer as the main bottleneck in waypoint-based navigation, revisited offline RL after diagnosing setup limitations, and rejected one of its own mechanistic hypotheses through targeted follow-ups.

Compared to an earlier autoresearch-style loop on the same codebase, our system explored broader algorithmic directions and carried unresolved questions and failed explanations across batches instead of repeatedly refining the current leader. Persistent hypothesis tracking and structured experimental memory appear to be useful ingredients for autonomous research systems aimed at falsifiable insight rather than incremental benchmark gains.

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

Do not include acknowledgements in the initial version of the paper submitted for blind review.

If a paper is accepted, the final camera-ready version can (and usually should) include acknowledgements. Such acknowledgements should be placed at the end of the section, in an unnumbered section that does not count towards the paper page limit. Typically, this will include thanks to reviewers who gave useful comments, to colleagues who contributed to the ideas, and to funding agencies and corporate sponsors that provided financial support.

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 MCP), how the profile used in this case study was *built* (predominantly via `kkanbu_ingest` and `kkanbu_curious`), 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—rather than workflow conveniences (preferred editor, preferred testing framework). In our case-study profile, the distinction is concrete: 51 nodes are typed `value`, 39 are `reasoning`, 302 are `pattern`, and only 5 are typed `preference`. The oracle is asked to speak on the user's behalf about epistemic stance, not about tool choice.

The character of the case-study profile is anchored by two themes that recur across the high-confidence value and reasoning nodes:

- **Generalisability *is* depth.** Representative value-typed nodes include "*depth comes from achieving the widest generalised value—depth and universality are the same thing to me, not trade-offs*" and "*the most universal one is the most efficient one*." The user does not treat universality as a separable axis to be optimised against depth; they collapse the two.
- **Massive experimentation as the furnace for axioms.** A representative reasoning-typed node states "*I view massive experimentation as a denoising mechanism—it strips away overfitted explanations and reveals which principles actually hold universally*," alongside the value "*empirical rigour and axiom-seeking are not opposing forces; experiments are the furnace that forges axioms.*" This taste sets the threshold for how many slots a question deserves before it can be claimed as understood, and is the substantive reason the loop runs ten experiments per batch rather than one.

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 confabulated.

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 contains 57 nodes and 86 edges. The load-bearing types for the oracle's voice—value and reasoning nodes—account for 18 and 11 entries respectively. Pattern nodes ($n = 12$) capture recurring habits of inference; observation nodes ($n = 15$) are passive extractions; one preference node records a tool-level preference.

A.3. Building the Profile: `kkanbu_ingest` and `kkanbu_curious`

The case-study profile was built primarily by two complementary intake pathways. (Other operations exist—`kkanbu_learn` for explicit teaching, `kkanbu_correct` for human revision, `kkanbu_reflect` for contradiction surfacing, `kkanbu_distill` for promoting ephemeral session history into graph nodes—but the bulk of the profile's high-confidence taste content arrived via the two operations described here.)

`kkanbu_ingest`: from documents to typed nodes. Given a file path or URL, kkanbu parses the document (Trafilatura for web pages, PyMuPDF for PDFs, python-docx for Word) and runs an LLM extraction pipeline that emits a small number of typed learnings, each with an explicit `why` field, written as nodes with `source = ingest`. The pathway is non-blocking: parse failures are logged and skipped rather than aborting the call. We use `kkanbu_ingest` to seed the graph from the user's prior writing and from a handful of domain papers (some agreeing with the user, some contradicting), so that the profile begins with a position rather than a blank slate.

`kkanbu_curious`: Socratic interview at depth. `kkanbu_curious` is the operation that produces the identity-level content the loop most depends on. Given the current graph dump, an LLM identifies a knowledge gap and emits a single question at *value*, *identity*, or *formative* depth. Surface preference questions are rejected by a regex 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 the graph as nodes whose `source` field labels them as having come from a `kkanbu_curious` session, rather than waiting for session expiry. Of the 29 high-confidence value, reasoning, and preference nodes in the case-study profile (confidence ≥ 0.8), 27 (93%) originated from `kkanbu_curious` sessions, including the generalisability-is-depth and experimentation-as-furnace statements quoted in Section A.1; the remaining two came from a single `kkanbu_ingest` call and one `distill:answer` event. `kkanbu_ingest` provides the *breadth* of the profile (positions extracted from prior writing); `kkanbu_curious` provides its *depth* (the identity-level commitments the user might never have written down).

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.

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 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 $\rightarrow$ pattern $\rightarrow$ preference $\rightarrow$ reasoning $\rightarrow$ 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 RAG context inside one assistant's loop, and they extract *facts about the user* ("works at X") rather than *normative entities* (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:

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

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:

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

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.

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

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.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

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

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:

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

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

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

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

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

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

B.5. TD3+BC

Table 6 presents hyperparameters for TD3+BC (Fujimoto & Gu, 2021b), the algorithm used by every offline-RL slot under the With-kkanbu condition. 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 slot per Section E.

Configuration that varies by slot. The actor output head and the training dataset are the only variables across the With-kkanbu TD3+BC slots; everything else in Table 6 is held constant. Section E (Table 15) gives the per-slot values.

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.

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))

C. PIL-Control Design Ablations

This appendix extends the PIL-Control hyperparameters of Section B with the design space the AI Scientist actually explored before retiring the PIL-Control stream, plus the per-density evaluation that supports the finding in Section 5.

C.1. Factorial Design

Across the first three batches, the PIL-Control stream ran nine configurations varying three factors. Eight of nine were direct-velocity-head BC; the ninth—a hybrid-head slot at the end of the third batch—tested a trajectory + velocity head with Pure-Pursuit inference.

C.2. Per-Density Success Rate

C.3. Failure-Mode Composition at $\delta = 0.16$

C.4. Validation Loss vs. Deployment Composite

D. PIL-Control Failure Mechanisms

The three mechanisms below are surfaced by experiment-reanalyzer card-part-6 probes on the nine PIL-Control variants of Table 9. Each is short, with a quantitative hook and a card / tool reference. Variant labels (P1–P9) match Table 9.

D.1. The Autoregressive-Shortcut Pathology

Surfaced by experiment-reanalyzer probes on the action-history ($k = 3$) variants P2, P4, and P7. The A* + Pure Pursuit expert has lag-1 action autocorrelation $r(v_{x,t-1}, v_{x,t}) = 0.998$ and $r(v_{\text{yaw},t-1}, v_{\text{yaw},t}) = 0.994$. A trivial copy-previous-action baseline reaches MSE $= 1.77 \times 10^{-3}$ on the held-out set, $6.6\times$ below P1's best validation loss. With action-history augmentation $k = 3$, behavior-cloning networks discover this shortcut: first-layer ℓ_1 weight attribution on P2 weighs the 9-D action-history channel $2.2\times$ more heavily than the 64-D LIDAR channel; LIDAR contributes $<10^{-3}$ incremental R^2 over the autoregressive prediction.

The pathology is asymmetric across action channels. In P7 the same analysis shows v_{yaw} is suppressed $4.2\times$ harder than v_x, even though expert v_x has *higher* lag-1 autocorrelation. The mechanism is not raw autocorrelation: 40% of expert v_{yaw}

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

commands are near-zero (straight-driving segments), so the network discovers that predicting $v_{\text{yaw}} \approx 0$ minimises loss on a large fraction of the dataset and locks there. v_x cannot collapse the same way because zero forward velocity is never safe.

D.2. Architecture × History Destructive Interference

Surfaced by the experiment-reanalyzer on P4. P4 (LIDAR-Attention with $k = 3$ action history) is *worse than either component separately at every density*, including in-distribution: 73% at $\delta = 0.01$ vs. 99% for both P1 (MLP, $k = 0$) and P3 (Attention, $k = 0$). The mechanism is architectural: the action-history 9-D vector enters a `Dense(32)` branch concatenated *post-attention* with the attended LIDAR features. The fusion MLP then has direct access to the shortcut without spatial reasoning ever being engaged. P4's training loss drops several times faster than P2's in the first 500 steps ($\approx 3\times$ faster on the loss-ratio metric in early training) and plateaus at slightly lower validation loss than its no-history sibling P3, yet *underperforms* every no-history variant at deployment. This is the cleanest in-codebase evidence that representation-side validation loss does not order policies by deployment performance: P4 reaches a comparable plateau loss to P3 yet scores -15 pp lower at deployment.

D.3. The Pure-Pursuit Coverage Failure

Surfaced by the experiment-reanalyzer on P9, the hybrid-head slot; this is the diagnosis cited as the cross-paradigm transfer in Section 5. P9 is the first card in the project to compute the Pure-Pursuit-coverage analysis. The hybrid head learned obstacle-aware trajectories that match the expert at $r > 0.99$ via finite-difference, but Pure-Pursuit conversion to velocity introduced six structural pathologies.

Approximately 9.5% of expert commands require *negative* v_x (reversal manoeuvres around obstacles) that Pure Pursuit cannot emit by construction—it turns to face the next waypoint and only drives forward. Pure Pursuit is therefore a strict action-space subset of the expert's bidirectional control distribution, and the gap widens monotonically with obstacle density.

The aggregate consequence is quantified one batch later on the PIL-Trajectory stream: $r(\text{PP-zero-rate}, \text{collision-rate}) = -0.973$ across the evaluation set (see Section 5). The next batch's PIL-Trajectory fix—replace trajectory + Pure Pursuit with a 3-D velocity head—is the remediation predicted by this analysis.

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. PIL-Control variants. Architecture: MLP $[512, 256, 128]$ or LIDAR-Attention. Expert: A* + Pure Pursuit at minimum-alignment-factor (MAF) 0.3 (no in-place rotation) or MAF $= 0$ (in-place rotation allowed). Action history $k \in \{0, 1, 3\}$. * early stopping triggered. All variants use MSE loss, batch 512, LR 10^{-3} cosine. Variants P1–P7 are seven cells of the batch-0 factorial; P8 is the batch-1 follow-up at the peak factorial cell with $k = 1$; P9 is the batch-2 hybrid-head slot.

Variant	Batch	Arch.	Expert	k	Output	Steps
P1	0	MLP	MAF $= 0.3$	0	velocity	600 K
P2	0	MLP	MAF $= 0.3$	3	velocity	80 K*
P3	0	Attn	MAF $= 0.3$	0	velocity	600 K
P4	0	Attn	MAF $= 0.3$	3	velocity	277 K*
P5 (factorial peak)	0	Attn	MAF $= 0$	3	velocity	75 K*
P6	0	MLP	MAF $= 0$	0	velocity	600 K
P7	0	MLP	MAF $= 0$	3	velocity	102 K*
P8 (batch-1 follow-up)	1	Attn	MAF $= 0$	1	velocity	146 K*
P9 (hybrid head)	2	Attn	MAF $= 0$	3	traj+vel, PP	150 K

Tool reference. A custom Pure-Pursuit mode-collapse diagnostic was promoted from the hybrid-head slot's scratchpad to the loop's shared toolkit at the experiment-reanalyzer's register turn, and re-used unchanged on the PIL-Trajectory reanalysis the following batch. Its toolkit entry is the artifact-protocol anchor for the cross-paradigm transfer claim in Section 5.

D.4. Behavior-Cloning Loss is Decoupled from Deployment Performance

Cross-experiment summary; load-bearing for Section 5. Pair-wise evidence within the PIL-Control factorial study (Appendix C):

The decoupling is a direct consequence of Appendix D.1 (autoregressive shortcut + 40% zero-mode in v_{yaw}): the BC objective is dominated by patterns that are *cheap to learn* but *uncorrelated with what matters at deployment*. Practical implication: any future BC-trained navigation policy on smooth-expert demonstrations should select checkpoints by an OOD density-sweep evaluation, not by validation loss.

E. Offline-RL Evolution Across Batches

This appendix complements the IQL hyperparameters of Section B and the reward formula of Section B.4 with the cross-batch arc that produced the offline-RL resurrection finding in Section 5. The TD3+BC hyperparameters used by every With-kkanbu offline-RL slot are documented separately in Section B.5.

E.1. Slot-by-Slot Configurations

E.2. Per-Density Success Rate

Two patterns visible only in the per-density view:

1. **The resurrection slot sacrifices in-distribution to gain out-of-distribution.** The resurrection slot's student scores *lower* than the retirement slot at the training density ($\delta = 0.04$: $86.7 < 92.7$) and at $\delta = 0.01$ ($88.7 < 96.3$), but $+8$

Table 10. Per-density success rate (%). δ in trees m^{-2}; training density $\delta = 0.04$. † training density. ‡ composite uses imputed $\delta = 0.12$ (mean of neighbouring densities) because the batch-0 slots skipped that density during evaluation. § card composite vs. stream-index composite for the same training run; see "Open issues" below. Variant labels match Table 9.

Variant	0.01	0.02	0.04†	0.08	0.12	0.16	Comp.
P1	99	96	89	64	—	17	55.9‡
P2	94	91	75	69	—	32	60.5‡
P3	99	97	96	73	—	25	62.4‡
P4	73	76	74	53	—	20	47.3‡
P5 (factorial peak)	**95**	**96**	**92**	**76**	—	**47**	**70.8**‡
P6	98	97	88	59	—	18	54.8‡
P7	83	83	77	59	—	26	53.5‡
P8 (batch-1 follow-up)	89	84	79	56	42	36	56.7 / 73.5§
P9 (hybrid head)	90	89.7	85.3	64	35.3	12.7	51.5

Table 11. Failure-mode composition at $\delta = 0.16$ (%). The pathology column distinguishes mechanism types surfaced by experiment-reanalyzer probes (card part 6). Detail in Appendix D. Variant labels match Table 9.

Variant	Succ.	Coll.	Time.	Fall	Pathology
P1	17	56	27	0	Oscillatory escalation
P2	32	67	1	0	Autoregressive shortcut
P3	25	65	10	0	Committed-arc collisions
P4	20	78	2	0	Attn $\times k$ destructive interference
P5 (factorial peak)	**47**	39	14	0	Heading-lock circling (75% lock rate)
P6	18	38	43	1	Mode-averaging indecision
P7	26	73	1	0	v_{yaw} zero-mode attractor
P8 (batch-1 follow-up)	**36**	64	0	0	Snap-commitment (lock by step 11)
P9 (hybrid head)	12.7	87.3	0	0	Pure-Pursuit coverage failure

pp higher at $\delta = 0.08$, +33 pp at $\delta = 0.12$, and +38 pp at $\delta = 0.16$. The critic-driven mode switch trades low-density precision for OOD survival; a single-density evaluation would have missed the unlock.

2. **Variance grows monotonically with density.** The variance-calibration slot's per-density σ goes 1.66 → 3.28 → 4.48 → 6.54 → 6.87 → 5.74, peaking at $\delta = 0.12$. The implication for any future offline-RL claim: in-distribution gaps below ∼3 pp are within the noise floor; OOD gaps below ∼5–6 pp are within it.

E.3. The Two Compounding Constraints

The point of the table is the unchanged-algorithm column. The 18.48 pp jump cannot be attributed to any algorithmic intervention (BC weight, critic update rate, batch size, training duration, or expectile-vs-BC anchor); the jump came from the input distributions the algorithm operates on.

F. Offline-RL Failure and Recovery Mechanisms

The findings below are surfaced by experiment-reanalyzer card-part-6 probes on the early IQL baseline, the retirement slot, the resurrection slot, and the variance-calibration slot of Section E. Each is short, with a quantitative hook and a card / tool reference.

F.1. Early IQL: Orbiting Near the Goal

The early IQL baseline failed primarily through *orbiting timeouts* rather than collisions: at $\delta = 0.01$, half of the *orbiting* failures had *zero* obstacles within 2 m of the trajectory at the moment of timeout. Reanalysis attributes the behaviour to IQL's expectile-based conservative Q producing a flat value landscape near the goal; the policy cannot distinguish among near-goal actions and oscillates indefinitely. This is a mechanism-level, not hyperparameter-level, objection: tuning τ, β, AWR clip, or learning rate would not eliminate the flatness without breaking the conservative-Q property that justifies IQL's choice in the first place. The brainstormer's batch-1 iteration uses this as the rationale for switching the offline-RL stream to TD3+BC for every subsequent slot (see Section F.5).

Table 12. Two PIL-Control variants with statistically indistinguishable best validation loss differ by 14 pp composite—direct evidence that MSE-BC is decoupled from deployment performance under high autocorrelation in the expert (Pearson $r \approx 0.98$ at lag 1 on both v_x and v_{yaw}; the autoregressive shortcut achieves $R^2 > 0.99$). Mechanism in Appendix D.4.

Pair	Best val. loss	Composite
P5 factorial peak (Attn + MAF= 0 + $k = 3$)	3.09×10^{-4}	$70.8^{\ddagger}$
P8 batch-1 follow-up (Attn + MAF= 0 + $k = 1$)	3.07×10^{-4}	$56.7^{\S}$
Δ	-0.7%	-14.1 pp

Table 13. Pure-Pursuit conversion pathologies surfaced by the experiment-reanalyzer on the hybrid-head slot P9.

Pathology	Magnitude
PP velocity error vs. FD, close-obstacle band	$41.3\times$
Expert commands needing $\lvert v_x \rvert <$ MAF floor (densest quintile)	22.4%
PP mean $\lvert v_{\text{yaw}} \rvert$ over-steer in dense fields	$+27\%$
PP mean speed in dense fields	$+37\%$
Speed dynamic-range compression by PP	$2.5\times$
Curvature $\times$ error coupling on high-curvature paths	$8.8\times$

F.2. The Retirement Slot: Two Compounding Structural Constraints

Reanalysis of the retirement slot localised the 58.65 ceiling to a constraint pair, not to TD3+BC. The actor output head $v_x = 0.75 \tanh(\cdot) + 0.75$ structurally clips v_x to $[0, +1.5]$, making backward motion mathematically impossible. The A* + Pure-Pursuit demonstration set at MAF $= 0$ is forward-only by construction (A* generates heading-aligned forward paths; Pure Pursuit converts them to forward velocities). Either constraint alone is harmful; they compound, because the architecture cannot represent what the data does not contain, and the data does not contain what the architecture cannot represent. The online-RL teacher operating in the same environment uses backward v_x as *primary* locomotion ($46-66\%$ of timesteps), so the retirement slot's constraint pair is not a property of the task—it is a property of the supervision pipeline.

F.3. The Resurrection Slot: A Density-Conditional Mode Switch

Reanalysis of the resurrection slot identified the load-bearing mechanism for the offline-RL resurrection finding's $+18.48$ pp jump: the TD3+BC critic, trained on bidirectional online-RL teacher rollouts with a symmetric actor head, learns a *density-conditional regime switch* that exists nowhere in the teacher data.

Mechanism summary: at low density, forward overshoot causes 26 early-terminations per 300-episode evaluation (the robot reaches the goal then drives past it before the success check fires); the critic learns a LIDAR-thresholded "don't backward at low density" rule that the teacher does not use. At high density the same rule inverts—the LIDAR threshold is exceeded by 66% of timesteps, and the bidirectional bang-bang regime kicks in. This is critic-driven, not actor-driven: the student's actor faithfully reproduces the teacher's bidirectional distribution at the network output level; the mode switch is implemented through advantage-weighting, which suppresses critic-disfavoured actions even when the actor would emit them.

This is offline-RL-only behaviour. A pure imitation student (PIL-Control on the same data, conjectured) would inherit the 51%-backward mode at $\delta = 0.01$ rather than reject it, because BC has no mechanism for the critic-side rejection. The mode switch is the reason the resurrection slot reaches within 5 pp of the teacher despite operating on a strict subset of the teacher's rollouts.

F.4. The Variance-Calibration Slot: Seed Variance Is the Dominant Source of Noise

The 4-seed replication of the retirement slot's configuration establishes the noise floor for any offline-RL claim on this task. Variance grows monotonically with density:

The composite-level $\sigma = 4.76$ pp is the largest paradigm σ in the With-kkanbu condition; for comparison, the online-RL calibrated benchmark (3-seed replication of the batch-3 leader) has $\sigma = 1.62$ pp at the composite level. Two custom diagnostics—a seed-variance decomposition and a seed-rank-correlation-dynamics probe—were promoted to the loop's shared toolkit at the experiment-reanalyzer's register turn for this slot. They found that the variance is wired into training via

Table 14. Validation-loss / composite decoupling across PIL-Control pairs. Variant labels match Table 9.

Pair	Δ Val. loss	Δ Composite
P5 vs. P8	$<0.7\%$	14.1 pp
P5 vs. P4	$\approx 10\%$ ($3.09/3.40 \times 10^{-4}$)	23.5 pp
P7 vs. P6	$-28.5\times$ for P7 (lower)	$+1.3$ pp (P7 worse)

Table 15. Offline-RL slot history. † the early IQL slot's evaluation skipped $\delta = 0.12$, which is imputed as the mean of $\delta = 0.08$ and $\delta = 0.16$ for the composite under the canonical scoring rule. * 4-seed mean (variance-calibration slot only). All With-kkanbu offline-RL slots use the same TD3+BC hyperparameters (Section B.5) except for the actor head and dataset, and a LIDAR-Attention encoder.

Slot	Algo	Data	Actor head	Status	Composite
Early IQL baseline	IQL	A* (legacy LIDAR convention)	— (legacy actor)	complete	62.8†
First TD3+BC attempt	TD3+BC	A* MAF= 0	$0.75\tanh(\cdot) + 0.75$	skipped	—
Second TD3+BC attempt	TD3+BC	A* MAF= 0	$0.75\tanh(\cdot) + 0.75$	skipped	—
Retirement slot (composite 58.65)	TD3+BC	A* MAF= 0 (fwd-only)	$0.75\tanh(\cdot) + 0.75$ (fwd-only)	complete	58.65
Resurrection slot (composite 77.13)	TD3+BC	online-RL teacher rollouts (bidir.)	$1.5\tanh(\cdot)$ (symmetric)	complete	**77.13**
Variance-calibration slot	TD3+BC	A* MAF= 0 (fwd-only)	$0.75\tanh(\cdot) + 0.75$ (fwd-only)	complete	52.90*

a 100–200 K-step BC-anneal scar window: the per-step weights-mean signal during this window has Spearman $\rho = +1.0$ with final composite, while standard val-loss-based checkpoint selection is blind to it (validation loss at the best step is identical across scarred and clean seeds).

F.5. Why TD3+BC and Not Continued IQL Tuning

The brainstormer's iteration for the offline-RL slot at the start of the With-kkanbu condition records the reasoning for the algorithm switch:

1. The early IQL baseline's orbiting failure was attributed to IQL's expectile-Q mechanism, not to a hyperparameter the loop had not yet swept.
2. The legacy demonstration data used an inverted-LIDAR convention and was outdated; combining the data upgrade and the algorithm switch in a single batch was the highest-information move.
3. TD3+BC has effectively no compute overhead vs. base TD3 ("minimalist approach", Fujimoto & Gu, 2021b), enabling more configurations within the slot budget. CQL and Fisher-BRC were considered and rejected on compute cost grounds.
4. Cal-QL and GOPlan were considered and rejected on implementation-weight grounds (Cal-QL adds online fine-tuning; GOPlan is too implementation-heavy for one slot).
5. Hong et al. (2023) ruled out action-history augmentation for offline RL (it breaks trajectory stitching) regardless of algorithm—so the action-history trick that helped PIL paradigms could not be transplanted, and the algorithm choice could not be hidden behind that confound.

The first item is the load-bearing one: the rationale for switching algorithms was a mechanism statement about the previous algorithm, not a leaderboard observation about the previous numbers. This is reflected in the touchpoint at the start of the With-kkanbu condition's first batch routing through the brainstormer (with kkanbu) rather than through an experiment-builder iteration on the existing IQL config. The audit trail therefore captures *why* the switch happened, not just *that* it did.

Open question for future work.

G. Findings: per-batch detail

This appendix collects the operational detail removed from the body subsections of Section 5: exact statistics, intermediate diagnostics, and the kkanbu / brainstormer hand-offs that produced each result. Per-paradigm sweeps and slot-by-slot configurations live in Appendices E and F for offline-RL.

Table 16. Per-density success rate (%) across offline-RL slots. † training density. ‡ the early IQL baseline skipped $\delta = 0.12$. The resurrection slot is single-seed; the variance-calibration slot reports the mean across 4 seeds with σ shown in parentheses.

Slot	**0.01**	**0.02**	**0.04**†	**0.08**	**0.12**	**0.16**
Early IQL baseline	90.0	88.0	84.0	64.0	—‡	41.0
Retirement slot	96.3	95.7	92.7	66.7	38.7	26.3
Resurrection slot	**88.7**	**90.7**	**86.7**	**83.7**	**71.3**	**64.0**
Variance-calibration slot (4-seed mean)	96.0 (σ1.7)	95.0 (3.3)	87.5 (4.5)	60.4 (6.5)	31.1 (6.9)	17.8 (5.7)

Table 17. The constraint pair that the retirement slot's claim absorbed into "algorithmic." The resurrection slot lifts both simultaneously.

Constraint	**Retirement slot**	**Resurrection slot**
Actor head v_x range	$[0, +1.5]$ (fwd-only)	$[-1.5, +1.5]$ (sym.)
Demonstration v_x distribution	A*: forward-only	online-RL teacher rollouts: bidirectional
Algorithm	TD3+BC, BC-anneal $2.5 \to 0.5$	*identical*
Hyperparameters	1 M steps, $\tau = 0.005$, batch 256	*identical*
Composite	58.65	**77.13**

G.1. Online-RL arc

Batch 1 (catastrophe). Batch 1 stacked three changes: density curriculum $0.04 \to 0.10$, proximity penalty $r_{\text{prox}} = -5.0 \max(0, 1 - d/1.5)$, and collision weight -50. Composite crashed to 17.1. Reanalysis showed the proximity penalty's per-step budget scaled as $O(\rho \cdot L \cdot r \cdot \beta)$ with density, 67–563× the goal reward at training densities, making "stand still" optimal.

Batches 2–3 (recovery, $+80$ pp by subtraction). Batch 2 *removed* the proximity penalty, softened collision to -10, and fixed a checkpoint-restoration bug at curriculum-phase transitions; composite jumped to 92.7. Batch 3 killed a residual $\sim$5–8% backward-spinning attractor ($v_x \approx -1.5$, $|v_{\text{yaw}}| \approx 0.8$, committed from step 1) with a bounded one-sided penalty $r_{v_x} = -2.0 \max(0, -v_x)$ and shortened a dead 430M-step hold phase to 20M (the batch-2 best checkpoint sat at step 17.9M of 430M). The attractor disappeared ($\text{frac}(v_x{<}0) = 0.000\%$ across 1.8M command steps); composite hit 97.30 on a single seed.

Batch 4, slot A (TTC falsification). Card part 7 had prescribed three reward interventions against a hypothesised "late hard swerve" kinematic ceiling, derived from a near-contact yaw-coherence diagnostic. Batch 4 implemented the TTC prescription with a reverse-KL anchor against the batch-3 leader. 3-seed reanalysis: pre-impact $\bar{v}_x$ indistinguishable from parent (Welch $t = -0.067$, $\Delta = -0.007$); *0 of 17* collision episodes matched late-swerve, and 13 of 17 showed $|v_{\text{yaw}}|$ *decreasing* as impact neared. The actual failure was *commit-and-freeze* 3–5 m before impact.

Batch 4, slot B (zero-code calibration). Three seeds of the batch-3 recipe with no code changes: IQM 96.28, stratified percentile-bootstrap CI $[95.19, 97.28]$, $\sigma = 1.62$. Variance concentrates at $\delta = 0.16$ ($\sigma \approx 4$ pp; $\sigma = 0$ at $\delta \leq 0.08$). A proximity-binned behavioural fingerprint diagnostic, promoted from this slot to the shared toolkit, found OOD variance wired in at training time via the close-band (0.6–0.9 m) $|v_{\text{yaw}}|$ repertoire width: seeds' close-band $|v_{\text{yaw}}|$ rank-predicts $\delta = 0.16$ success at Spearman $\rho = -1.0$.

Where structure was load-bearing. The brainstormer's batch-3 proposal picked the negative-v_x penalty over action-clipping (boundary gradient issues) and over a two-variable privileged-critic+curriculum change (confounded variables). The batch-4 TTC design came from a websearch that surfaced TTC-force and a stratified percentile-bootstrap protocol for IQM intervals; kkanbu picked TTC-force / reverse-KL over an asymmetric dead-zone on minimum-detectable-effect grounds. The calibration slot was kkanbu's response to a sibling 3-seed replication measuring $\sigma_{\text{composite}} \approx 4.3$ pp; kkanbu attached explicit decision rules ("$\sigma < 2$ pp ⇒ leader rock-solid; > 5 pp ⇒ mandate 3-seed"). Both diagnostics were promoted at the register turn of their producing slots.

Table 18. Behaviour of the resurrection slot's student vs. its online-RL teacher. Backward-v_x fraction is computed over success episodes only.

Density	Teacher bwd-v_x	Student behaviour	Δ vs. retirement slot
$\delta = 0.01$	51%	forward-only ($v_x \in [0, 0.8]$, rejects backward mode)	-7.66 pp
$\delta = 0.04$ (train)	53%	selective ($\bar{v}_x = -0.33$, left-heavy mix)	-6.0 pp
$\delta = 0.16$	66%	bidirectional bang-bang (dominates)	$+37.7$ pp

Table 19. Per-density variance from the variance-calibration slot's 4-seed replication of the retirement slot's configuration. The implication: in-distribution effects $\lesssim 3$ pp and OOD effects $\lesssim 5-6$ pp are within the seed-variance floor. The composite-level entry sits in the highest-variance tier of the With-kkanbu condition.

Density	Mean	σ	Range	Tier
$\delta = 0.01$	96.00	1.66	4.0	moderate
$\delta = 0.02$	95.00	3.28	7.7	high
$\delta = 0.04$	87.50	4.48	10.4	high
$\delta = 0.08$	60.42	6.54	15.4	very high
$\delta = 0.12$	31.08	6.87	16.7	very high
$\delta = 0.16$	17.84	5.74	12.7	high
Composite	52.88	4.76	11.6	highest (red flag)

G.2. Offline-RL resurrection (mode-switch detail)

The student sees 107k teacher-backward samples at $\delta = 0.01$ during training and rejects them at deployment, with median $\tanh v_x = +0.4$ versus the teacher's -0.87. At $\delta = 0.01$ the same critic produces -7.66 pp (26 of 34 failures are early-arena-exit overshoots). A teacher-mode-commitment analyser (promoted to the shared toolkit) was the diagnostic that surfaced the active rejection. The Pure-Pursuit mode-collapse diagnostic on the PIL-Trajectory side produced the $r = -0.973$ correlation between zero-collapse waypoints and collision rate that motivated the cross-paradigm dataset reuse. Per-batch tables and slot configurations are in Appendices E–F.

G.3. Seed-discipline detail

Calibration math. A batch-3 3-seed calibration replication of an existing PIL-Trajectory leader measured $\sigma_{\text{composite}} = 4.34$ pp, $\sigma_{\delta=0.16} = 9.4$ pp at the OOD endpoint. kkanbu's batch-4 brainstormer turn computed the explicit minimum-detectable effect: with $\sigma \approx 4.3$ pp and $n = 3$ seeds per arm, two-sample Welch's at $\alpha = 0.05$, power 0.8, $\Delta \approx 2.8\,\sigma\,\sqrt{2/n} \approx 9.8$ pp. The mandate that followed was the zero-code 3-seed replication of the online-RL leader of §5.1.

Prospective re-discovery. A subsequent batch added a post-network action cap $v_{x,\text{out}} = \min(v_x, 2.5 \cdot d_{\min})$ ($d_{\min}$ is the minimum LIDAR return). Per-seed composites: 97.67, 94.94, 94.72. 3-seed IQM: 94.94, CI $[93.72, 97.06]$, 1.34 pp *below* the calibrated benchmark.

Bidirectionality detail. The runtime-toggle slot in the next batch toggled the cap at eval time on existing checkpoints ($\sim$5 lines of evaluation code, no retraining). 3-seed IQM $\Delta = -0.17$ pp, CI $[-0.54, +1.69]$. Per-seed deltas: $+2.44$, -0.67, -0.17. The cap-firing-vs-lock-onset alignment diagnostic (promoted to the shared toolkit) found the cap fires 12–17 steps before lock onset on 3 of 4 cap-hurt episodes, supporting the causal direction.

H. Implementation Details

H.1. Environment Specification

Table 20 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 m), collision, fall, or timeout.

Table 20. Environment parameters.

Parameter	Value
Environment	`Go1NavigationForestEnvV5`
Training room	30m × 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	$[-1.2,\ 1.2]\,\mathrm{rad/s}$

H.2. Observation Space

Table 21 details the observation space components (75D total). The encoding deliberately omits absolute position so the policy is scale-invariant to room size.

Table 21. 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 $(v_x, v_y, v_{\mathrm{yaw}})$

H.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 approximately 3,000 successful episodes for PIL methods and $\sim$9.4M transitions for IQL.

H.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.

I. Example Trajectories

To make the score gap between a strong policy and a weaker one concrete, Figure 4 shows twelve evaluation rollouts each at $\delta \in \{0.01, 0.04, 0.16\}$ for two policies. The leader is the With-kkanbu top entry from Table 1 (PPO with a spatially-gated forward-KL anchor; composite 97.50, single representative seed). The weaker policy is the IQL baseline from Section 3.8 (composite $\approx$38). Both policies share the same low-level locomotion checkpoint and the same evaluation harness; the only difference is the navigation policy. Start positions are blue dots, goals are yellow stars, obstacles are grey, completed paths are green, and incomplete (failed) paths are red.

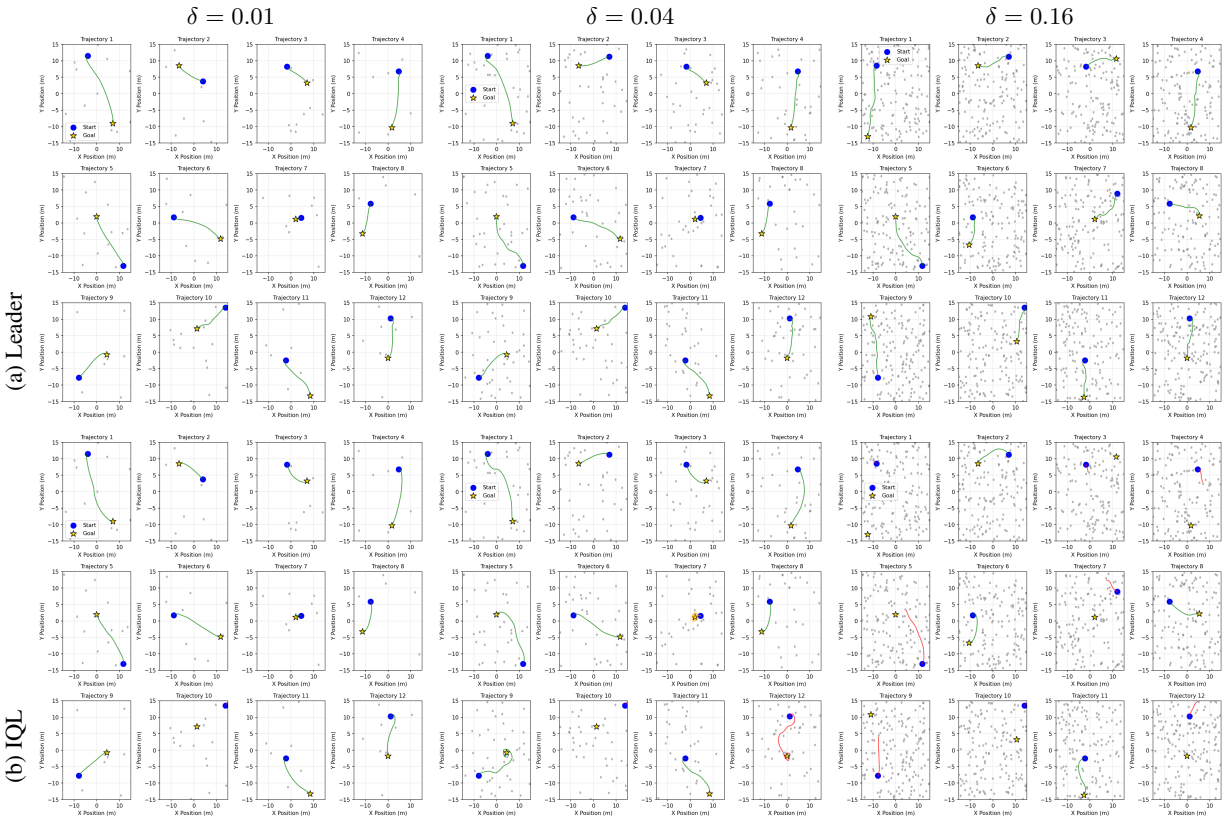

Figure 4. Twelve evaluation rollouts at $\delta \in \{0.01, 0.04, 0.16\}$ for (a) the With-kkanbu leader (PPO + spatially-gated forward-KL anchor, composite 97.50) and (b) the IQL baseline (composite ≈ 38). Green paths reach the goal; red paths fail. The leader completes most rollouts at every density; IQL shows scattered failures already at $\delta = 0.01$ and reaches only a quarter of the goals at the OOD endpoint $\delta = 0.16$.

At low density ($\delta = 0.01$) the leader reaches every goal cleanly, while the IQL baseline already shows scattered failures: short red stubs near the start indicate the policy halts before navigating around even sparse clutter. At in-distribution density for the offline data ($\delta = 0.04$) the gap widens; the IQL baseline produces a mix of near-goal orbits and outright stalls. By the OOD endpoint $\delta = 0.16$, the contrast is the visual companion to the headline score gap: the leader still completes most rollouts (success rate 0.93), whereas the IQL baseline reaches only a quarter of the goals (success rate 0.25), with the failure modes named in Appendix F (orbiting near the goal, halting in front of dense clutter).