Keep or Roll Back?
Out-of-Loop Judges for Self-Improving VLA and End-to-End Embodied Policies

Zilin Huang[1] Zhengyang Wan[1] Kangan Qian[2] Zihao Sheng[1] Sikai Chen[1,*]
[1]University of Wisconsin–Madison [2]University of California, Santa Cruz

Project page: https://zilin-huang.github.io/SealedLoop/

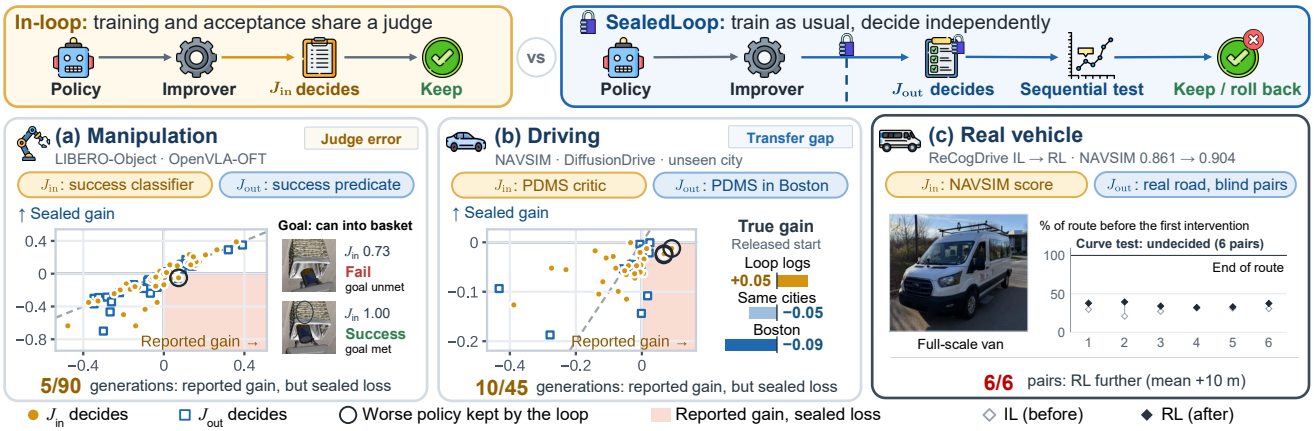


Figure 1. Self-improvement loops report gains that a sealed judge does not find. Top: today the loop's own judge decides what is kept; SEALEDLOOP lets a sealed judge decide through a sequential test. (a, b) Reported gain against sealed gain, one point per generation; the shaded quadrant holds reported gains with a sealed loss. (c) Real-vehicle field experiment on a published improvement step, in blind pairs.

Abstract

Self-improvement is now a practical recipe for embodied policies: a policy collects experience, is retrained, and is kept when a judge rates it higher. However, the judge is usually trained or chosen inside the same loop, so its evidence is not independent of the policy. We introduce SEALEDLOOP*, an evaluation framework that adds a* sealed judge*: an evaluator outside the loop that the improver cannot see or optimize against. A new policy is kept only when a sequential test on the sealed judge confirms the gain, with a bounded probability of keeping a worse one. Running the same loop under each judge, we separate two sources of a false gain, which we call* judge error *(the loop's judge is wrong on its own data) and* transfer gap *(a real gain there does not carry over to unseen conditions). We evaluate on LIBERO manipulation, NAVSIM driving, and a full-scale real vehicle. In manipulation, a success classifier accepted 65 of 196 failures in one generation. In driving, policies that gain on the loop's own frames lose in an unseen city, so even an exact in-loop judge would keep 8/15 of them. When the sealed test decides, a driving loop built to improve ends with a better policy than under its own critic (0.674 against 0.649 PDMS, from 0.638), and on driving pairs with known sealed gain the test accepted 11 of 12 real gains and none of 59 losses. In a real-world field experiment on a published improvement step, the reinforced version drove further into a curve than the version it started from in all 6 blind pairs, though neither completed it. These results show that evidence from inside the loop, however accurate, is not enough to decide whether a policy has improved.* SEALEDLOOP *offers a foundation for trustworthy self-improvement of embodied policies, in which progress is judged by evidence the improving system cannot influence.*

*Corresponding author: sikai.chen@wisc.edu

1. Introduction

Self-improvement is now a practical recipe for embodied policies: a policy rolls out, a judge labels its episodes, the policy is fine-tuned on what was judged good, and the cycle repeats. Robot policies improve this way through reinforcement-learning fine-tuning and deployment experience [1, 8, 11, 12, 38, 52, 92], and driving planners through self-play and reinforcement learning against a simulator's score [14, 42, 59]. Every round ends with the same decision: is the new policy better, and should it replace the old one? Run without a human, the cycle is a bounded form of recursive self-improvement [9, 36, 62, 88], and it is only as reliable as this decision. In most loops, however, the judge that supplies the training signal or selects the training episodes also provides the evidence for this decision. The evidence is therefore not independent of the policy it evaluates. In language-model self-improvement, the loops that run reliably without a human are those whose judge the optimizer cannot influence, such as a code executor or a test suite [54, 91, 93], whereas loops with a learned judge stop improving within a few rounds [66, 67].

Embodied loops face the same problem. The gain a loop reports can disagree with an independent evaluation for two reasons. First, the judge may mismeasure performance on the conditions it observes. Learned judges carry measurement error even before a policy is optimized against them: the best of 22 general robot reward models differs from human-verified progress labels by 0.67 points on a five-point scale on average [37]. Optimization then widens the error. A policy trained on a learned reward overshoots the true objective [20, 46, 47, 77], and progress rewards can be gamed by stopping short of the goal [68]. Second, a genuine gain on those conditions may not transfer to conditions the loop has not seen. We call the first *judge error* and the second the *transfer gap*. Figure 1 shows one case of each. In manipulation, a success classifier accepts failed placements that look like successes (a). In driving, gains on the loop's own evaluation frames turn into losses in an unseen city (b). A more accurate judge reduces judge error. The transfer gap can only be seen with evidence from outside the loop. But physical evaluation is slow and cannot be replayed, so embodied loops rarely collect such evidence.

A wrong decision does not stay in its round, because each kept policy collects the data for the next one. In driving, trajectory refinement guided by a learned critic peaks after two to three iterations [33]. Seeing such effects requires following a loop generation by generation and counting the cases where the policy exploits its judge. Among the 56 embodied loops we surveyed, a per-generation curve with a quantified case of such exploitation has not yet been reported (Tab. 1). How the two sources affect successive acceptance decisions has not been measured. Measuring it requires an evaluator the improver cannot influence and an acceptance rule with a controlled error rate. Both exist elsewhere: sealed evaluation and anytime-valid acceptance for software agents [26, 27, 63], and sequential tests for comparing robot policies [65]. We bring them to embodied self-improvement, where the judge is learned from the loop's own data and sealed conditions differ from those the loop sees.

We propose SEALEDLOOP, an evaluation framework that adds a sealed judge to the loop (Fig. 2). A *sealed judge* is an evaluator outside the loop whose data, parameters and schedule are hidden from the improver. It runs beside the loop, and a new policy is kept only when an anytime-valid sequential test [75] on it certifies the gain. The sealed conditions are fixed before the improver runs and drawn independently of it. The probability that the test keeps a worse policy therefore does not exceed a level set in advance, whatever the improver does. The test is also sequential, so each decision comes with the amount of evaluation it used. Running the same loop twice, once with its own judge deciding and once with the sealed judge deciding, shows what the choice of the deciding judge changes. Both judges score every generation in either run. The difference between the gain the loop reports and the sealed gain, the *judge gap*, can therefore be measured in every generation. The true score on the loop's own data separates it into judge error and transfer gap.

We apply SEALEDLOOP to two simulated loops, LIBERO manipulation and NAVSIM driving, and to a full-scale real vehicle. In manipulation the judge gap is judge error: in one generation the success classifier accepted 65 of 196 failures of the policy it trained, and 5 of 90 generations report a gain where the sealed judge finds a loss (Fig. 1a). In driving a transfer gap appears in every generation, and 10 of 45 report a gain with a sealed loss (Fig. 1b). Policies started from the released checkpoint gain on the loop's own evaluation frames and lose in an unseen city, so even an exact in-loop judge, one with no error on the loop's own data, would keep 8/15 of them. The sealed test still accepts real improvements: on driving pairs whose sealed gain is known, it accepted 11 of 12 real gains and none of 59 losses. In a driving loop built to improve, it certified all 11 first-step gains ($+0.029$ to $+0.043$ PDMS), of which the loop's learned critic accepted one. In a real-world field experiment, the reinforced version of ReCogDrive [42], whose gain is measured in simulation, drove further into a left curve in all 6 blind pairs, but neither completed it (Fig. 1c).

Our contributions are:

- **We propose SEALEDLOOP, an evaluation framework for embodied self-improvement loops.** A judge sealed from the improver, rather than the judge the loop trains with, decides each update through a sequential test, and running the loop under both judges separates a false gain into judge error and transfer gap.

Loop	Judge	Rounds	Curve	Exploitation
Manipulation				
SimpleVLA-RL [38]	simulator	n.s.	no	no
HIL-SERL [52]	classifier + human	n.s.	no	no
$\pi^*_{0.6}$ (RECAP) [1]	human raters	2	partial	no
VLAC [92]	VLM critic	n.s.	no	no
Robo-Dopamine [68]	reward model	n.s.	no	described
SARM2 [11]	reward model	3	partial	mentioned
VLAW [28]	VLM reward model	2	partial	error only
Driving				
GIGAFLOW [14]	simulator rules	1	no	error only
ReCogDrive [42]	simulator score	1	no	no
EvaDrive [33]	learned critic	$\geq 4^\dagger$	partial	no
R^2LPL [25]	rule detectors	5	yes	no
AD-R1 [83]	world-model critic	1	no	error only
This paper (both)	in-loop vs. sealed	5 to 8	yes	quantified

Table 1. Embodied self-improvement loops and their judges, a selection of the 56 surveyed (all in Appendix E). Rounds: improvement rounds reported (n.s.: not stated; †refinement iterations at inference). Curve: results per generation. Exploitation: policy gaming its judge is quantified (error only: the judge's error).

- **We show that evidence from inside the loop is not enough to approve an update.** On LIBERO and NAVSIM, in-loop judges keep policies that are not better, and even an exact in-loop judge would keep 8/15 of the driving policies that lose in an unseen city.
- **We test a published improvement step on a full-scale real vehicle with a sealed judge.** In a blind, paired field experiment, the direction of its simulator gain holds on the road: the reinforced version goes further into a curve in all 6 pairs, though neither version completes it.

2. Related Work

Self-improvement and its verifiers. Self-improvement loops run without a human when the optimizer cannot influence their judge, as with agents judged by an executor or a test suite [9, 58, 72, 84, 90, 91, 93]. Loops that update weights under a learned judge plateau within a few rounds [66, 67, 89]: judges favor outputs like their own [73], and learned rewards are over-optimized [20, 77]. For software agents, the judge has been moved out of the loop by hidden-rollout harnesses [26], test windows scored once [27], error-controlled verifiers [61, 63] and pre-registered analyses [30]. In embodied loops the judge is learned from the loop's own data and the sealed conditions differ from those the loop sees, so a reported gain can fail through judge error or a transfer gap; SEALEDLOOP measures both, where a sealed evaluation alone gives only their sum. Anytime-valid inference [56, 75] has been used to compare robot policies [65, 71], evaluate language models with fewer samples [95] and tell a drifting judge from a drifting system [41]. We use it as the loop's acceptance rule, bounding the probability of keeping a worse policy.

Self-improving embodied policies. Reinforcement-learning fine-tuning of vision-language-action models is judged by the simulator [8, 38, 69]. On hardware, the judge is a human, a reward classifier trained in the loop [1, 11, 12, 52, 80], a vision-language critic [92], a synthetic verifier [78] or a world model [28, 32, 86, 96]. Such judges carry measurement error before any optimization [37, 83], and exploitation has so far been described qualitatively [11, 68]. Driving planners trained against rule scores [13, 14, 42, 59, 64, 98] report single runs, and learned-critic refinement peaks or saturates within a few iterations [33, 87]. Across the 56 surveyed loops, a per-generation curve with a quantified case of exploitation has not yet been reported (Tab. 1). Transfer is a further open problem: open-loop and closed-loop metrics disagree [5, 15, 43], a benchmark gain can reverse when only the executor changes [97], and driving models lose 22.8% of their success rate under distribution shift [23], while reinforcement-learning fine-tuning leaves visual robustness unchanged [50]. Simulated evaluation must be calibrated against hardware [2, 40]. Physical evidence is rare: one study compares two policies over about 200 km in three runs [39]. Building on these loops, we measure judge error and the transfer gap generation by generation on two embodiments, and compare both versions of a published improvement step in blind pairs on a full-scale vehicle.

3. Self-Improvement under a Sealed Judge

SEALEDLOOP adds a sealed judge and a sequential test on it to a self-improvement loop, and runs the loop once under each judge (Fig. 2). We define the quantities the two judges measure, separate their disagreement into two sources, and give the test that decides each update.

3.1. Problem setup

A self-improvement loop holds a policy π_g at generation g. An improver $\mathcal{I}$ produces a candidate $\pi_{g+1} = \mathcal{I}(\pi_g; J_{\text{in}})$ by collecting episodes and retraining with an *in-loop judge* J_{in}, such as a success classifier, a learned critic or a vision-language model. The improver may also update J_{in}. The loop judges candidates on its own evaluation set $\mathcal{D}_{\text{in}}$ and reports the gain

$$\hat{\Delta}_g = J_{\text{in}}(\pi_{g+1}; \mathcal{D}_{\text{in}}) - J_{\text{in}}(\pi_g; \mathcal{D}_{\text{in}}), \tag{1}$$

where $J_{\text{in}}(\pi; \mathcal{D}_{\text{in}})$ is the mean score the judge gives π on $\mathcal{D}_{\text{in}}$ and one version of the judge scores both policies. Let $s_c(\pi) \in [0, 1]$ be the true task score of π on a condition c (a success predicate, a rule-based driving score). A condition is a start state with a seed in manipulation and a log in driving, where s_c is the mean PDMS over the frames of the log. Let $S(\pi; \mathcal{D})$ be the mean of $s_c(\pi)$ over the conditions of $\mathcal{D}$. A *sealed judge* J_{out} applies S on a pool $\mathcal{P}$

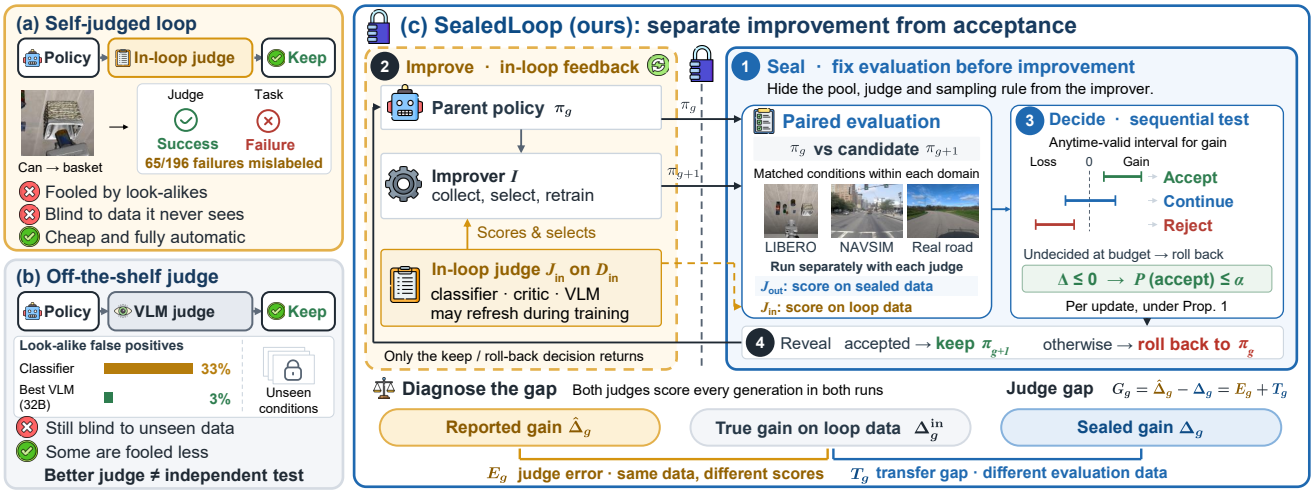


Figure 2. Who decides what the loop keeps. (a) The loop's own judge. (b) An off-the-shelf judge: fooled less, still blind to unseen data. (c) SEALEDLOOP: the improver trains with J_{in} only, a sequential test on the sealed J_{out} decides, and both judges score each generation.

whose data, judge parameters and schedule are hidden from the improver. The *sealed gain* and the *judge gap* are

$$\Delta_g = S(\pi_{g+1}; \mathcal{P}) - S(\pi_g; \mathcal{P}), \qquad G_g = \hat{\Delta}_g - \Delta_g. \quad (2)$$

3.2. Decomposing the judge gap

With $\Delta_g^{\text{in}} = S(\pi_{g+1}; \mathcal{D}_{\text{in}}) - S(\pi_g; \mathcal{D}_{\text{in}})$ the true gain on the loop's own data,

$$G_g = \underbrace{\hat{\Delta}_g - \Delta_g^{\text{in}}}_{E_g\text{: judge error}} + \underbrace{\Delta_g^{\text{in}} - \Delta_g}_{T_g\text{: transfer gap}}. \quad (3)$$

We obtain Δ_g^{in} by applying the true score S to $\mathcal{D}_{\text{in}}$; the loop itself does not use it. A better judge reduces E_g, but no judge that sees only $\mathcal{D}_{\text{in}}$ can detect T_g, the part of a true gain that does not carry over to deployment conditions, such as an unseen city or a real vehicle. T_g vanishes up to noise when $\mathcal{D}_{\text{in}}$ and $\mathcal{P}$ share a distribution, as on manipulation.

3.3. Certified acceptance

The test draws conditions $c_1, c_2, \ldots$ from $\mathcal{P}$, runs both policies on each, and reads the paired differences $d_i = s_{c_i}(\pi_{g+1}) - s_{c_i}(\pi_g) \in [-1, 1]$, whose mean is Δ_g. A betting confidence sequence [75] gives intervals $C_t = [L_t, U_t]$ from $d_1, \ldots, d_t$ with

$$\Pr\left(\forall t \ge 1 : \ \Delta_g \in C_t\right) \ge 1 - \alpha, \quad (4)$$

so the test may read C_t after every pair [56]. It *accepts* if $L_t > 0$, *rejects* if $U_t < 0$, and otherwise continues up to a budget N; an undecided candidate is rolled back. We call a gain *certified* when the test accepts, and a loss certified when it rejects.

Proposition 1. *Assume that each d_i has mean Δ_g given $d_1, \ldots, d_{i-1}$. Then, for any improver and any stopping time, the probability that the test accepts a candidate with $\Delta_g \le 0$ or rejects one with $\Delta_g \ge 0$ is at most α.*

Proof. Acceptance at t implies $L_t > 0 \ge \Delta_g$ and rejection implies $U_t < 0 \le \Delta_g$. In both cases $\Delta_g \notin C_t$, which has probability at most α over all t jointly.

The assumption holds when the conditions are drawn independently of each other and of both policies. Sealing secures the second part: the conditions of a generation are fixed before the improver runs, so the improver cannot fit the candidate to them. Our draws are stratified without replacement. Under this sampling, the test accepts a candidate without a real gain in at most 1.2% of simulated generations and none that loses 0.05 or 0.10 (Appendix A). Over K generations it makes at most αK wrong decisions in expectation under the same assumption.

3.4. SEALEDLOOP

SEALEDLOOP combines the sealed judge and the test into four steps per generation (Fig. 2):

1. **Seal:** fix the evaluation pool, the judge and the rule that draws this generation's conditions.
2. **Improve:** produce π_{g+1} with J_{in} only; nothing from J_{out} reaches the improver.
3. **Decide:** evaluate π_g and π_{g+1} on paired conditions until the test accepts, rejects or reaches its budget.
4. **Reveal:** keep π_{g+1} if it was accepted and roll back otherwise; record the readings of both judges.

We run each loop in variants that differ in the judge that feeds the test in step 3. In the *in-loop* variant the loop's own judge decides, and in the *sealed* variant the sealed judge does. Both variants are scored by J_{out}, so every generation

yields $\hat{\Delta}_g$, Δ_g and both decisions, where a loop run once under its own judge yields only $\hat{\Delta}_g$. We count two kinds of wrong in-loop decisions: candidates kept with a negative sealed gain (*worse kept*), and candidates rolled back although the sealed judge certifies their gain (*gains lost*).

4. Experimental Setup

Manipulation. The released OpenVLA-OFT checkpoints already solve LIBERO-Object (Tab. 2), so we adapt a LIBERO-Spatial checkpoint with a few demonstrations (Appendix B); the three start policies score 0.661 / 0.746 / 0.491 on the sealed pool. Each generation collects episodes on 300 fresh start conditions per task and fine-tunes with LoRA on those the classifier labels successful. The classifier is refreshed on these labels, as in loops that learn it from their own episodes [52, 80]. The sealed pool crosses the suite's 50 start states with environment seeds; each generation draws a disjoint set, stratified by start state.

Driving. The start policy is DiffusionDrive trained from scratch on 20,000 frames from Las Vegas, Pittsburgh and Singapore. It was chosen by its PDMS on 3,731 Boston frames of the sealed pool; replaying every sealed test on the other 12,954 frames gives the same decision for 43 of 45 candidates and accepts none (Appendix C). Control runs start from the released checkpoint. The critic predicts PDMS, NAVSIM's rule-based driving score in $[0, 1]$, from the frozen encoder's features, as learned critics do in published driving loops [33, 87]. It is refreshed every generation on true labels of fresh proposals or on its own pairwise preferences. Each generation, the critic picks one of 20 proposals per training-city frame as the target, hard frames are replayed [25, 33], and the policy is retrained. The sealed judge reads PDMS on a frozen pool of 16,685 Boston frames, a city no generation trains on.

Real-world field experiment. The third testbed is a full-scale drive-by-wire electric van on a real road, with an RTK-corrected inertial navigation system and an on-board GPU (Fig. 6a). Both policy versions run through the same software, plan from the same front camera once per 0.80 s, and are executed by the same trajectory tracker at a speed limit of 4 m/s. Each camera frame is reprojected to the geometry of the NAVSIM front camera before it reaches the policy (Appendix D). The main route is 144 m long, with a left curve of about 55°; a 60 m straight route and a 143 m route with a right curve complete the study. A pair is one drive of each version, back to back in a random order; the safety driver and the operator see only blind labels. All scored drives took place in November 2026. A drive's score is the fraction of the route completed before the first intervention (a brake, steering or throttle override, a timeout or a stall), read from the drive-by-wire records.

Implementation details. All tests use $\alpha = 0.05$, at least 10 pairs and a threshold of zero. Runs last five generations, and 8 classifier-decided manipulation runs continue to eight. On driving the test reads the mean difference of each log. Improver processes run with the sealed judge's files masked.

5. Experimental Results

5.1. Reported gains do not hold under a sealed judge

On driving, the two judges diverge at once (Figs. 1 and 3 and Tab. 3). At the first generation the judge gap is $+0.095$ $[+0.087, +0.104]$ for the critic refreshed on its own labels and $+0.045$ $[+0.037, +0.052]$ for the one refreshed on true labels. The former kept one candidate per seed, each scoring lower in Boston than the policy it replaced. The sealed judge rolled back all 15 driving candidates (mean sealed gain at the first generation -0.028). In this reduced-data setting the improver produced no gain that reached the unseen city, and the sealed judge shows it: with its own critic deciding, the same loop kept a worse policy in every seed.

On manipulation the improver works, and the in-loop judge still makes both kinds of wrong decision. The judges diverge later than on driving: in the median run that reaches this threshold, the gap first exceeds 0.10 in success rate at generation 3. The sealed judge kept 12 generations (mean certified gain $+0.147$); the classifier kept one candidate certified as worse and rolled back 3 certified gains. Both variants end with better policies: re-run on new seeds, the final policies gain $+0.18$ (sealed) and $+0.21$ (classifier) over their start policies on average, and $+0.06$ and $+0.02$ on new start states, where one kept candidate of each variant loses (Appendix B). Over the 45 classifier-decided generations, replacing the classifier by the success predicate on the same pairs changes 6 decisions and replacing 300 pairs by the sealed budget changes 8, so the judge and the amount of evaluation each change decisions. A third variant runs this comparison as a loop, in which the true success of the same 300 pairs decides. Its decisions match the sealed test in 44 of 45 generations (Tab. 3), so on manipulation the wrong decisions come from the classifier's labels.

5.2. The gap has two sources

Judge error. The in-loop judge is reliable on the data it was fitted to and loses that reliability as the loop moves the policy and the data away from it. Before the first generation the critic's pick beats the policy's own choice by $+0.010$ PDMS on the loop's frames; in Boston its false-positive rate rises from 0.164 to 0.383 and its pick is worse (-0.042; Appendix C). Refreshed on its own labels, the critic's error is $+0.070$ $[+0.061, +0.079]$ at the first generation, where it keeps worse policies (Fig. 4a); later the refreshed critics turn pessimistic. On manipulation, $\mathcal{D}_{\text{in}}$ and $\mathcal{P}$ share one distribution, so the gap is judge error. The classifier's false-positive rate rises with the policy's own failures (Fig. 4c;

	Manipulation (simulation)	Driving (simulation)	Real-world vehicle
Benchmark	LIBERO-Object [48], 3 tasks	NAVSIM [15], unseen city	full-scale van, three routes on a real road
Start policy	OpenVLA-OFT [35], few-shot adapted	DiffusionDrive [45], reduced data	ReCogDrive-2B, imitation version (IL) [42]
Improver	rejection-sampling fine-tuning	best-of-20 self-distillation	published reinforcement-learning step (RL)
In-loop judge	success classifier, refreshed on own labels	PDMS critic, refreshed on true or own labels	simulator score: NAVSIM
Sealed judge	success predicate, unseen start conditions	PDMS in Boston	route completed before the first intervention
Runs	3 tasks × 3 seeds × 3 variants	3 seeds × 3 variants, 3 control runs	one step; blind paired drives of IL and RL
Per generation	in-loop 300, sealed up to 1,000 pairs	in-loop 5,000, sealed 16,685 frames	6, 3 and 1 pairs on the three routes

Table 2. Testbeds. The variants of a testbed differ in which judge decides; on manipulation the two tests also differ in budget.

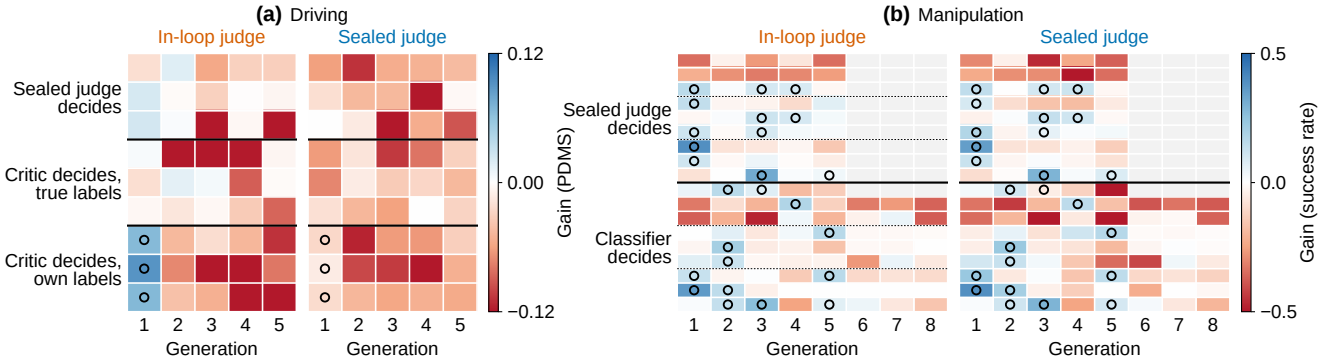


Figure 3. Every generation as seen by the two judges. Each cell is a candidate's gain, as reported by the in-loop judge (left grid of each pair) and as measured by the sealed judge (right grid). Rows are runs; rings mark kept candidates.

Testbed	Judge	Runs	Kept	Worse kept	Gains lost
Manipulation	sealed	9	12	–	–
	classifier	9	13	1	3
	true success	9	13	0	1
Driving	sealed	3	0	–	–
	critic	6	3	3	0

Table 3. Acceptance decisions in generations 1 to 5 of each run.

across runs, 0.000 to 0.227 at the first generation and 0.000 to 1.000 at the last), in two patterns. The new policy can fail with the object held at the edge of the basket, which looks like a success; the classifier accepted 65 of 196 failures in one such generation. In 3 generations of one run, a refresh without failures blinded the classifier to the next policy's failures (Appendix B). Elsewhere the refresh changes few decisions: with the classifier frozen before the loop, 7 of 114 differ and no run keeps a different policy (Appendix F).

Transfer gap. Even a judge without error would keep worse policies, because gains on the loop's own data do not carry over to data the loop has not seen. In every driving generation and seed, the true gain on the loop's frames exceeds the gain in Boston ($+0.008$ to $+0.123$). Runs from the released checkpoint change their score by $+0.047$ to $+0.058$ on the loop's evaluation frames and by -0.107 to -0.075 in Boston (Fig. 4b). The loss does not need a new city: on unseen logs of the training cities the same candidates change by -0.064 to -0.038, 70% of the drop from the loop's frames to Boston, and 44 of the 45 candidates of the main runs lose there too. Nor does it need judge error: fed with the *true* scores on the loop's evaluation frames, the test accepts 8/15 control generations, but none of the 60 candidates on the unseen logs. Nor do better targets remove it: trained on the proposal with the highest true score on each frame (0.965 on average against 0.887 for the policy's own choice), the first-generation candidate still changes its Boston score by -0.018, -0.008 and -0.108 at three learning rates (Appendix C). The transfer gap comes from evaluating on the loop's data; only a sealed pool exposes it.

5.3. Off-the-shelf judges do not replace a sealed one

A stronger judge is the obvious remedy for judge error, yet none of the off-the-shelf judges we tried replaces the sealed one. We re-scored the recorded manipulation episodes with four off-the-shelf vision-language models and a robot reward model [37] (Fig. 5). None is blinded by the failure-free refresh, which affects only the judge trained in the loop. On the generation whose failures look like successes, only two of the five accept fewer failures than the classifier (23% and 3% against 33%), and three accept more. No judge reaches the success predicate's decision on the same pairs as often as the loop's classifier (35 of 38 generations against at most 27). The largest judge, Qwen3-VL-32B, accepts the fewest failures but also rejects many true successes (true-positive rate 0.69) and agrees on 26 generations. We also ran the

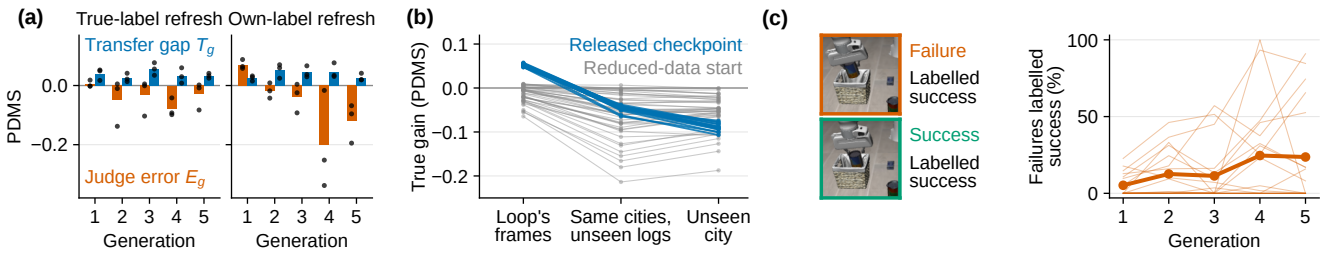


Figure 4. Two sources of the judge gap. (a) Driving: judge error E_g and transfer gap T_g per generation under the two critic refresh rules (bars: three-seed mean; dots: seeds). (b) True gain of each driving candidate on three sets of frames. (c) Manipulation: a failure and a success, both labeled success by the classifier, and the share of each candidate's failures so labeled (thin lines: runs; thick line: mean).

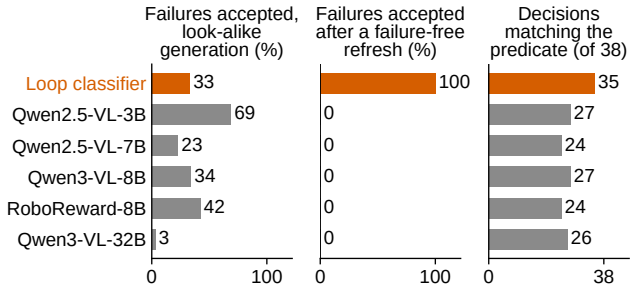


Figure 5. Off-the-shelf judges on manipulation. Share of failures accepted on a look-alike generation (196 failures) and after a failure-free refresh (36), and generations where the test fed by each judge matches the success predicate.

Judge	Rule	Kept	Worse kept	Regret	Frames
Sealed	sequential test (ours)	0	0	0.000	354,754
	first 17 logs, read once	0	0	0.000	41,160
	whole pool	0	0	0.000	750,825
In-loop	sequential test	3	3	0.061	n/a
	peak of its report	8	8	0.267	n/a

Table 4. Acceptance rules replayed on the driving runs. Worse kept, regret: number and total sealed loss of kept candidates with negative sealed gain. In-loop test: six critic-decided runs.

loop with a vision-language judge deciding, in three runs. It kept 3 of 15 candidates and no worse policy, but rolled back 2 certified gains, and its decisions matched those of the success predicate on 11 of 15 generations against 13 of 15 for a frozen classifier. A larger judge thus trades accepted failures for rejected successes. Appendix B gives the inputs and thresholds of every judge.

5.4. Certified acceptance keeps gains, not losses

Known gains. The sealed test accepts gains where they exist. It kept 12 manipulation generations and no driving one, since no driving candidate gained in the unseen city. On 76 pairs of driving policies whose sealed gain is known from the whole pool (Fig. 7a), it accepts 11 of the 12 real gains and none of the 59 losses or the 5 pairs near zero. Averaged over 500 log orders, a bootstrap over 50 logs fixed in advance accepts 10.8 of the 12 per-log gains with 22% of the frames; the sequential test accepts 11.2 with 53%. What the test adds is an error rate that holds wherever it stops: on pools built to have no gain, the same bootstrap read after every 10 logs accepts 5.9%, against a nominal 2.5%, and the sequential test accepts none (Appendix C).

A loop built to improve. When a loop does improve, the sealed test certifies its gains, and the judge that loses them is the loop's own. A start policy trained on 5,000 frames imitates the logged expert on 20,000 new frames per generation. Of the 32 candidates, 11 are a first step from the start policy and gain $+0.029$ to $+0.043$ PDMS in the unseen city. The sealed test accepted all 11 (Fig. 7b); the learned critic accepted one. The other 21 build on an accepted policy and gain -0.004 to $+0.017$; the sealed test left all of them undecided. The held policy reaches 0.674 from 0.638 under the sealed judge, 0.649 under the critic, and 0.689 under a paired test on the true scores of the loop's own frames (Fig. 7c). Judge error thus costs in both directions. The paired test on true scores keeps more here because these gains transfer; on the self-training runs above the same evidence kept policies that lose.

Acceptance rules. Which judge feeds the rule matters more than the rule itself (Tab. 4). Rules fed by the sealed judge keep no worse policy. Fed by the in-loop judge, the same test and the rollback-to-peak rule proposed for language models [47] each keep some. The sequential test reaches the decisions of reading the whole pool every generation with 47% of its frames.

5.5. Field experiment on a full-scale vehicle

Tested step. Finally, in a real-world field experiment, we test whether the simulator gain of one published fine-tuning step shows up on a real vehicle. ReCogDrive [42] fine-tunes an imitation-learned driving policy (IL) by reinforcement learning (RL) whose reward is the NAVSIM score, and it reports its gain with the same simulator and metric, on the separate navtest split. On our pipeline the two released versions score 0.861 and 0.904 on navtest, a paired gain of

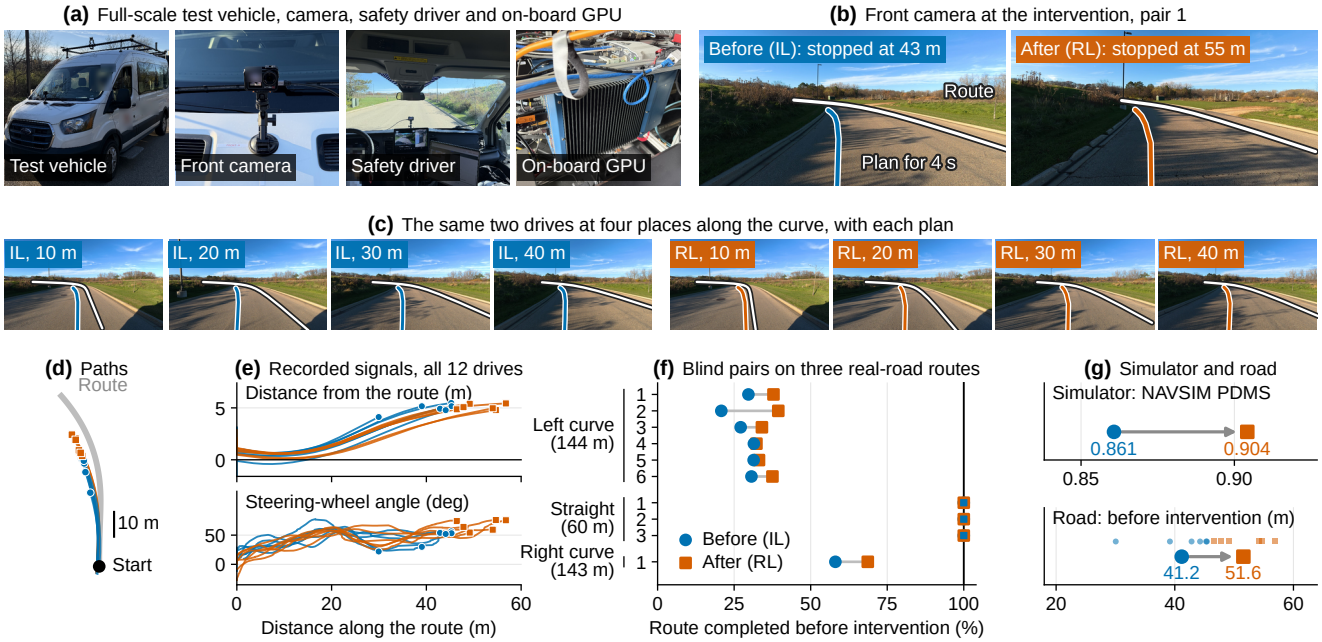


Figure 6. Field experiment on a full-scale vehicle. (a) Test vehicle. (b, c) Front camera of the first pair at the intervention and before it, with planned trajectories. (d, e) Paths, distance from the route and steering angle on the left curve. (f) Route completed before the first intervention per blind pair. (g) The same step in the simulator and on the left curve (large markers: means; small: pairs).

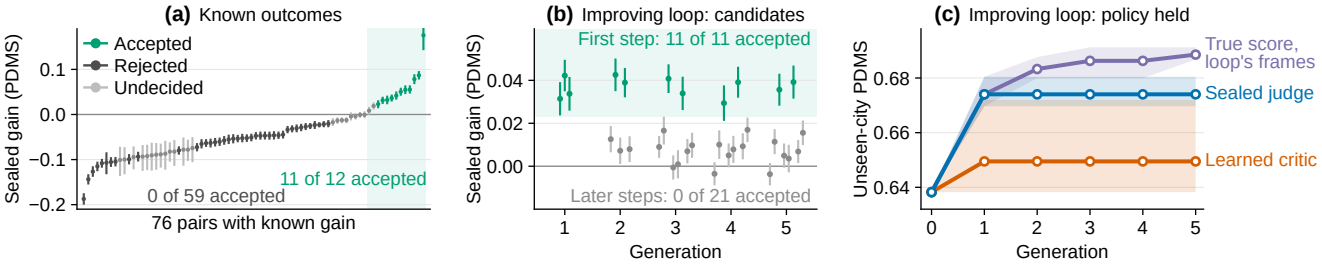


Figure 7. What the sequential test accepts. (a) 76 policy pairs with known sealed gain (95% intervals), colored by the test's decision. (b) Candidates of a loop built to improve. (c) Unseen-city PDMS of the policy it holds under three judges (line: seed mean; band: range).

+0.043 (95% interval +0.039 to +0.048) over 12,118 scenes. The sealed judge is the full-scale vehicle of Sec. 4: both versions drove in blind pairs, and each drive is scored by the fraction of the route completed before the first intervention.

On the road. No drive completed the main route (0 of 12). Both versions are deployed zero-shot, without fine-tuning on data from our vehicle, so the interventions reflect a deployment gap that both share. The safety driver stopped every drive in the first half of the curve, between 30 and 57 m (Fig. 6d). In both versions the planned trajectory leaves the route toward the inner curb of the curve (Fig. 6b, c; paths and steering in Fig. 6d, e; Appendix D). RL went further into the curve in 6 of 6 pairs (exact sign test, one-sided $p = 0.016$), by 10.4 m on average (+0.072 of the route; Fig. 6f). The sequential test, which bounds the size of the gain, is undecided after 6 pairs (confidence sequence $[-0.50, +0.57]$; Appendix D). On the straight route both versions completed every drive (6 of 6), and on the right curve RL went further in the one pair driven. The direction of the simulator gain held on the road in all 6 pairs: the step that raises the score from 0.861 to 0.904 moves the intervention from 41.2 to 51.6 m along the 144 m route (Fig. 6g).

6. Conclusion

Self-improving embodied policies, in driving and in manipulation, can report improvement that a judge outside the loop does not find, because the in-loop judge errs on the loop's own data and real gains on that data fail to transfer to unseen data. Stronger off-the-shelf judges remove neither. SEALEDLOOP instead seals the deciding judge from the improver. The sealed test accepted 11 of 12 real gains and none of 59 losses, and certified all 11 first-step gains of a loop built to improve, of which the loop's own critic ac-

cepted one. In our real-world field experiment, the direction of a gain measured in simulation held on the road: the reinforced version drove 10.4 m further into a curve that neither version completed.

A separate test set is not enough, since our in-loop variants had the sealed reading and still kept worse policies. The sealed judge must therefore make the decision itself, and a sequential test lets it do so at 47% of the evaluation cost. So far, our loops run in simulation, and the field experiment tests one published improvement step on one vehicle. Future work will run the loop on hardware, with a simulator as the in-loop judge and the physical platform as the out-of-loop evaluator.

Acknowledgment

This research was supported by the U.S. Department of Transportation (No. 69A3552348305) and by grants from NVIDIA utilizing NVIDIA RTX PRO 6000 Blackwell Max-Q Workstation Edition GPUs.

LLM Usage Statement

We used Claude Code (Anthropic), a coding assistant built on large language models, as a coding and writing assistant in this work. It helped write and review experiment scripts, prepare figures and tables from the recorded results, and edit the text for clarity. All experimental designs, hypotheses and conclusions were decided by the authors, every reported number is produced by scripts from the sealed records, and the authors take full responsibility for the content of the paper.

References

[1] Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace B. Connors, James Darpinian, Karan Dhabalia, Jared Di Carlo, Danny Driess, Michael Robert Equi, Adnan Esmail, Yunhao Fang, Chelsea Finn, Catherine Glossop, Thomas Godden, Ivan Goryachev, Lachy Groom, Hunter Hancock, Karol Hausman, Gashon Hussein, Brian Ichter, Szymon Jakubczak, Rowan Jen, Tim Jones, Benjamin Katz, Liyiming Ke, Chandra Kuchi, Marinda Lamb, Devin Leblanc, Sergey Levine, Adrian Li-Bell, Yao Lu, Vishnu Mano, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Charvi Sharma, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, Will Stoeckle, Alexander Swerdlow, James Tanner, Marcel Torne, Quan Vuong, Anna Walling, Haohuan Wang, Blake Williams, Sukwon Yoo, Lili Yu, Ury Zhilinsky, and Zhiyuan Zhou. $\pi^*_{0.6}$: a VLA That Learns From Experience. In *Proceedings of Robotics: Science and Systems*, 2026. 2, 3, 26

[2] Apurva Badithela, David Snyder, Lihan Zha, Joseph Mikhail, Matthew O'Kelly, Anushri Dixit, and Anirudha Majumdar. Reliable and Scalable Robot Policy Evaluation with Imperfect Simulators. In *2026 IEEE International Conference on Robotics and Automation (ICRA)*, pages 21728–21735. IEEE, 2026. 3

[3] Daniel Bogdoll, Rajanikant Patnaik Ananta, Abeyankar Giridharan, Isabel Moore, Gregory Stevens, and Henry X. Liu. Mcity Data Engine: Iterative Model Improvement Through Open-Vocabulary Data Selection. In *2025 IEEE 28th International Conference on Intelligent Transportation Systems (ITSC)*, pages 130–136. IEEE, 2025. 27

[4] Konstantinos Bousmalis, Giulia Vezzani, Dushyant Rao, Coline Manon Devin, Alex X. Lee, Maria Bauza Villalonga, Todor Davchev, Yuxiang Zhou, Agrim Gupta, Akhil Raju, Antoine Laurens, Claudio Fantacci, Valentin Dalibard, Martina Zambelli, Murilo Fernandes Martins, Rugile Pevceviciute, Michiel Blokzijl, Misha Denil, Nathan Batchelor, Thomas Lampe, Emilio Parisotto, Konrad Zolna, Scott Reed, Sergio Gómez Colmenarejo, Jonathan Scholz, Abbas Abdolmaleki, Oliver Groth, Jean-Baptiste Regli, Oleg Sushkov, Thomas Rothörl, Jose Enrique Chen, Yusuf Aytar, David Barker, Joy Ortiz, Martin Riedmiller, Jost Tobias Springenberg, Raia Hadsell, Francesco Nori, and Nicolas Heess. RoboCat: A Self-Improving Generalist Agent for Robotic Manipulation. *Transactions on Machine Learning Research*, 2024. 26

[5] Wei Cao, Marcel Hallgarten, Tianyu Li, Daniel Dauner, Xunjiang Gu, Caojun Wang, Yakov Miron, Marco Aiello, Hongyang Li, Igor Gilitschenski, Boris Ivanovic, Marco Pavone, Andreas Geiger, and Kashyap Chitta. Pseudo-Simulation for Autonomous Driving. In *Proceedings of The 9th Conference on Robot Learning*, pages 4709–4722. PMLR, 2025. 3

[6] Wei-Jer Chang, Akshay Rangesh, Kevin Joseph, Matthew Strong, Masayoshi Tomizuka, Yihan Hu, and Wei Zhan. SPACeR: Self-Play Anchoring with Centralized Reference Models. In *International Conference on Learning Representations*, pages 47010–47027, 2026. 27

[7] Hanzhi Chen, Anran Zhang, Simon Schaefer, Kejia Chen, Shi Chen, Daniel Cremers, Oier Mees, and Stefan Leutenegger. Robot Self-Improvement via Human-Video Dynamics Models. arXiv preprint arXiv:2606.21406, 2026. 26

[8] Kang Chen, Zhihao Liu, Tonghe Zhang, Zhen Guo, Si Xu, Hao Lin, Hongzhi Zang, Xiang Li, Quanlu Zhang, Zhaofei Yu, Guoliang Fan, Tiejun Huang, Yu Wang, and Chao Yu. π_{RL}: Online RL Fine-tuning for Flow-based Vision-Language-Action Models. arXiv preprint arXiv:2510.25889, 2025. 2, 3, 26

[9] Mingguang Chen, Licheng Wang, and Bo Qu. Recursive Self-Improvement in AI: From Bounded Self-Refinement to Autonomous Research Loops. arXiv preprint arXiv:2607.07663, 2026. 2, 3

[10] Qimao Chen, Fang Li, Yuechen Luo, Zehan Zhang, Haiyang Sun, Fangzhen Li, Bing Wang, Guang Chen, Yang Ji, Jiong Deng, Hongwei Xie, Hangjun Ye, Long Chen, and Yi Zhang. DriveReward: A Comprehensive Dataset and Generative Vision-Language Reward Model for Autonomous Driving. arXiv preprint arXiv:2606.08525, 2026. 27

[11] Qianzhong Chen, Hau Zheng, Justin Yu, Suning Huang, Jiankai Sun, Ken Goldberg, Chuan Wen, Pieter Abbeel,

Yide Shentu, Philipp Wu, and Mac Schwager. SARM2: Multi-Task Stage Aware Reward Modeling for Self Improving Robotic Manipulation. arXiv preprint arXiv:2606.10305, 2026. 2, 3, 26

[12] Yuhui Chen, Shuai Tian, Shugao Liu, Yingting Zhou, Haoran Li, and Dongbin Zhao. ConRFT: A Reinforced Fine-tuning Method for VLA Models via Consistency Policy. In *Proceedings of Robotics: Science and Systems*, 2025. 2, 3, 26

[13] Daphne Cornelisse, Aarav Pandya, Kevin Joseph, Joseph Suárez, and Eugene Vinitsky. Building reliable sim driving agents by scaling self-play. arXiv preprint arXiv:2502.14706, 2025. 3, 27

[14] Marco Cusumano-Towner, David Hafner, Alexander Hertzberg, Brody Huval, Aleksei Petrenko, Eugene Vinitsky, Erik Wijmans, Taylor W. Killian, Stuart Bowers, Ozan Sener, Philipp Kraehenbuehl, and Vladlen Koltun. Robust Autonomy Emerges from Self-Play. In *Proceedings of the 42nd International Conference on Machine Learning*, pages 11710–11737. PMLR, 2025. 2, 3, 27

[15] Daniel Dauner, Marcel Hallgarten, Tianyu Li, Xinshuo Weng, Zhiyu Huang, Zetong Yang, Hongyang Li, Igor Gilitschenski, Boris Ivanovic, Marco Pavone, Andreas Geiger, and Kashyap Chitta. NAVSIM: Data-Driven Non-Reactive Autonomous Vehicle Simulation and Benchmarking. In *Advances in Neural Information Processing Systems*, pages 28706–28719. Curran Associates, Inc., 2024. 3, 6, 19

[16] Haoyuan Deng, Yudong Lin, Yuanjiang Xue, Haoyang Du, Qianzhun Wang, Boyang Zhou, Zhenyu Wu, and Ziwei Wang. E2HiL: Entropy-Guided Sample Selection for Efficient Real-World Human-in-the-Loop Reinforcement Learning. *IEEE Robotics and Automation Letters*, 11(7):8084–8091, 2026. 26

[17] Haoyu Fu, Diankun Zhang, Zongchuang Zhao, Jianfeng Cui, Hongwei Xie, Bing Wang, Guang Chen, Hangjun Ye, Dingkang Liang, and Xiang Bai. MindDrive: A Vision-Language-Action Model for Autonomous Driving via Online Reinforcement Learning. In *Computer Vision – ECCV 2026*, pages 503–522. Springer Nature Switzerland, 2026. 27

[18] Hao Gao, Shaoyu Chen, Bo Jiang, Bencheng Liao, Yiang Shi, Xiaoyang Guo, Yuechuan Pu, Haoran Yin, Xiangyu Li, Xinbang Zhang, Ying Zhang, Wenyu Liu, Qian Zhang, and Xinggang Wang. RAD: Training an End-to-End Driving Policy via Large-Scale 3DGS-based Reinforcement Learning. In *Advances in Neural Information Processing Systems*, pages 32551–32576. Curran Associates, Inc., 2025. 27

[19] Hao Gao, Shaoyu Chen, Yifan Zhu, Yuehao Song, Wenyu Liu, Qian Zhang, and Xinggang Wang. RAD-2: Scaling Reinforcement Learning in a Generator-Discriminator Framework. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2026. 27

[20] Leo Gao, John Schulman, and Jacob Hilton. Scaling Laws for Reward Model Overoptimization. In *Proceedings of the 40th International Conference on Machine Learning*, pages 10835–10866. PMLR, 2023. 2, 3

[21] Yinfeng Gao, Qichao Zhang, Deqing Liu, Zhongpu Xia, Guang Li, Kun Ma, Guang Chen, Hangjun Ye, Long Chen, Da-Wei Ding, and Dongbin Zhao. PerlAD: Towards Enhanced Closed-Loop End-to-End Autonomous Driving With Pseudo-Simulation-Based Reinforcement Learning. *IEEE Robotics and Automation Letters*, 11(5):5821–5828, 2026. 27

[22] Junhao Ge, Zuhong Liu, Longteng Fan, Yifan Jiang, Jiaqi Su, Yiming Li, Zhejun Zhang, and Siheng Chen. Unraveling the Effects of Synthetic Data on End-to-End Autonomous Driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 28859–28869, 2025. 27

[23] Simon Gerstenecker, Andreas Geiger, and Katrin Renz. Fail2Drive: Benchmarking Closed-Loop Driving Generalization. *arXiv preprint arXiv:2604.08535*, 2026. 3

[24] Varun Giridhar, Anant Khandelwal, Jeremy A. Collins, Ignat Georgiev, and Animesh Garg. Beyond Imitation: Self-Improving Robot Policies via Off-Policy Q-Planning. arXiv preprint arXiv:2608.21204, 2026. 26

[25] Cheng Gong, Haoyang Wang, Chao Lu, Zirui Li, and Jianwei Gong. Learning from Mistakes: Rollout-Retrieval Lifelong Policy Learning for Autonomous Driving. arXiv preprint arXiv:2606.30537, 2026. 3, 5, 27

[26] Diandian Guo, Cong Cao, Fangfang Yuan, Yingqi Wang, Yueshan Wang, and Dakui Wang. Self-Authored Verification Is Unreliable in Heuristic Self-Improving Agents. arXiv preprint arXiv:2607.24300, 2026. 2, 3

[27] Jiacheng Guo, Suozhi Huang, Yunlong Gao, Zihao Li, Jason Ge, Shushu Liang, Xu Kuang, and Mengdi Wang. AQuA: Recursively Self-Improving Quantitative Trading Research Agents. arXiv preprint arXiv:2608.12841, 2026. 2, 3

[28] Yanjiang Guo, Tony Lee, Lucy Xiaoyang Shi, Jianyu Chen, Percy Liang, and Chelsea Finn. VLAW: Iterative Co-Improvement of Vision-Language-Action Policy and World Model. In *Proceedings of the 43rd International Conference on Machine Learning*, pages 37998–38011. PMLR, 2026. 3, 26

[29] Pu Hua, Minghuan Liu, Annabella Macaluso, Yunfeng Lin, Weinan Zhang, Huazhe Xu, and Lirui Wang. GenSim2: Scaling Robot Data Generation with Multi-modal and Reasoning LLMs. In *Proceedings of The 8th Conference on Robot Learning*, pages 5030–5066. PMLR, 2025. 26

[30] Mehmet Iscan. Falsification, Not Exposure: An Internally Preregistered Placebo-Controlled Decomposition of Self-Repair Feedback in Frozen Small Code Models. arXiv preprint arXiv:2606.31511, 2026. 3

[31] Bernhard Jaeger, Daniel Dauner, Jens Beißwenger, Simon Gerstenecker, Kashyap Chitta, and Andreas Geiger. CaRL: Learning Scalable Planning Policies with Simple Rewards. In *Proceedings of The 9th Conference on Robot Learning*, pages 5301–5338. PMLR, 2025. 27

[32] Joel Jang, Seonghyeon Ye, Zongyu Lin, Jiannan Xiang, Johan Bjorck, Yu Fang, Fengyuan Hu, Spencer Huang, Kaushil Kundalia, Yen-Chen Lin, Loïc Magne, Ajay Mandlekar, Avnish Narayan, You Liang Tan, Guanzhi Wang, Jing Wang, Qi Wang, Yinzhen Xu, Xiaohui Zeng, Kaiyuan Zheng, Ruijie Zheng, Ming-Yu Liu, Luke Zettlemoyer, Dieter Fox, Jan Kautz, Scott Reed, Yuke Zhu, and Linxi Fan. DreamGen: Unlocking Generalization in Robot Learning through Video World Models. In *Proceedings of The 9th Conference on Robot Learning*, pages 5170–5194. PMLR, 2025. 3, 26

[33] Siwen Jiao, Kangan Qian, Hao Ye, Yang Zhong, Ziang Luo, Sicong Jiang, Zilin Huang, Yangyi Fang, Jinyu Miao, Zheng Fu, Yunlong Wang, Kun Jiang, Diange Yang, Rui Fan, and Baoyun Peng. EvaDrive: Evolutionary Adversarial Policy Optimization for End-to-End Autonomous Driving. In *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2027. 2, 3, 5, 27

[34] Alex Kendall, Jeffrey Hawke, David Janz, Przemyslaw Mazur, Daniele Reda, John-Mark Allen, Vinh-Dieu Lam, Alex Bewley, and Amar Shah. Learning to Drive in a Day. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8248–8254. IEEE, 2019. 27

[35] Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-Tuning Vision-Language-Action Models: Optimizing Speed and Success. In *Proceedings of Robotics: Science and Systems*, 2025. 6, 15

[36] Zae Myung Kim, Young-Jun Lee, Seungyeon Jwa, and Dongyeop Kang. Meta^n: Recursive Self-Improvement through Emergent Depth. In *Advances in Neural Information Processing Systems*, 2026. 2

[37] Tony Lee, Andrew Wagenmaker, Karl Pertsch, Percy Liang, Sergey Levine, and Chelsea Finn. RoboReward: General-Purpose Vision-Language Reward Models for Robotics. arXiv preprint arXiv:2601.00675, 2026. 2, 3, 6, 18, 26

[38] Haozhan Li, Yuxin Zuo, Jiale Yu, Yuhao Zhang, Zhaohui Yang, Kaiyan Zhang, Xuekai Zhu, Yuchen Zhang, Tianxing Chen, Ganqu Cui, Dehui Wang, Dingxiang Luo, Yuchen Fan, Youbang Sun, Jia Zeng, Jiangmiao Pang, Shanghang Zhang, Yu Wang, Yao Mu, Bowen Zhou, and Ning Ding. SimpleVLA-RL: Scaling VLA Training via Reinforcement Learning. In *International Conference on Learning Representations*, pages 125846–125869, 2026. 2, 3, 26

[39] Tianyu Li, Li Chen, Caojun Wang, Haochen Liu, Kashyap Chitta, Zhenjie Yang, Yuhang Lu, Naisheng Ye, Yihang Qiu, Yufei Wang, Luoxi Zou, Jiaxin Peng, Jin Pan, Zhaoyu Su, Andrei Bursuc, Shengbo Eben Li, Andreas Geiger, Peng Su, and Hongyang Li. World Engine: Towards the Era of Post-Training for Autonomous Driving. arXiv preprint arXiv:2606.19836, 2026. 3, 27

[40] Xuanlin Li, Kyle Hsu, Jiayuan Gu, Oier Mees, Karl Pertsch, Homer Rich Walke, Chuyuan Fu, Ishikaa Lunawat, Isabel Sieh, Sean Kirmani, Sergey Levine, Jiajun Wu, Chelsea Finn, Hao Su, Quan Vuong, and Ted Xiao. Evaluating Real-World Robot Manipulation Policies in Simulation. In *Proceedings of The 8th Conference on Robot Learning*, pages 3705–3728. PMLR, 2025. 3

[41] Yitao Li. Who Drifted: the System or the Judge? Anytime-Valid Attribution in LLM Evaluation Pipelines. arXiv preprint arXiv:2606.15474, 2026. 3

[42] Yongkang Li, Kaixin Xiong, Xiangyu Guo, Fang Li, Sixu Yan, Gangwei Xu, Lijun Zhou, Long Chen, Haiyang Sun, Bing Wang, Kun Ma, Guang Chen, Hangjun Ye, Wenyu Liu, and Xinggang Wang. ReCogDrive: A Reinforced Cognitive Framework for End-to-End Autonomous Driving. In *International Conference on Learning Representations*, pages 157518–157556, 2026. 2, 3, 6, 7, 23, 27

[43] Zhiqi Li, Zhiding Yu, Shiyi Lan, Jiahan Li, Jan Kautz, Tong Lu, and Jose M. Alvarez. Is Ego Status All You Need for Open-Loop End-to-End Autonomous Driving? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14864–14873, 2024. 3

[44] Zeqiao Li, Yijing Wang, Haoyu Wang, Zheng Li, Peng Li, Wenfei Liu, and Zhiqiang Zuo. From Learning to Mastery: Achieving Safe and Efficient Real-World Autonomous Driving with Human-in-the-Loop Reinforcement Learning. In *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7925–7932. IEEE, 2025. 27

[45] Bencheng Liao, Shaoyu Chen, Haoran Yin, Bo Jiang, Cheng Wang, Sixu Yan, Xinbang Zhang, Xiangyu Li, Ying Zhang, Qian Zhang, and Xinggang Wang. DiffusionDrive: Truncated Diffusion Model for End-to-End Autonomous Driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12037–12047, 2025. 6, 19

[46] Jianzhe Lin. When Good Verifiers Go Bad: Silent Negative Transfer in Verifier-Guided VLM Training. arXiv preprint arXiv:2606.14629, 2026. 2, 27

[47] Jianzhe Lin. Self-Improvement Can Self-Regress: The Rise-and-Collapse Failure Mode of LLM Self-Training. arXiv preprint arXiv:2606.21090, 2026. 2, 7, 27

[48] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking Knowledge Transfer for Lifelong Robot Learning. In *Advances in Neural Information Processing Systems*, pages 44776–44791. Curran Associates, Inc., 2023. 6, 15

[49] Haochen Liu, Tianyu Li, Haohan Yang, Li Chen, Caojun Wang, Ke Guo, Haochen Tian, Hongchen Li, Hongyang Li, and Chen Lv. Reinforced Refinement With Self-Aware Expansion for End-to-End Autonomous Driving. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 48(5): 5774–5792, 2026. 27

[50] Jijia Liu, Feng Gao, Bingwen Wei, Xinlei Chen, Qingmin Liao, Yi Wu, Chao Yu, and Yu Wang. What Can RL Bring to VLA Generalization? An Empirical Study. In *Advances in Neural Information Processing Systems*, pages 97121–97151, 2025. 3

[51] Han Lu, Xiaosong Jia, Yichen Xie, Siyu Sun, Wenlong Liao, Xiaokang Yang, and Junchi Yan. ActiveAD: Planning-Oriented Active Learning for End-to-End Autonomous Driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3656–3666, 2026. 27

[52] Jianlan Luo, Charles Xu, Jeffrey Wu, and Sergey Levine. Precise and dexterous robotic manipulation via human-in-the-loop reinforcement learning. *Science Robotics*, 10(105): eads5033, 2025. 2, 3, 5, 26

[53] Yuechen Luo, Fang Li, Qimao Chen, Shaoqing Xu, Jiaxin Liu, Ziying Song, Zhi-xin Yang, and Fuxi Wen. Unleashing VLA Potentials in Autonomous Driving via Explicit Learning from Failures. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 24833–24842, 2026. 27

[54] Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco J. R. Ruiz, Abbas

Mehrabian, M. Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli, and Matej Balog. AlphaEvolve: A coding agent for scientific and algorithmic discovery. arXiv preprint arXiv:2506.13131, 2025. 2

[55] Muleilan Pei, Shaoshuai Shi, and Shaojie Shen. Advancing Multi-agent Traffic Simulation via R1-Style Reinforcement Fine-Tuning. In *International Conference on Learning Representations*, pages 130791–130807, 2026. 27

[56] Aaditya Ramdas, Peter Grünwald, Vladimir Vovk, and Glenn Shafer. Game-Theoretic Statistics and Safe Anytime-Valid Inference. *Statistical Science*, 38(4):576–601, 2023. 3, 4, 14

[57] Xuanchi Ren, Yifan Lu, Tianshi Cao, Ruiyuan Gao, Shengyu Huang, Amirmojtaba Sabour, Tianchang Shen, Tobias Pfaff, Jay Zhangjie Wu, Runjian Chen, Seung Wook Kim, Jun Gao, Laura Leal-Taixe, Mike Chen, Sanja Fidler, and Huan Ling. Cosmos-Drive-Dreams: Scalable Synthetic Driving Data Generation with World Foundation Models. arXiv preprint arXiv:2506.09042, 2025. 27

[58] Maxime Robeyns, Martin Szummer, and Laurence Aitchison. A Self-Improving Coding Agent. In *ICLR 2025 Workshop on Scaling Self-Improving Foundation Models without Human Supervision*, 2025. 3

[59] Luke Rowe, Roger Girgis, Rodrigue de Schaetzen, Daphne Cornelisse, Alaap Grandhi, Felix Heide, Eugene Vinitsky, Christopher Pal, and Liam Paull. Scaling Self-Play for End-to-End Driving. In *Proceedings of the 10th Conference on Robot Learning*, 2026. 2, 3, 27

[60] Hongzhi Ruan, Pei Liu, Weiliang Ma, Zhengning Li, Xueyang Zhang, Jun Ma, Dan Xu, and Kun Zhan. Closed Loop Dynamic Driving Data Mixture for Real-Synthetic Co-Training. In *Advances in Neural Information Processing Systems*, 2026. 27

[61] Shuvom Sadhuka, Drew Prinster, Clara Fannjiang, Gabriele Scalia, Bonnie Berger, Aviv Regev, and Hanchen Wang. E-valuator: Reliable Agent Verifiers with Sequential Hypothesis Testing. In *ICML 2026 Workshop on Statistical Frameworks for Uncertainty in Agentic Systems*, 2026. 3

[62] Jürgen Schmidhuber. Gödel Machines: Fully Self-referential Optimal Universal Self-improvers. In *Artificial General Intelligence*, pages 199–226. Springer Berlin Heidelberg, 2007. 2

[63] Biswa Sengupta. Self-Evolving Agents with Anytime-Valid Certificates. arXiv preprint arXiv:2607.00871, 2026. 2, 3

[64] Shuyao Shang, Yuntao Chen, Yuqi Wang, Yingyan Li, and Zhao-Xiang Zhang. DriveDPO: Policy Learning via Safety DPO For End-to-End Autonomous Driving. In *Advances in Neural Information Processing Systems*, pages 81565–81585. Curran Associates, Inc., 2025. 3, 27

[65] David Snyder, Apurva Badithela, Nikolai Matni, George J. Pappas, Anirudha Majumdar, Masha Itkina, and Haruki Nishimura. Beyond Binary Success: Sample-Efficient and Statistically Rigorous Robot Policy Comparison. In *Proceedings of Robotics: Science and Systems*, 2026. 2, 3

[66] Yuda Song, Hanlin Zhang, Carson Eisenach, Sham Kakade, Dean Foster, and Udaya Ghai. Mind the Gap: Examining the Self-Improvement Capabilities of Large Language Models. In *International Conference on Learning Representations*, pages 39894–39931, 2025. 2, 3

[67] Yifan Sun, Yushan Liang, Zhen Zhang, Xin Liu, and Jiaye Teng. Theoretical Modeling of Large Language Model Self-Improvement Training Dynamics Through Solver-Verifier Gap. In *International Conference on Learning Representations*, pages 18389–18425, 2026. 2, 3

[68] Huajie Tan, Sixiang Chen, Yijie Xu, Zixiao Wang, Cheng Chi, Yuheng Ji, Yaoxu Lyu, Zhongxia Zhao, Xiansheng Chen, Peterson Co, Shaoxuan Xie, Guocai Yao, Pengwei Wang, Zhongyuan Wang, and Shanghang Zhang. General Process Reward Modeling for Robotic Reinforcement Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 22412–22422, 2026. 2, 3, 26

[69] Shuhan Tan, Kairan Dou, Yue Zhao, and Philipp Krähenbühl. Interactive Post-Training for Vision-Language-Action Models. In *CVPR 2025 Workshop on Foundation Models Meet Embodied Agents*, 2025. 3

[70] Haochen Tian, Tianyu Li, Haochen Liu, Jiazhi Yang, Yihang Qiu, Guang Li, Junli Wang, Yinfeng Gao, Zhang Zhang, Liang Wang, Hangjun Ye, Long Chen, and Hongyang Li. SimScale: Learning to Drive via Real-World Simulation at Scale. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 36365–36374, 2026. 27

[71] Joseph A. Vincent, Haruki Nishimura, Masha Itkina, Paarth Shah, Mac Schwager, and Thomas Kollar. How Generalizable Is My Behavior Cloning Policy? A Statistical Approach to Trustworthy Performance Evaluation. *IEEE Robotics and Automation Letters*, 9(10):8619–8626, 2024. 3

[72] Wenyi Wang, Piotr Piękos, Li Nanbo, Firas Laakom, Yimeng Chen, Mateusz Ostaszewski, Mingchen Zhuge, and Jürgen Schmidhuber. Huxley-Gödel Machine: Human-Level Coding Agent Development by an Approximation of the Optimal Self-Improving Machine. In *International Conference on Learning Representations*, pages 80356–80386, 2026. 3

[73] Koki Wataoka, Tsubasa Takahashi, and Ryokan Ri. Self-Preference Bias in LLM-as-a-Judge. In *NeurIPS 2024 Workshop on Safe Generative AI*, 2024. 3

[74] Ian Waudby-Smith and Aaditya Ramdas. Confidence sequences for sampling without replacement. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 20204–20214, 2020. 15

[75] Ian Waudby-Smith and Aaditya Ramdas. Estimating means of bounded random variables by betting. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 86(1): 1–27, 2024. 2, 3, 4, 14, 15

[76] Bingchuan Wei, Bingqi Huang, Jingheng Ma, Zeyu Zhang, and Sen Cui. FATE: Closed-Loop Feasibility-Aware Task Generation with Active Repair for Physically Grounded Robotic Curricula. arXiv preprint arXiv:2603.01505, 2026. 26

[77] Lorenz Wolf, Robert Kirk, and Mirco Musolesi. Reward Model Overoptimisation in Iterated RLHF. In *NeurIPS 2025 Workshop on Aligning Reinforcement Learning Experimentalists and Theorists*, 2025. 2, 3

[78] Wenli Xiao, Jia Xie, Tonghe Zhang, Haotian Lin, Letian Fu, Haoru Xue, Jalen Lu, Yi Yang, Cunxi Dai, Zi Wang, Jimmy Wu, Guanzhi Wang, S. Shankar Sastry, Ken Goldberg, Linxi Fan, Yuke Zhu, and Guanya Shi. ENPIRE: Agentic Robot Policy Self-Improvement in the Real World. In *Proceedings of the 10th Conference on Robot Learning*, 2026. 3, 26

[79] Zikang Xiong, Weixin Li, Zhouchonghao Wu, Akshay Rangesh, Saarth Bonde, Grantland Hall, Chen Tang, Yihan Hu, and Wei Zhan. TerraTransfer: Learning End-to-End Driving Policies Without Expert Demonstrations. arXiv preprint arXiv:2606.17386, 2026. 27

[80] Charles Xu, Qiyang Li, Jianlan Luo, and Sergey Levine. RLDG: Robotic Generalist Policy Distillation via Reinforcement Learning. In *Proceedings of Robotics: Science and Systems*, 2025. 3, 5, 26

[81] Charles Xu, Jost Tobias Springenberg, Michael Robert Equi, Ali Amin, Adnan Esmail, Sergey Levine, and Liyiming Ke. RL Token: Bootstrapping Online RL with Vision-Language-Action Models. In *Proceedings of the 10th Conference on Robot Learning*, 2026. 26

[82] Yuquan Xue, Le Xu, Zeyi Liu, Zhenyu Wu, Zhengyi Gu, Xinyang Song, Bofang Jia, and Ziwei Wang. WorldSample: Closed-loop Real-robot RL with World Modelling. arXiv preprint arXiv:2607.02431, 2026. 26

[83] Tianyi Yan, Tao Tang, Xingtai Gui, Yongkang Li, Jiasen Zheng, Weiyao Huang, Lingdong Kong, Wencheng Han, Xia Zhou, Xueyang Zhang, Yifei Zhan, Kun Zhan, Cheng-zhong Xu, and Jianbing Shen. AD-R1: Closed-Loop Reinforcement Learning for End-to-End Autonomous Driving with Impartial World Models. In *2026 IEEE/CVF Conference on Computer Vision and Pattern Recognition - Findings Track (CVPRF)*, pages 1085–1095. IEEE, 2026. 3, 27

[84] Haoyan Yang, Mario Xerri, Solha Park, Huajian Zhang, Yiyang Feng, Sai Akhil Kogilathota, and Jiawei Zhou. Self-Improvement of Large Language Models: A Technical Overview and Future Outlook. *Transactions on Machine Learning Research*, 2026. 3

[85] Jiazhi Yang, Kashyap Chitta, Shenyuan Gao, Long Chen, Yuqian Shao, Xiaosong Jia, Hongyang Li, Andreas Geiger, Xiangyu Yue, and Li Chen. ReSim: Reliable World Simulation for Autonomous Driving. In *Advances in Neural Information Processing Systems*, pages 167710–167741. Curran Associates, Inc., 2025. 27

[86] Jiazhi Yang, Kunyang Lin, Wencong Zhang, Jinwei Li, Tianwei Lin, Longyan Wu, Ya-Qin Zhang, Hao Zhao, Ping Luo, Zhizhong Su, Hongyang Li, Xiangyu Yue, and Li Chen. Self-Improving Robot Policy with Compositional World Model. In *Proceedings of Robotics: Science and Systems*, 2026. 3, 26

[87] Lijin Yang, Zhongzhan Huang, Jianing Huang, Shu Liu, and Hao Yang. Judge, Then Act: A Critic-Centric Vision Language Action Framework for Autonomous Driving. In *Proceedings of the 10th Conference on Robot Learning*, 2026. 3, 5, 27

[88] Xunjian Yin, Xinyi Wang, Liangming Pan, Li Lin, Xiaojun Wan, and William Yang Wang. Gödel Agent: A Self-Referential Agent Framework for Recursively Self-Improvement. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 27890–27913. Association for Computational Linguistics, 2025. 2

[89] Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason E Weston. Self-Rewarding Language Models. In *Proceedings of the 41st International Conference on Machine Learning*, pages 57905–57923. PMLR, 2024. 3

[90] Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. STaR: Bootstrapping Reasoning With Reasoning. In *Advances in Neural Information Processing Systems*, pages 15476–15488. Curran Associates, Inc., 2022. 3

[91] Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. Darwin Gödel Machine: Open-Ended Evolution of Self-Improving Agents. In *International Conference on Learning Representations*, pages 104223–104294, 2026. 2, 3

[92] Qi Zhang, Shaopeng Zhai, Shengzhe Zhang, Litao Liu, Tianyi Zhang, Fuxian Huang, and Ming Zhou. A Generalist Pair-wise Progress Critic Model for Vision-Language-Action Robots. In *Proceedings of the 43rd International Conference on Machine Learning*, pages 160301–160316. PMLR, 2026. 2, 3, 26

[93] Andrew Zhao, Yiran Wu, Tong Wu, Quentin Xu, Yang Yue, Matthieu Lin, Shenzhi Wang, Qingyun Wu, Zilong Zheng, and Gao Huang. Absolute Zero: Reinforced Self-play Reasoning with Zero Data. In *Advances in Neural Information Processing Systems*, pages 105816–105879. Curran Associates, Inc., 2025. 2, 3

[94] Yinuo Zhao, Huiqian Jin, Lechun Jiang, Xinyi Zhang, Kun Wu, Pei Ren, Zhiyuan Xu, Zhengping Che, Lei Sun, Dapeng Wu, Chi Harold Liu, and Jian Tang. Real-world Reinforcement Learning from Suboptimal Interventions. arXiv preprint arXiv:2512.24288, 2025. 26

[95] Zhijian Zhou, Zesheng Ye, Zhaorun Chen, Bo Li, and Feng Liu. CELEUS: Certifiable and Efficient LLM Evaluation via E-Processes. In *Advances in Neural Information Processing Systems*, 2026. 3

[96] Fangqi Zhu, Zhengyang Yan, Zicong Hong, Quanxin Shou, Xiao Ma, and Song Guo. WMPO: World Model-based Policy Optimization for Vision-Language-Action Models. In *International Conference on Learning Representations*, pages 62486–62502, 2026. 3, 26

[97] Morui Zhu, Deyuan Qu, Qi Chen, Kentaro Oguchi, and Qing Yang. When the Score Becomes the Target: Rethinking Metric Validity in Autonomous Driving. arXiv preprint arXiv:2609.34440, 2026. 3

[98] Jialv Zou, Shaoyu Chen, Bencheng Liao, Zhiyu Zheng, Yuehao Song, Lefei Zhang, Qian Zhang, Wenyu Liu, and Xinggang Wang. DiffusionDriveV2: Reinforcement Learning-Constrained Truncated Diffusion Modeling in End-to-End Autonomous Driving. arXiv preprint arXiv:2512.07745, 2025. 3, 27

A. Method details

Algorithm 1 lists one generation of the sealed variant, with $n_0 = 10$. In the in-loop variant the differences of line 5 come from J_{in} on $\mathcal{D}_{\text{in}}$, and line 12 is unchanged.

A.1. The sequential test

The confidence sequence. The test of Sec. 3.3 reads paired differences $d_1, d_2, \cdots \in [-1, 1]$ and tests their mean Δ (we drop the generation index). Let $u_i = (d_i + 1)/2 \in [0, 1]$, and for a candidate mean $m \in (-1, 1)$ let $\tilde{m} = (m + 1)/2$. The bets are computed from earlier pairs only:

$$\begin{aligned} \hat{\mu}_{i-1} &= \frac{\frac{1}{2} + \sum_{j<i} u_j}{i}, \\ \hat{\sigma}^2_{i-1} &= \frac{\frac{1}{4} + \sum_{j<i} (u_j - \hat{\mu}_{j-1})^2}{i}, \\ \lambda_i &= \sqrt{\frac{2\log(2/\alpha)}{\hat{\sigma}^2_{i-1}\, i \log(1+i)}}. \end{aligned} \tag{5}$$

Each bet is truncated with $c = 0.75$,

$$\lambda_i^+(m) = \min\{\lambda_i, c/\tilde{m}\}, \quad \lambda_i^-(m) = \min\{\lambda_i, c/(1-\tilde{m})\}, \tag{6}$$

and the capital of a bettor against the candidate m is

$$\begin{aligned} K_t(m) = \frac{1}{2} \prod_{i=1}^{t} \big(1 + \lambda_i^+(m)(u_i - \tilde{m})\big) \\ + \frac{1}{2} \prod_{i=1}^{t} \big(1 - \lambda_i^-(m)(u_i - \tilde{m})\big). \end{aligned} \tag{7}$$

The confidence sequence keeps the candidates against which the capital has stayed small [75]:

$$C_t = \Big\{ m : \max_{r \le t} K_r(m) < 1/\alpha \Big\}, \tag{8}$$

with $L_t = \inf C_t$ and $U_t = \sup C_t$.

Lemma A.1 (Coverage)**.** *If $\mathbb{E}[d_i \mid d_1, \ldots, d_{i-1}] = \Delta$ for every i, then $\Pr(\exists\, t \ge 1 : \Delta \notin C_t) \le \alpha$.*

Proof. Set $m = \Delta$. Since $u_i - \tilde{m} \ge -\tilde{m}$ and $\lambda_i^+ \le c/\tilde{m}$, every factor of the first product in Equation (7) is at least $1 - c > 0$, and likewise for the second. The bets depend on $d_1, \ldots, d_{i-1}$ only and $\mathbb{E}[u_i - \tilde{m} \mid d_1, \ldots, d_{i-1}] = 0$, so each factor has conditional mean one. $K_t(\Delta)$ is therefore a nonnegative martingale with $K_0 = 1$, and Ville's inequality [56] gives $\Pr(\sup_t K_t(\Delta) \ge 1/\alpha) \le \alpha$.

Proposition 1 follows: an acceptance at t with $\Delta \le 0$ or a rejection with $\Delta \ge 0$ implies $\Delta \notin C_t$. The bound holds for any rule that chooses when to read C_t, including a first reading at pair 10 and a stop at the budget.

Algorithm 1 One generation of SEALEDLOOP

Require: policy π_g, improver $\mathcal{I}$, in-loop judge J_{in} with data $\mathcal{D}_{\text{in}}$, sealed pool $\mathcal{P}$ with true score s, level α, budget N, first reading n_0

1: **Seal:** fix $\mathcal{P}$, s and the rule that draws $c_1, c_2, \ldots$ from $\mathcal{P}$
2: **Improve:** $\pi_{g+1} \leftarrow \mathcal{I}(\pi_g; J_{\text{in}})$ ▷ no access to $\mathcal{P}$, s or the draws
3: **Decide:** decision $\leftarrow$ undecided
4: **for** $t = 1, \ldots, N$ **do**
5: run π_g and π_{g+1} on c_t; $d_t \leftarrow s_{c_t}(\pi_{g+1}) - s_{c_t}(\pi_g)$
6: compute $[L_t, U_t]$ from $d_1, \ldots, d_t$ ▷ Equation (8)
7: **if** $t \ge n_0$ and $L_t > 0$ **then** decision $\leftarrow$ accept; **break**
8: **else if** $t \ge n_0$ and $U_t < 0$ **then** decision $\leftarrow$ reject; **break**
9: **end if**
10: **end for**
11: **Reveal:** keep π_{g+1} if decision is accept, otherwise roll back to π_g; the improver receives only this outcome
12: record $\hat{\Delta}_g$ from J_{in} on $\mathcal{D}_{\text{in}}$ and Δ_g from s on $\mathcal{P}$

Lemma A.2 (Sealing gives the assumption)**.** *Let the conditions $c_1, c_2, \ldots$ be drawn independently from a distribution P on the pool, independently of (π_g, π_{g+1}), and let the two policies be run on each condition with fresh randomness. Then $\mathbb{E}[d_i \mid d_1, \ldots, d_{i-1}] = \Delta_g$ with $\Delta_g = \mathbb{E}_{c \sim P}[s_c(\pi_{g+1}) - s_c(\pi_g)]$.*

Proof. Given the two policies, the pairs $d_1, d_2, \ldots$ are independent and identically distributed with mean Δ_g, so conditioning on earlier pairs does not change the mean of d_i.

The independence from π_{g+1} is what sealing provides. An improver that had seen the conditions could fit the candidate to them; the d_i would then have a mean above the gain on new conditions, and the test would certify a gain that exists only on the conditions it reads.

Corollary A.1 (Several generations)**.** *Let a loop run K generations, each decided by the test at level α, and let the assumption of Lemma A.1 hold in every generation given all earlier generations. Let W be the number of generations in which the test accepts a candidate with $\Delta_g \le 0$ or rejects one with $\Delta_g \ge 0$. Then $\mathbb{E}[W] \le \alpha K$ and $\Pr(W \ge 1) \le \alpha K$.*

Proof. Given the earlier generations, generation g is decided wrongly with probability at most α by Proposition 1. Taking expectations and summing over g gives $\mathbb{E}[W] \le \alpha K$, and $\Pr(W \ge 1) \le \mathbb{E}[W]$.

Proposition A.1 (An in-loop rule cannot see the transfer gap)**.** *Let the pool contain no condition of $\mathcal{D}_{\text{in}}$, and let an acceptance rule, possibly randomized, depend only on what the two policies do on $\mathcal{D}_{\text{in}}$. Then for any such data there are two score functions on the pool, one with $\Delta_g > 0$ and one with $\Delta_g < 0$, under which the rule behaves identically. If it accepts with probability p, it accepts a loss with probability p in one case and rolls back a gain with probability $1 - p$ in the other, so the two error probabilities sum to one.*

Sampling	False acceptances		≥ 1 in five
	Worst setting	Pooled	generations
Manipulation			
By start state (ours)	1.2%	0.23%	5.7%
Uniform, no replacement	1.9%	0.66%	9.1%
With replacement	2.1%	0.68%	10.3%
Driving			
Per-log result decides (ours)		0 of 1,499	
Per-frame test alone		10 of 1,499	

Table 5. False acceptances under the sampling used. Share of simulated generations without a real gain that the test accepts. Last column: at least one acceptance in five such generations, at the worst setting's rate.

Proof. The rule's input does not involve s_c for c in the pool. Choose $s_c(\pi_{g+1}) - s_c(\pi_g) = \delta$ on every pool condition in the first case and $-\delta$ in the second, for some $\delta \in (0, 1]$. The input, and hence the distribution of the decision, is the same in both.

In the decomposition of Equation (3), the two cases have the same $\hat{\Delta}_g$ and Δ_g^{in}, hence the same judge error E_g, and differ only in the transfer gap T_g. No improvement of the in-loop judge separates them; a reading on the pool does.

Size of a certified gain. An acceptance at a stopping time τ certifies the sign of Δ_g. The mean of the pairs read up to τ overstates its size, because the test stops when the differences seen so far are large. The interval $[L_\tau, U_\tau]$ remains valid by Lemma A.1, and we report sealed gains from all pairs of the generation or from the whole pool.

A.2. Parameters, sampling and sealing

Test parameters. Every test in the paper uses the betting confidence sequence [75] at $\alpha = 0.05$ with a threshold of zero. Bets are truncated as in that method, and the test reads its interval after every pair from pair 10 on. A candidate undecided at the budget is not kept.

Pairs and sampling. On manipulation, a pair is one start condition (one of the task's 50 start states with an environment seed) on which both policies are run. The sealed pool of a task holds 200 seeds per start state, 10,000 conditions in all. Each generation draws a new set that is disjoint from all earlier ones and stratified by start state, up to a budget of 1,000 pairs. On driving, a pair is one log scored for both policies, and its paired difference is the mean over the frames of the log. The test reads the logs of the sealed pool once, in a fixed pseudo-random order. We also run the test on single frames; frames of one log are correlated, so the per-log result decides when the two disagree. In a bootstrap over logs, the per-log test detects a gain of about 0.030 PDMS with probability 0.8 on the main pool of 280 logs.

Error rate under the sampling used. Proposition 1 and Corollary A.1 assume independent pairs (Lemma A.2). Our manipulation pairs are drawn without replacement from a finite pool, stratified by start state, and driving frames share logs, so we measure the error rate by simulation (Tab. 5). Manipulation pools are built from the measured success rate of every start state and the measured agreement between seeds (216 settings, 189,000 simulated generations); on driving, the two scores of a pair of start policies are swapped in a random half of the logs. With our draw, the test accepts a candidate without a real gain in at most 1.2% of generations in any setting, below α; five such generations contain an acceptance with probability 5.7%, against a union bound of 25%. Under all three draws, no candidate that loses 0.05 or 0.10 is accepted (0 of 9,000), and a gain of 0.03 is accepted in 75% to 99% of generations at 1,000 pairs. On driving, the per-frame test alone accepts 10 of 1,499 pools without a gain and the per-log test none. Replayed with a confidence sequence for sampling without replacement [74], every recorded acceptance and rejection that can be re-read is unchanged (52 on manipulation and 50 on driving). Manipulation draws are disjoint across generations, so no earlier decision depends on the conditions a generation reads; on driving every generation reads the same frozen pool, from which the improver learns one decision per generation. The error rate of the test therefore stays below α under the sampling we use.

Isolation of the sealed judge. Every improver process runs with the files of the sealed judge masked from its file system, and the evaluation conditions of the two judges are disjoint. On driving, all 850 improver steps passed a check of file access; on manipulation a check of every generation found no access to them (114/114).

B. Manipulation

B.1. Tasks, policies and judges

The testbed uses three LIBERO-Object tasks [48]: placing the alphabet soup, the cream cheese and the BBQ sauce in the basket (tasks 1 to 3). The source model is the OpenVLA-OFT checkpoint for LIBERO-Spatial [35]. For each task we adapt it with LoRA on a few demonstrations and take the checkpoint whose success on calibration conditions lies between 0.50 and 0.80. On 1,000 sealed conditions the three start policies succeed at rates of 0.661 / 0.746 / 0.491. In every generation the policy runs 300 new start conditions per task, the in-loop judge selects the episodes it rates as successful, and the LoRA adapter is trained further on these episodes alone (rank 32, 2,000 steps, batch size 8, learning rate 10^{-4}). The in-loop judge is a success classifier that reads the final camera frames of an episode and is retrained every generation on its own labels. Its test reads 300 pairs on conditions excluded from collection. The sealed judge is the simulator's success predicate on the sealed pool

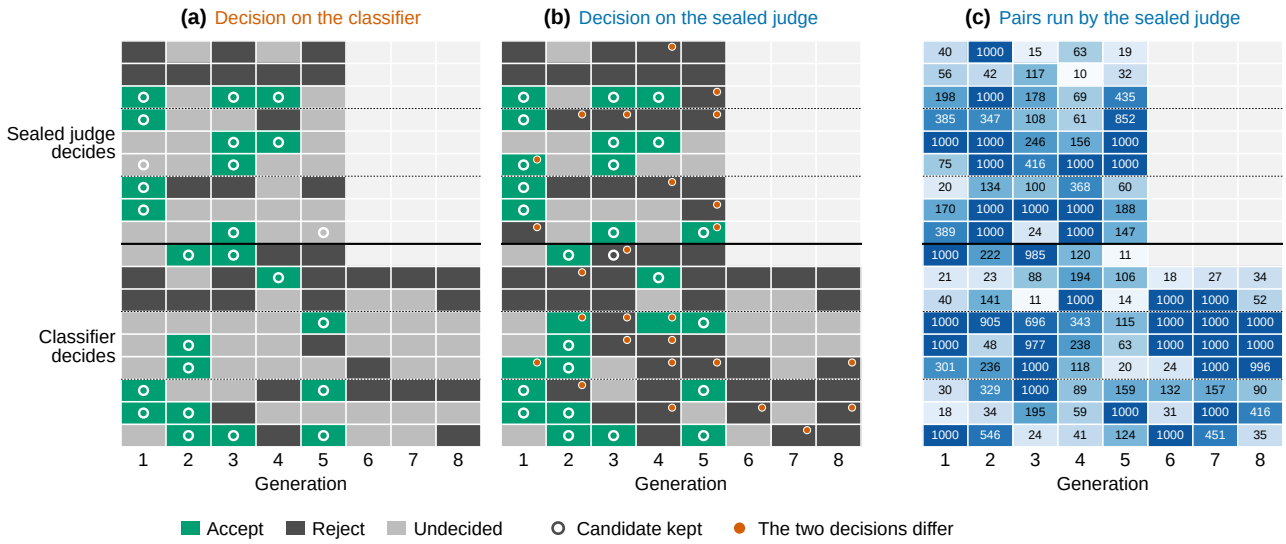


Figure 8. Decisions in every manipulation generation. Rows are the eighteen runs in the order of Fig. 3, with dashed lines between tasks. (a) Decision of the test on the classifier's verdicts. (b) Decision on the sealed judge; an orange dot marks the 26 generations in which the two differ. (c) Pairs the sealed judge ran before the test stopped.

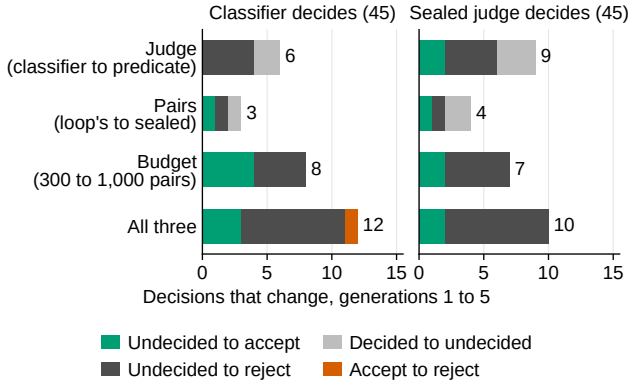


Figure 9. Judge, pairs and budget. Decisions of the first five generations that change when the judge, the pairs, the budget or all three are replaced by those of the sealed test.

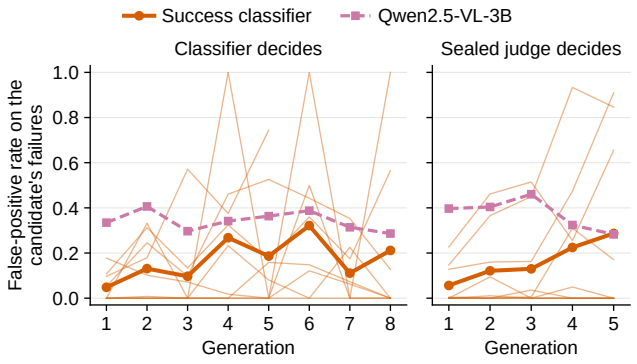


Figure 10. False-positive rate of the classifier by generation. Share of the candidate's failed episodes that the classifier labels as successes (thin lines: runs; thick line: mean). Dashed: the same rate for Qwen2.5-VL-3B, which is not trained in the loop.

of Appendix A. Three seeds per task and variant give eighteen runs of five generations. The classifier-decided runs that had kept no candidate certified as worse by the sealed judge continued to eight generations (8 runs, 24 generations).

B.2. Disagreements between the two decisions

Decisions of the two judges. Figure 8 shows both decisions for all 114 generations. They differ in 26 generations, 22 of them within the first five. In 25 of these the classifier's test is undecided at its 300 pairs while the sealed judge decides, and in 1 the classifier accepts a candidate that the sealed judge rejects. The sealed judge stops after a median of 191 pairs and uses its whole budget in 32 generations. In generations 6 to 8 the classifier kept no candidate (0 of 24). The two decisions thus differ mostly where the classifier's test cannot decide, and once where it keeps a policy that the sealed judge rejects.

Judge, pairs and budget. The two tests differ in the judge, in the pairs and in the budget. Figure 9 changes one factor at a time. Over the 45 classifier-decided generations, replacing the classifier by the success predicate on the same pairs changes 6 decisions, replacing the pairs changes 3, and raising the budget changes 8. The sealed-decided runs give 9, 4 and 7 of 45. The judge and the budget therefore matter about equally on this testbed. On the same pairs the classifier never accepts a candidate that the predicate rejects. Reading the loop's pairs once with a fixed-sample test on

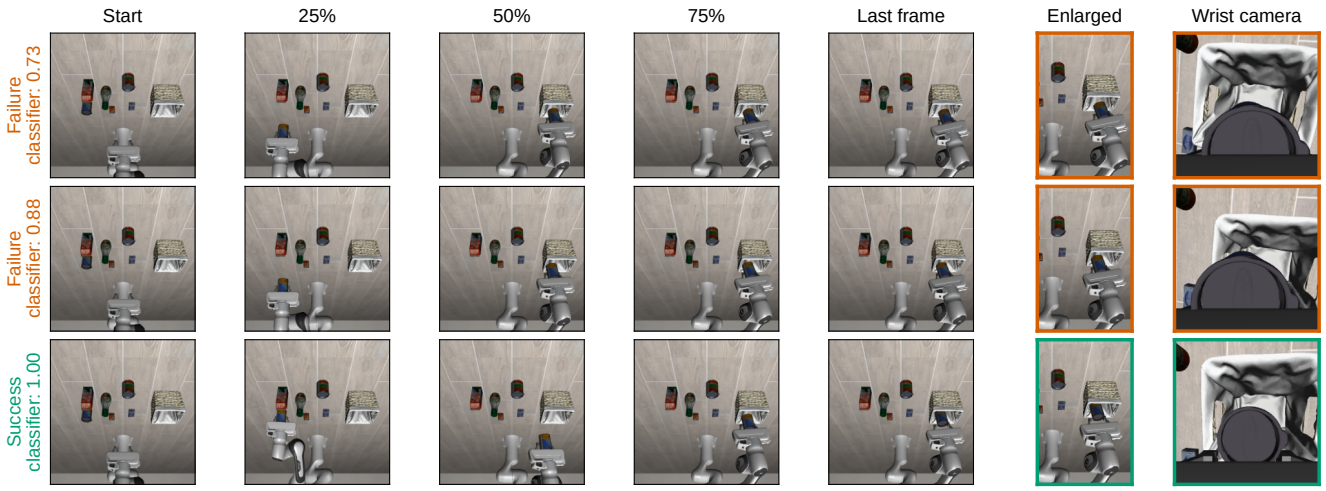


Figure 11. Failures that look like successes. Three episodes of one candidate on the same task: two failures that the classifier labels as successes (scores 0.73 and 0.88) and one success. In the failures the gripper still holds the can at the near edge of the basket when the episode ends; in the success the can is inside it. The final frames differ by a few pixels.

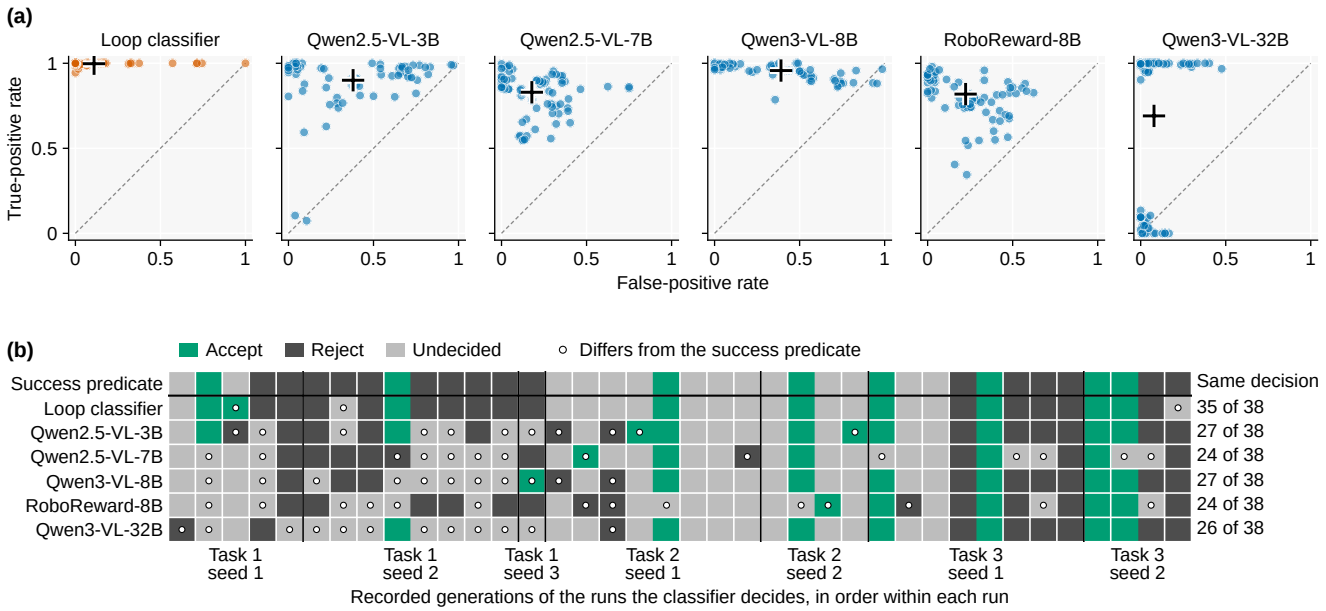


Figure 12. Six judges on the recorded episodes. (a) True-positive against false-positive rate, one point per episode set with both outcomes and a cross for all 22,800 episodes pooled. (b) Decision of the test on each judge's verdicts in the 38 recorded generations of the classifier-decided runs, with the decision on the success predicate in the top row. The loop classifier matches it most often.

the success predicate (exact McNemar, $\alpha = 0.05$) gives the sealed test's decision in 109 of 114 generations, and all 25 candidates it accepts are certified by the sealed test.

True success on the classifier's budget. A third variant keeps the improver of the classifier-decided runs, with the classifier choosing the training episodes and refreshed in the same way, and changes only the decision: the true success of the same 300 pairs enters one exact McNemar test (two-sided, $\alpha = 0.05$). It runs on the same three tasks with three seeds for five generations, and the sealed test is read in every generation without deciding. It kept 13 of 45 candidates; the sealed test certified all 13 as gains, and 1 certified gain was rolled back. The decision to keep or roll back equals that of the sealed test in 44 generations, against 1 worse policy kept and 3 gains lost when the classifier decides. At the classifier's budget of 300 pairs, true success thus reproduces the sealed decisions, so the wrong decisions of the classifier-decided runs come from the judge and not from the budget.

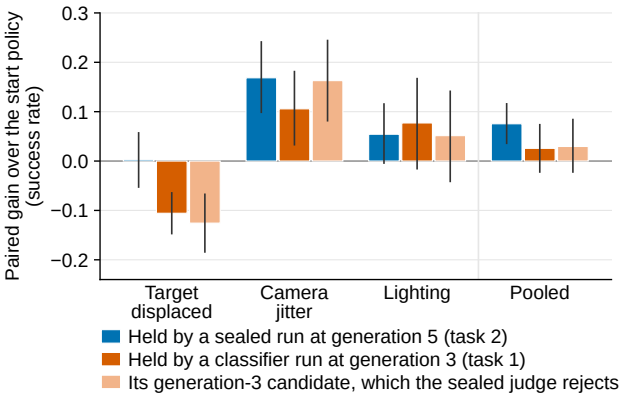


Figure 13. Three policies on perturbed start conditions. Paired gain over the start policy by kind of perturbation and pooled (95% intervals). Two of the three lose when the target is displaced.

		Sealed	Vision-language judge		Success predicate	
Task	Gen.	gain	Decision	Pairs	Decision	Pairs
1	1	+0.150	Accept	62	Accept	173
	2	+0.058†	Undecided	300	Undecided	300
	3	−0.800	Reject	120	Reject	11
	4	−0.438	Undecided	300	Reject	29
	5	−0.039	Undecided	300	Undecided	300
2	1	+0.029	Undecided	300	Undecided	300
	2	+0.118†	Undecided	300	Accept	26
	3	+0.016	Undecided	300	Undecided	300
	4	−0.385	Undecided	300	Reject	162
	5	−0.006	Undecided	300	Undecided	300
3	1	−0.120	Reject	100	Reject	210
	2	+0.168	Accept	111	Accept	39
	3	+0.010	Undecided	300	Undecided	300
	4	+0.095	Accept	253	Undecided	300
	5	−0.194	Reject	30	Reject	41

Table 6. A vision-language judge inside the loop. Decisions of the test on the verdicts of Qwen2.5-VL-3B, which decides, and on the success predicate for the same pairs. †Certified gains that the run rolls back.

B.3. Errors of the classifier

Classifier error over generations. Figure 10 follows the classifier's false-positive rate on the failures of each candidate. In the classifier-decided runs its mean is 0.05 at the first generation and peaks at 0.32 at generation 6; it exceeds 0.2 in 31 of all 114 generations. The rate does not rise steadily from one generation to the next. The mean rate of Qwen2.5-VL-3B, which is not trained in the loop, lies between 0.28 and 0.46 and is higher than the classifier's in all but one generation of the two panels. The classifier's error thus comes and goes with the failures of each candidate, and a model that is not trained in the loop is not more reliable on them.

Accepted failures. Figure 11 shows one generation whose failures look like successes; a video on the project page shows the three episodes in full. Of the candidate's 196 failures, the classifier accepts 65, and in none of these did the object ever reach the goal region. The classifier frozen before the loop gives the two failures shown scores of 0.96 and 0.95, so the new policy found an error that the classifier already had, and retraining did not produce it. A second pattern appears in 3 generations: after a generation whose collected episodes contain no failure, the retrained classifier accepts failures of the next candidate that the classifier frozen before the loop rejects. The classifier's wrong acceptances thus arise both without retraining and through it.

B.4. Other judges

Off-the-shelf judges. We re-scored the recorded evaluation episodes with four vision-language models and a robot reward model [37] and ran the test on each judge's verdicts (Fig. 12). Each vision-language model sees nine frames of the episode from each of the two cameras and one fixed question, whether the instruction was completed, and its score is the probability of the answer yes; the reward model gives its own progress score. Thresholds are fitted per task on episodes of the start policy. The loop's classifier reaches the success predicate's decision in 35 of 38 generations. The other judges reach it in 27 (Qwen2.5-VL-3B), 24 (Qwen2.5-VL-7B), 27 (Qwen3-VL-8B), 24 (RoboReward-8B) and 26 (Qwen3-VL-32B). On the 31 recorded generations of the sealed-decided runs the classifier again agrees most often, in 25 against at most 23. Pooled over all episodes the classifier has a true-positive rate of 1.00 at a false-positive rate of 0.11. Qwen3-VL-32B has the lowest false-positive rate (0.08) but a true-positive rate of 0.69, and it takes 722 s per thousand episodes against 371 s for Qwen3-VL-8B. The off-the-shelf judges remove the two failure patterns only in part. Of the 36 failures of one generation that follows a refresh without failures, the classifier accepts 100% and every other judge 0%. On the 196 failures of the generation in Fig. 11, the classifier accepts 33%; two of the five other judges accept fewer (Qwen2.5-VL-7B 23%, Qwen3-VL-32B 3%) and three accept more (Qwen2.5-VL-3B 69%, Qwen3-VL-8B 34%, RoboReward-8B 42%).

A vision-language model as the in-loop judge. In three further runs, one per task, Qwen2.5-VL-3B replaces the classifier: its verdicts select the training episodes and feed the test, with a fixed threshold (Tab. 6). It keeps 3 of 15 candidates and no candidate that the sealed judge rejects, and it rolls back 2 gains that the sealed judge certifies. Its decision equals the predicate's on the same pairs in 11 of 15 generations. A classifier frozen before the loop reaches 13 of 15 on these pairs. The model's test decides within the budget in 6 of 15 generations (median stop at 300 pairs) and the predicate's in 8 of 15 (210). Of the 5 rolled-back candi-

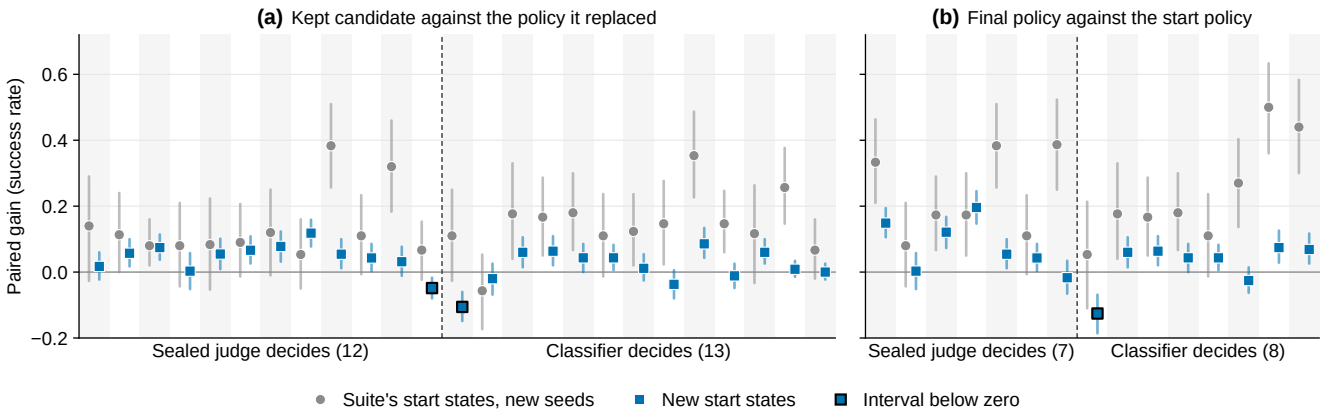


Figure 14. Kept policies on conditions that no run used. (a) Each kept candidate against the policy it replaced. (b) The final policy of each run against its start policy; the 2 sealed-decided runs and the 1 classifier-decided run that kept no candidate end at their start policy and are not drawn. Paired gains with 95% intervals on the suite's start states with new seeds and on new start states; a black edge marks an interval below zero.

dates with a positive sealed gain, the predicate accepts 1 on the same pairs and is undecided on the other 4. A vision-language model inside the loop thus keeps no loss, but it gives up certified gains, mostly because its test runs out of budget.

B.5. Reach of a certified gain

A certified gain is a statement about the pool the pairs were drawn from. We tested how far it reaches in two ways. First, three policies were compared with their start policy on start conditions with a displaced target, a moved camera or changed lighting (Fig. 13). The policy kept by a sealed-decided run gains $+0.075$ pooled over the three kinds, and the test accepts it there as well. The policy held by a classifier-decided run at generation 3 and its next candidate, which the sealed judge rejects, gain $+0.026$ and $+0.030$; the test is undecided on both, and both lose when the target is displaced (-0.106 and -0.126). Second, all 12 candidates kept by the sealed judge and all 13 kept by the classifier were run on 300 conditions per task with the suite's start states and new seeds, and on 350 new start states per task with the target displaced and rotated (Fig. 14). On the suite's start states the 95% interval of no kept candidate lies below zero, and the final policies of the nine runs of each variant gain on average $+0.18$ (sealed) and $+0.21$ (classifier) over their start policies, with a gain of zero for the runs that kept no candidate. On the new start states the gains shrink to $+0.06$ and $+0.02$, the interval of one kept candidate of each variant lies below zero, and one classifier-decided run ends -0.13 below its start policy. A certified gain holds for the start states of its pool and carries over to new ones only in part.

C. Driving

C.1. Data, policies and judges

Data and start policy. The testbed is NAVSIM [15] with DiffusionDrive [45] as the policy. All frames from Boston are excluded from the loop, which sees only Las Vegas, Pittsburgh and Singapore. The main start policy is trained from scratch on 20,000 training-city frames for 20 epochs, so that it can still improve. It relies on its camera: with the image replaced by a black or a gray frame, its PDMS on 3,731 Boston frames changes by -0.126 and -0.107. We chose this policy among six training configurations as the one with the highest PDMS on 3,731 Boston frames (0.698). These frames are 22% of the sealed pool. As a check, we ran each generation's sealed test again on the other 12,954 frames, from 229 logs that played no part in the choice. The decision is the same for 43 of the 45 candidates of the nine main runs; 2 rejections become undecided and no candidate is accepted. The gain is negative for 43 candidates, and its 95% interval over logs lies below zero for 42 and above zero for none. The three candidates that the critic kept change the score on these frames by -0.026, -0.012 and -0.023. Choosing the start policy on part of the sealed pool therefore does not change the outcome of the runs.

Improver and critic. In every generation the policy draws 20 trajectory proposals on 20,000 new unlabeled frames, the critic picks one proposal per frame, and the policy is trained for five epochs on these picks, mixed one to one with expert frames. The fifth of the frames on which the critic's pick scores furthest above the policy's own first choice is duplicated. The in-loop judge is this critic, a network that predicts PDMS from frozen encoder features and a trajectory, fitted before the loop on 20,000 frames with the true

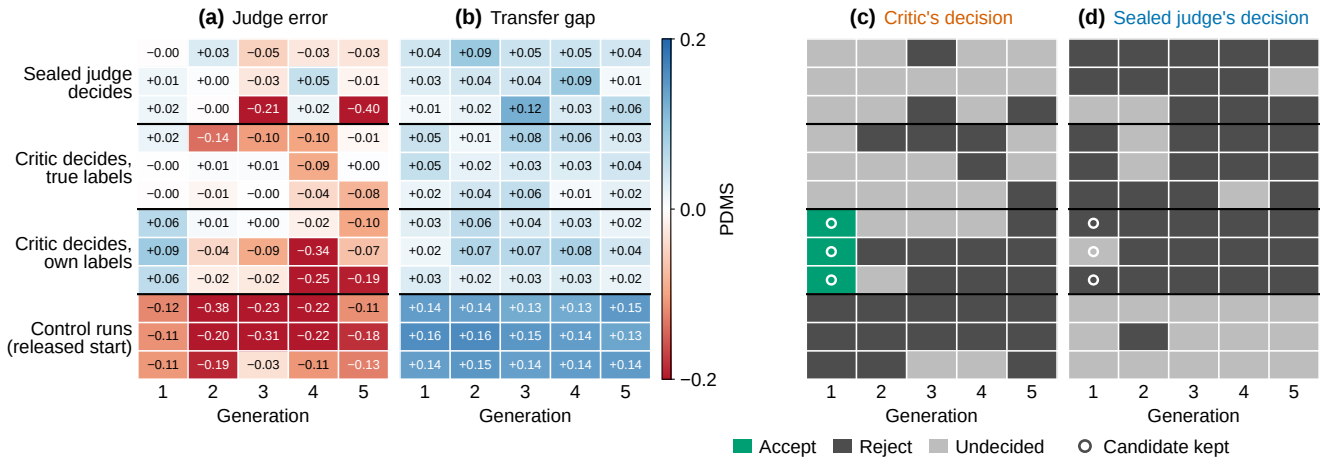


Figure 15. Every driving run. Rows are runs, columns are generations. (a) Judge error and (b) transfer gap of each candidate. (c) Decision of the test on the critic's scores and (d) on the sealed judge; a ring marks a candidate that the run kept. The transfer gap is positive in every generation, and the critic keeps three candidates, none of which the sealed judge accepts.

PDMS of their proposals. It is refreshed every generation in one of two ways. With *true labels*, 2,000 of the generation's frames receive the true PDMS of their proposals and the critic is fitted again on all labels so far. With *own labels*, the critic is trained further to prefer its own pick over the other proposals of each frame.

Judges and runs. The critic's test reads its scores for both policies on 5,000 training-city frames disjoint from all training data. Their logs are disjoint from the logs of the improver's frames and of the critic's labels; the start policy was trained on other frames of the same logs. The sealed judge is the true PDMS on 16,685 Boston frames from 280 logs. Gains are means per log; scores of single policies are means over frames. Three seeds of three variants (sealed judge decides, critic decides with either refresh) give nine main runs of five generations. Three control runs, one per variant, start from the released DiffusionDrive checkpoint and are judged on 3,731 Boston frames from 51 logs.

Learning rate. To set the improver's learning rate, we trained the start policy on its own first choice in place of the critic's pick. At learning rates of 3×10^{-5}, 1×10^{-4} and 6×10^{-4} its PDMS on 5,000 frames changes by -0.001, $+0.000$ and -0.002, and the test accepts none of the three. All runs use the largest rate.

C.2. The critic and the judge gap

The critic before the first generation. Table 7 compares the critic with the true PDMS before any policy has been trained on its picks. On the loop's own frames and on unseen logs of the training cities it ranks proposals well, and its pick is slightly better than the policy's own choice ($+0.010$ PDMS). In Boston the rank correlation falls by almost half, the false-positive rate rises from 0.164 to 0.383,

Frames	n	Rank corr.	Top-1	FPR	Pick − own
Main runs (start policy trained from scratch)					
Loop's own frames	5,000	0.74	0.67	0.16	$+0.010 \pm 0.003$
Unseen logs	2,000	0.73	0.68	0.18	$+0.006 \pm 0.005$
Boston (sealed pool)	16,685	0.40	0.25	0.38	-0.042 ± 0.004
Control runs (released checkpoint)					
Loop's own frames	5,000	0.75	0.73	0.18	$+0.001 \pm 0.002$
Unseen logs	2,000	0.75	0.71	0.17	$+0.002 \pm 0.003$
Boston (sealed pool)	3,731	0.48	0.39	0.47	-0.073 ± 0.008

Table 7. The critic before the first generation. Rank correlation of its scores with the true PDMS, share of frames where its pick has the highest true PDMS (top-1), share of zero-PDMS proposals it scores above its threshold (FPR), and true PDMS of its pick minus that of the policy's first choice ($\pm$ standard error over logs). Unseen logs are training-city logs outside the loop.

and its pick is worse than the policy's own choice (-0.042). The control runs show the same pattern ($+0.001$ against -0.073). The critic is therefore already unreliable outside its training cities before the loop starts.

Results by run. Figure 15 gives the two terms of the judge gap and both decisions for all 45 main and 15 control generations. The transfer gap is positive in all 60 of them, +0.04 PDMS on average in the main runs and +0.14 in the control runs. The judge error averages -0.05 and -0.18: after the first generation the critic mostly reports less gain than the candidate has on the loop's own frames. In the control runs the judge error is -0.376 to -0.034 and the transfer gap $+0.125$ to $+0.160$ across generations: the critic underreports in every one, and the whole positive gap is transfer. It keeps 3 candidates, all at the first generation of the runs that refresh it on its own labels. The sealed judge accepts 0 of the 45 main candidates, rejects 38 and is undecided on 7. In the control runs the critic rejects 13 of 15 candidates,

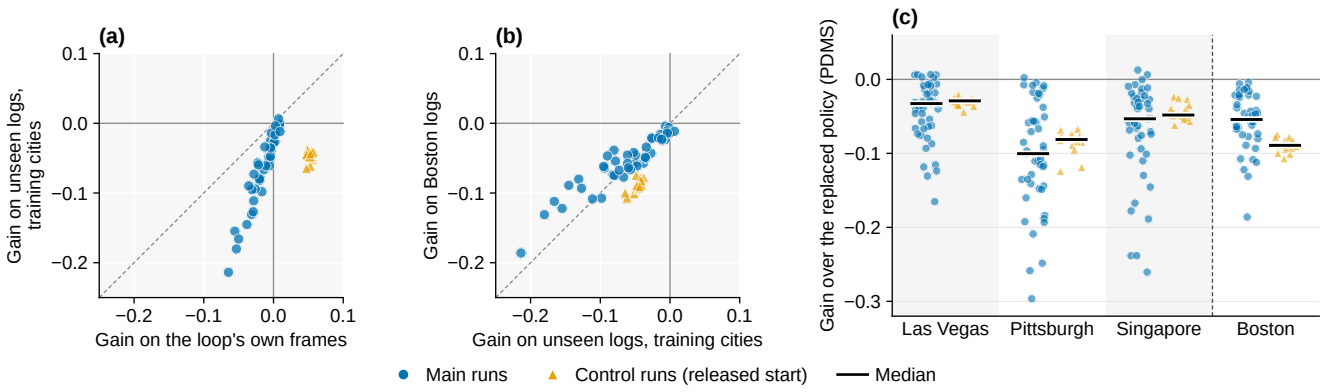


Figure 16. Where the gain is lost. True gain of every candidate over the policy it would replace. (a) On unseen logs of the training cities against the loop's own evaluation frames. (b) On Boston logs against unseen logs of the training cities. (c) By city, with medians. Most of the loss appears on unseen logs of the training cities.

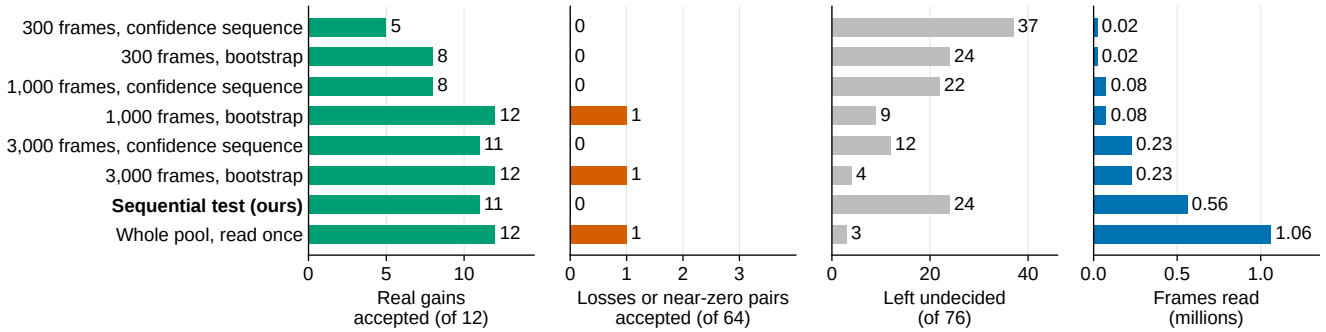


Figure 17. Acceptance rules on 76 pairs with known gains. For each rule: real gains accepted, losses or pairs near zero accepted, pairs left undecided and frames read. Fixed-size rules read a sample once, with the confidence sequence or a 95% bootstrap over logs. The sequential test accepts no loss and no pair near zero and reads about half of what the whole pool costs.

and the sealed judge rejects 1 and is undecided on 14, since its pool holds only 51 logs. Refreshing does not repair the critic outside its cities: in the control runs the rank correlation of the refreshed critics with the true score in Boston is 0.132 to 0.439, below the value before the loop (Tab. 7).

C.3. Origin of the transfer gap

Training on the best proposals. We repeated the first generation with targets chosen by the true score in place of the critic. On each of 20,000 training frames the target is the proposal with the highest PDMS, 0.965 on average against 0.887 for the policy's own first choice. At learning rates of 2×10^{-5}, 1×10^{-4} and 6×10^{-4} the candidate changes PDMS on the 16,685 Boston frames by -0.018 [-0.021, -0.014], -0.008 [-0.012, -0.004] and -0.108 [-0.117, -0.099] (95% bootstrap intervals over logs). Better targets alone thus do not give the candidate a gain in Boston.

Loss on unseen logs of the training cities. The sealed pool differs from the loop's frames in two ways: its logs are new and its city is new. To separate the two, we scored every candidate on 8,415 frames from logs of the training cities that no run used (Fig. 16). 44 of 45 main candidates lose on these frames, with a median of -0.054 PDMS against -0.054 in Boston. Reweighting these frames to the city mix of the loop's own evaluation frames leaves the loss in place: the median is -0.043 for the main candidates (-0.186 to $+0.006$) and -0.037 for the control candidates (-0.053 to -0.029), so the loss on unseen logs does not come from a different share of the three cities. The control candidates gain $+0.053$ on the loop's own evaluation frames and lose -0.045 on unseen logs of the same cities and -0.089 in Boston, so 70% of the drop is present before the city changes. The median gain of the main candidates is -0.033 in Las Vegas, -0.100 in Pittsburgh, -0.053 in Singapore and -0.054 in Boston. A critic without error would not have helped the control runs: the test on the true PDMS of the loop's own evaluation frames accepts 15/15 control generations per frame and 8/15 per log, and all of them lose in Boston. On the main runs the same test accepts 0/45. Applied to the unseen logs of the training cities, the same test keeps none of the 60 candidates and rejects 20. Most of the transfer gap thus opens between logs of the same cities, where no judge that reads the loop's own frames can see it.

Rule	Gains accepted	Other pairs accepted		Frames read	Accepted, no-gain pools	
		Losses	Near zero		Mean	Worst pair
Sequential test (ours)	11.2	0	0	53%	0.0%	0.0%
50 logs once, bootstrap	10.8	0	0.17	22%	0.8%	2.5%
50 logs once, confidence sequence	1.1	0	0	22%	0.0%	0.0%
17 logs once, bootstrap	8.4	0.01	0.22	8%	2.2%	4.2%
5 logs once, bootstrap	6.1	0.25	0.45	2%	6.3%	10.7%
Bootstrap after every 10 logs	12.0	0.06	1.16	13%	5.9%	11.4%

Table 8. Acceptance rules on one unit. Every rule reads whole logs of each of the 76 pairs in 500 random orders; entries are means per pass over the pairs. Frames read are a share of the whole pools. Last two columns: share accepted of the 18,972 pools without a gain, built by swapping the two scores in a random half of the logs and keeping the pools whose mean gain is not positive.

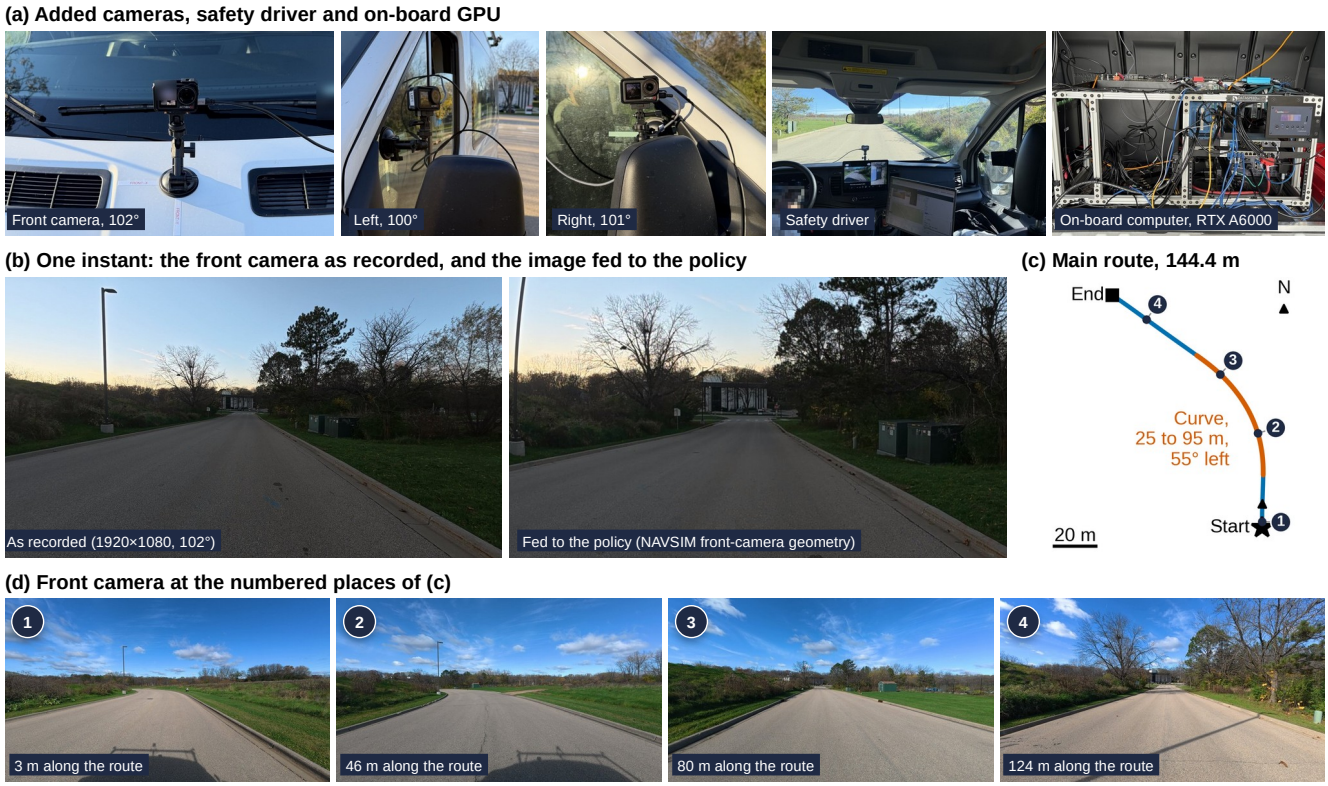


Figure 18. Vehicle setup and route. (a) The three added cameras, the safety driver and the on-board GPU; both versions plan from the front camera only. (b) One instant as recorded by the front camera and as fed to the policy, reprojected to the geometry of the NAVSIM front camera. (c) Main route with its left curve. (d) Front camera at the numbered places of (c).

C.4. The sequential test on known gains

Pairs with known gains. To test the sequential test on real gains, we ran it on 76 pairs of policies whose gain is known from the whole pool: the 60 candidates, 15 pairs among 6 start policies trained from scratch with different amounts of data, and the main start policy against the released checkpoint. A pair counts as a real gain or loss when the 95% bootstrap interval over the logs of the whole pool lies above or below zero. This gives 12 gains, 59 losses and 5 pairs near zero (Fig. 17). The sequential test accepts 11 of the 12 gains and no other pair. The one gain it misses (+0.019 PDMS) stays undecided with the whole pool read, and the accepted ones need 2,098 to 14,452 frames each. Over the 76 pairs it reads 561,665 frames, counting every frame of the logs it opened. In the nine main runs the sequential test reads 354,754 of the 750,825 frames that scoring the whole pool in every generation would need (47%). It decides 38 generations after a median of 5,316 frames and leaves 7 undecided. The test thus accepts nearly every real gain and nothing else, at about half the cost of scoring the whole pool.

Sequential test and fixed sample. The sequential test adds

Seed	Gen.	Kept before	Sealed gain	Decision		
				Sealed	Own frames	Critic
0	1	0	+0.031	Accept	Accept	Undecided
0	2	1	+0.013	Undecided	Accept	Undecided
0	2	0	+0.043	Accept	Accept	Undecided
0	3	2	+0.009	Undecided	Accept	Undecided
0	3	1	+0.017	Undecided	Accept	Undecided
0	3	0	+0.041	Accept	Accept	Undecided
0	4	3	−0.004	Undecided	Undecided	Undecided
0	4	1	+0.010	Undecided	Undecided	Undecided
0	4	0	+0.029	Accept	Accept	Undecided
0	5	3	−0.004	Undecided	Undecided	Undecided
0	5	1	+0.011	Undecided	Undecided	Undecided
0	5	0	+0.036	Accept	Accept	Undecided
1	1	0	+0.042	Accept	Accept	Undecided
1	2	1	+0.007	Undecided	Accept	Undecided
1	2	0	+0.039	Accept	Accept	Undecided
1	3	2	−0.001	Undecided	Undecided	Undecided
1	3	1	+0.001	Undecided	Accept	Undecided
1	3	0	+0.034	Accept	Accept	Undecided
1	4	2	+0.005	Undecided	Undecided	Undecided
1	4	1	+0.008	Undecided	Accept	Undecided
1	4	0	+0.039	Accept	Accept	Undecided
1	5	2	+0.005	Undecided	Undecided	Undecided
1	5	1	+0.004	Undecided	Accept	Undecided
1	5	0	+0.039	Accept	Accept	Undecided
2	1	0	+0.034	Accept	Accept	Accept
2	2	1	+0.008	Undecided	Accept	Undecided
2	3	2	+0.007	Undecided	Undecided	Undecided
2	3	1	+0.010	Undecided	Accept	Undecided
2	4	2	+0.009	Undecided	Undecided	Undecided
2	4	1	+0.017	Undecided	Accept	Undecided
2	5	2	+0.007	Undecided	Accept	Undecided
2	5	1	+0.016	Undecided	Accept	Undecided

Table 9. Every candidate of the loop that improves. Number of earlier candidates kept on its history, sealed gain over the policy it would replace (PDMS) and the decision under each judge.

an error rate that holds wherever the reading stops. Table 8 puts the rules on one unit. Every rule reads whole logs in a random order, the gain of a pair is the mean difference per log, and each pair is read in 500 orders. By the per-log mean, 12 of the 76 pairs are gains, 59 are losses and 5 lie near zero. A bootstrap over 50 logs read once accepts 10.8 of the 12 gains on average with 22% of the frames. The sequential test accepts 11.2 with 53%, 2.4 times as many frames, and neither rule accepts a loss. The sequential test differs in that it may read on. At 50 logs the confidence sequence has accepted 1.1 gains and continues. The bootstrap read after every 10 logs until its interval leaves zero accepts 1.16 pairs near zero per pass, one of them in 88% of the orders. We also built pools without a gain by swapping the two scores in a random half of the logs and keeping the 18,972 pools whose mean gain is not positive. The bootstrap read after every 10 logs accepts 5.9% of them and up to 11.4% for one pair, against a nominal 2.5%. Read once over 50 logs it accepts 0.8%, and the sequential test accepts none. With a budget fixed in advance, a bootstrap over logs is the cheaper rule. The sequential test is the rule whose error rate holds when the amount of evaluation is decided while reading.

C.5. A loop built to improve

Here the start policy is DiffusionDrive trained on 5,000 frames, and each generation imitates the logged expert on 20,000 new frames, so real gains are expected. Each of the 3 seeds runs 5 generations under three deciding judges: the learned critic with the sequential test, the true score on the loop's own frames with one paired t test, and the sealed judge with the sequential test (Tab. 9). Of the 32 candidates, 11 are a first step from the start policy, reached over the three seeds and different roll-back histories. The sealed test accepted all of them; the learned critic accepted one and left the start policy in place in two of three seeds. The other 21 build on an accepted policy; the sealed test left all of them undecided, and the paired test on the loop's own frames accepted 12 of them (23 of all 32). The sealed test read 85% of the pool on average. The gains transfer in this loop, so the loss under the critic comes from its error, and the sealed test recovers 0.036 of the 0.051 PDMS that the paired test on true scores reaches.

D. Real-world field experiment

D.1. The two versions before the drives

The two versions in simulation. We scored both versions of ReCogDrive [42] on the NAVSIM navtest split with one pipeline. On the 12,118 scenes scored for both, the imitation version (IL) reaches 0.861 PDMS and the reinforced version (RL) 0.904, a paired difference of +0.043 (95% interval +0.039 to +0.048). Table 10 splits the difference by the terms of PDMS. RL gains by staying in the drivable area (0.946 to 0.976; 440 scenes penalized only for IL against 72 only for RL) and by progress (0.804 to 0.858). At-fault collisions change little (101 against 88 scenes), and driving-direction compliance, which PDMS leaves out, falls from 0.981 to 0.967. The routes with a curve therefore test drivable-area compliance, the term that contributes most to the simulator's gain.

Plans on the vehicle's camera. We ran both versions on 893 camera windows recorded on the vehicle under a human driver. Their planned trajectories differ by 0.68 m on average over the 4 s horizon, and by more than 0.5 m in 85% of the windows. Repeated sampling of one version moves its plan by 0.11 m (IL) and 0.12 m (RL). The two versions thus differ on real camera input by about six times their

Term	IL	RL	RL − IL	Scenes penalized only for IL / for RL	PDMS change, this term alone
No at-fault collision (NC)	0.980	0.981	+0.001	101 / 88	−0.004
Drivable-area compliance (DAC)	0.946	0.976	+0.030	440 / 72	+0.015
Time to collision within bound (TTC)	0.941	0.951	+0.010	337 / 216	+0.002
Ego progress (EP)	0.804	0.858	+0.054		+0.006
Comfort (C)	1.000	1.000	+0.000	5 / 0	+0.000
PDMS	0.861	0.904	+0.043		
Driving-direction compliance (DDC)*	0.981	0.967	−0.014	31 / 221	

Table 10. PDMS terms of the two versions. Means over the 12,118 navtest scenes scored for both. Penalized: the term is below one. Last column: change in mean PDMS when only this term takes the RL values. *Not part of PDMS.

Route	Pair	Order	IL	RL	RL − IL (route fraction)
Left	1	IL first	43 m, brake	55 m, brake	+0.082
	2	RL first	30 m, brake	57 m, brake	+0.186
	3	RL first	39 m, brake	49 m, brake	+0.069
	4	IL first	45 m, brake	47 m, brake	+0.008
	5	RL first	45 m, brake	48 m, brake	+0.018
	6	IL first	44 m, brake	54 m, steer	+0.069
Straight	1	IL first	completed	completed	+0.000
	2	RL first	completed	completed	+0.000
	3	IL first	completed	completed	+0.000
Right	1	RL first	83 m, brake	98 m, steer	+0.106

Table 11. Every scored pair of the field experiment. IL and RL: distance along the route at the first intervention and its kind (brake or steering override), or completed.

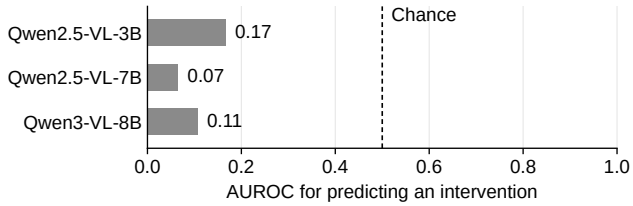


Figure 19. Vision-language judges on drive videos. AUROC for predicting which of the 20 scored drives needed an intervention.

sampling spread.

D.2. Vehicle and procedure

Full-scale vehicle and software. The vehicle is a full-scale electric van with a drive-by-wire kit, an RTK-corrected inertial navigation system, three wide-angle cameras and an on-board NVIDIA RTX A6000 (Fig. 18). The two versions share the vision-language backbone and differ only in the weights of the diffusion planner. Each plans an eight-point trajectory over 4 s from the front camera with five diffusion steps, once per 0.80 s. The camera calibration, the trajectory tracker and the speed limit of 4 m/s are the same for both.

Routes and scoring. The main route (Fig. 18c) is 144.4 m long: about 25 m straight, a left curve of about 55° between about 25 and 95 m, and about 50 m straight. A straight route (60.0 m) and a route with a right curve (143.1 m) lie on the same road. Every drive starts with the rear axle within 3 m of the route start, in daylight on a dry road, with a safety driver who keeps a foot over the brake. A drive ends at the first intervention (a brake, steering or throttle override, a timeout or a stall) or at the end of the route. Its score is $\min(D, L)/L$, where D is the distance along the route at the first intervention and L the route length.

Order and blinding. The two versions are driven in pairs from the same start. Pairs come in blocks of two, and the order within each block is random. The safety driver and the operator saw the versions only under two blind labels. After every drive the safety driver guessed the version. The guesses were right in 2 of 12 drives on the main route, 4 of 4 on the straight route (2 answers of "don't know") and 2 of 2 on the right curve.

D.3. Results

Results by pair. Table 11 lists every scored pair: 6 on the main route, 3 on the straight route and 1 on the right curve. No drive completed the main route (0 of 12). The safety driver stopped every drive between 30 and 57 m, in the first half of the curve, after 41.2 m on average for IL and 51.6 m for RL. RL went further in 6 of 6 pairs (exact sign test, one-sided $p = 0.016$), by $+0.072$ of the route on average, and the confidence sequence after 6 pairs is $[-0.50, +0.57]$, so the sequential test is undecided. Every drive on the straight route completed it (6 of 6). On the right curve IL was stopped at 83.1 m and RL at 98.2 m. Figure 20 shows the front camera when the safety driver took over. The videos on the project page show both drives of every scored pair with the same overlays. On both curves and in both versions, the planned trajectory leaves the route toward the inner curb.

Vision-language judges on drive videos. Qwen2.5-VL-3B, Qwen2.5-VL-7B and Qwen3-VL-8B each read 12 front-camera frames of every scored drive, evenly spaced from the start. All clips of a route have one length and



Figure 20. Front camera at the intervention. Last planning frame before the safety driver took over, for every pair on the two curves, with the planned trajectory over 4 s (blue: IL, orange: RL) and the route (white) drawn on the road. The last two frames show the first pair on the straight route in mid-drive.

end 1 s before the earliest intervention on that route (9.7 s on the main route), so no clip shows an intervention. The judges saw no label and gave the probability that the drive would need an intervention. Against the records of the 20 scored drives their AUROC is 0.17, 0.07 and 0.11 (Fig. 19): all three models rated the straight drives, which were all completed, as more likely to need an intervention than most curve drives. On the main route no judge separates the versions: the mean probabilities they gave IL and RL differ by at most 0.02.

E. Survey of self-improvement loops

Surveyed loops. Table 1 shows a selection of the 56 embodied self-improvement loops we surveyed, 24 in robot manipulation (Tab. 12) and 32 in driving (Tab. 13). The survey takes the term broadly: it covers loops with several rounds, single post-training steps and refinement at inference. For each loop we recorded the judge that decides inside it, the number of rounds it reports, whether it gives a result for every generation, and whether it measures a case of the policy exploiting its judge. Of these loops, 2 report a complete per-generation curve and 9 a partial one, 11 quantify the error of their judge, and 0 report both a curve and a measured case of exploitation. Table 14 adds the two loops outside embodied AI that report a loss after the peak.

F. Additional analyses

F.1. Anticipating the judge gap

Early signs of the judge gap. In 5 of 9 classifier-decided manipulation runs, the 95% interval of the judge gap lies above zero in at least one of the first five generations. We asked whether quantities that a loop can log without a sealed judge predict this one generation ahead. For generations two to five of every run, the signals of the preceding generation are paired with the event that the judge gap reaches 0.10 in success rate on manipulation (13 of 72 pairs) or 0.02 PDMS on driving (17 of 36). The signals are statistics of the policy's actions, error rates of the judges and, on driving, statistics of the critic's picks. Figure 21a,b give the AUC of each. On manipulation 20 of 48 signals reach 0.6. The best is the summed commanded motion along the vertical axis, with 0.81 [0.69, 0.91] and 0.80 within tasks. Not every signal survives this control: the number of gripper toggles falls from 0.68 to 0.31 within tasks, so it separates tasks, not generations. On driving only 2 of 13 signals reach 0.6, and the best, the frozen critic's rank correlation with the true score, reaches 0.67. Changes in the policy's motion thus carry some warning on manipulation, and none of the 13

Loop	Judge inside the loop	Rounds reported	Per-gen. curve	Judge exploitation
SimpleVLA-RL [38]	simulator environment; real-robot physical task completion	n.s.	no	no
π_{RL} [8]	simulator	n.s.	no	no
ConRFT [12]	binary classifier + human interventions	n.s.	no	mentioned
RLDG [80]	HIL-SERL classifier	n.s.	no	no
$\pi^*_{0.6}$ (RECAP) [1]	human raters	2	partial	no
RL Token [81]	human; per episode	n.s.	no	no
VLAC [92]	VLM process reward + done signal	n.s.	no	no
Robo-Dopamine [68]	reward-model threshold	n.s.	no	described
HIL-SERL [52]	classifier (200 positive / 1000 negative) + human	n.s.	no	no
SiLRI [94]	unassisted completion	n.s.	no	no
RoboCat [4]	reward model trained on crowd-sourced labels	n.s.	no	judge error only
Q-Planning [24]	human; per episode	5	partial	described
SARM2/SPIRAL [11]	stage reward model; final evaluation by humans	3	partial	mentioned
Human-Video Dynamics [7]	human-labeled outcomes	n.s.	no	no
ENPIRE [78]	sensor-fusion synthetic criterion	n.s.	partial	no
DreamGen [32]	10 real-robot trials per checkpoint	n.s.	no	judge error only
WMPO [96]	30 real-robot trials	n.s.	partial	judge error only
VLAW [28]	fine-tuned Qwen3-VL; threshold 0.8	2	partial	judge error only
RISE [86]	milestone scoring; 20 trials	n.s.	no	no
WorldSample [82]	goal condition	n.s.	no	no
E2HiL [16]	human + classifier	n.s.	no	no
GenSim2 [29]	solver success rate	n.s.	no	no
FATE [76]	RoboBrain 2.0 + simulated execution	n.s.	no	judge error only
RoboReward [37]	human vs VLM comparison	n.s.	no	judge error only

Table 12. Surveyed loops in robot manipulation. Judge exploitation: whether the paper mentions, describes or measures the policy exploiting its judge; "judge error only" marks papers that quantify the judge's error without such a case. n.s.: the paper states no number of rounds.

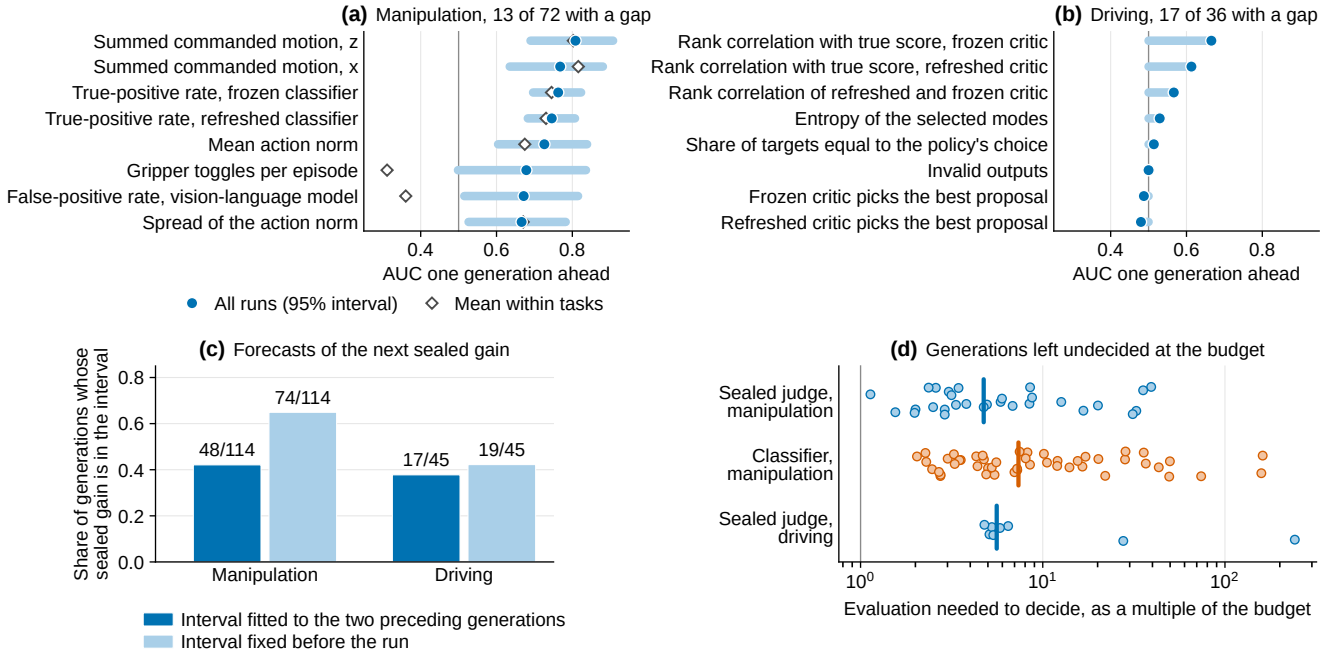


Figure 21. Warning signals, forecasts and undecided generations. (a, b) AUC with which a signal of the preceding generation predicts that the judge gap of a generation reaches the threshold; the eight best signals of each testbed. (c) Share of generations whose sealed gain lies in the interval forecast for it. (d) Evaluation that the test would need to decide each generation it left undecided, as a multiple of the budget it had; bars mark medians.

Loop	Judge inside the loop	Rounds reported	Per-gen. curve	Judge exploitation
GIGAFLOW [14]	simulator rules + WOSAC	1; about 10^{12} transitions	no	judge error only
Cornelisse self-play [13]	simulator rules	1; 2×10^9 steps	no	no
SPACeR [6]	reference-model likelihood + rules	1	no	described
Scaling Self-Play [59]	privileged teacher + rules	DAgger; 1.5×10^8 steps	no	no
TerraTransfer [79]	self-play teacher	1	no	no
RAD [18]	3DGS rule-based reward	1	no	no
RAD-2 [19]	learned discriminator	1	no	no
CaRL [31]	route-completion reward	1; 3×10^8 / 5×10^8 samples	no	described
PerlAD [21]	pseudo-simulation rules + world model	1	no	judge error only
MindDrive [17]	sparse rule-based reward	2 online epochs	no	no
ReCogDrive [42]	NAVSIM simulator PDMS	1	no	no
DiffusionDriveV2 [98]	trajectory-level rule-based reward	1	no	no
DriveDPO [64]	rule-based PDMS preferences	1 DPO stage	no	no
SMART-R1 [55]	WOSAC meta-metric	one SFT-RFT-SFT cycle; per stage	partial	no
ActiveAD [51]	displacement / soft collision / uncertainty	2; per-round readings	yes	no
Mcity Data Engine [3]	open-vocabulary ensemble	1	no	judge error only
R2SE [49]	PDMS + loss + entropy	1	no	no
AutoScale [60]	closed-loop evaluation feedback	T=5; final value only	no	no
R^{2}LPL [25]	rule-based detectors	5 (plus a 10-round envelope)	yes	no
ELF-VLA [53]	Qwen3-VL-32B diagnosis	k=1	no	mentioned
SceneCrafter [22]	3DGS simulator	1	no	no
SimScale [70]	pseudo-expert + rules	1	no	no
Cosmos-Drive-Dreams [57]	none (supervised)	1	no	no
World Engine [39]	rules in reconstructed environments	1; multiple rounds unstable	no	no
AD-R1 [83]	impartial world-model critic	1	no	judge error only
ReSim [85]	Video2Reward	1	no	judge error only
EvaDrive [33]	learned multi-objective critic	$\geq$4; best at 2 to 3	partial	no
CriticVLA [87]	critic trained on rule-based labels	multi-step refinement; diminishing	partial	no
DriveReward [10]	1B VLM reward model	1	no	no
Learning to Drive in a Day [34]	safety-driver takeover	11 to 35 episodes	no	no
H-DSAC [44]	human interventions	10^5 steps	no	no
Waymo (blog post, Dec. 2025)	simulator + critic	n.s.	no	no

Table 13. Surveyed loops in driving. Columns as in Tab. 12.

Loop	Judge	Reported effect
Self-Regress [47] (competitive programming)	CodeGrader	collapse of 17 points after the peak
Verifiers Go Bad [46] (multimodal QA)	frozen VLM verifier	-3.4 to -10.9 points across tasks

Table 14. Loops outside embodied AI. Both report a loss from further rounds of self-improvement.

	Learning		Success		
Training data	rate	n	Start	Continued	Change
Own successes	5×10^{-4}	60	0.633	0.383	-0.25 [-0.44, -0.06]
Own successes	1×10^{-4}	60	0.633	0.500	-0.13 [-0.30, $+0.03$]
Own successes	3×10^{-5}	60	0.633	0.333	-0.30 [-0.44, -0.16]
Own successes	1×10^{-4}	120	0.675	0.408	-0.27 [-0.38, -0.16]
Demonstrations	1×10^{-4}	120	0.675	0.983	$+0.31$ [$+0.23$, $+0.39$]
Both, 1:1	1×10^{-4}	120	0.675	0.733	$+0.06$ [-0.06, $+0.17$]

Table 15. Training data of the improver. A start policy of task 1 trained for 2,000 more steps on each kind of data. Demonstrations: those used to adapt the start policy. Change: paired change in success rate with 95% interval.

signals we logged does on driving.

Forecasting the sealed gain. Before each sealed reading we fitted an interval for the sealed gain to the two preceding generations of the run. The measured gain fell in it in 48/114 manipulation and 17/45 driving generations (Fig. 21c). An interval fixed before the run and never updated contains 74/114 and 19/45. In these runs the interval fitted to the two preceding generations does no better than the fixed one.

F.2. The sealed judge and its budget

A vision-language model as the sealed judge. The sealed judge of the manipulation runs is the simulator's success predicate, which a robot outside simulation does not have. We tested Qwen2.5-VL-7B as its stand-in, with one threshold fitted on 510 episodes. On 300 further episodes it reaches a true-positive rate of 0.82 at a false-positive rate of 0.24 against the predicate (AUC 0.85), so sensitivity plus specificity minus one is 0.585. The value differs widely between tasks (0.47, 0.36 and 0.94). A sealed judge this noisy would add its own error to every acceptance. The perturbed pool of Fig. 13 is a second sealed judge that keeps the predicate. Of the four pairs of policies of one task compared on

Deciding			Refreshed		Frozen		Sealed		FPR	
judge	Task	Gen.	Decision	Pairs	Decision	Pairs	gain	Failures	Refreshed	Frozen
Classifier	2	4	Undecided	300	Reject	161	-0.153	89	0.46	0.10
	2	5	Undecided	300	Reject	13	-0.350	97	0.53	0.07
	3	4	Undecided	300	Reject	54	-0.136	36	1.00	0.00
	3	6	Undecided	300	Reject	35	-0.226	68	1.00	0.00
Sealed judge	2	4	Accept	188	Undecided	300	$+0.109$	19	0.47	0.21
	3	4	Undecided	300	Reject	270	-0.035	42	0.00	0.00
	3	5	Undecided	300	Accept	162	$+0.082$	35	0.00	0.03

Table 16. Generations that the refreshed and the frozen classifier decide differently. The classifier's test replayed on the same pairs with the verdicts of the classifier refreshed in the loop and of the classifier frozen before it. Failures: failed episodes of the candidate among the 300 pairs; FPR: share of them each classifier labels successful.

both pools, the test decides one on the perturbed pool, and it orders that pair as the sealed pool does (Kendall's τ 1.00; 0.50 over the point estimates of all four). The perturbed pool thus agrees with the sealed pool on the one pair it can decide.

Evaluation needed by undecided generations. A generation that is undecided at its budget is rolled back. For each one we projected the number of pairs at which the test would decide if the observed trend of the paired differences continued (Fig. 21d). On manipulation this covers 77 undecided tests, which would need 133 to 48,024 further pairs. The median is 4.7 times the budget for the 29 tests fed by the sealed judge and 7.4 times for the 48 fed by the classifier. On driving, 8 generations of the sealed-decided main and control runs are undecided per log with the whole pool read; they would need a median of 5.6 times the logs of the pool and up to 242 times. The undecided candidates are those whose gain is close to zero, and a moderately larger budget would not have decided them.

F.3. Components of the manipulation loop

Training data of the improver. Before the first generation we continued one start policy on three kinds of data and measured it on calibration conditions (Tab. 15). Trained on its own successful episodes, the policy loses at each of three learning rates (-0.30 to -0.13 on 60 conditions, -0.27 on 120). Trained on the original demonstrations it gains $+0.31$, and on an equal mixture of both $+0.06$ [-0.06, $+0.17$]. Demonstrations replayed through the same path as the collected episodes give $+0.25$ [$+0.15$, $+0.35$] on 120 conditions, so the loss comes from the collected episodes and not from the path they take. A loop that replays demonstrations would therefore improve through the demonstrations under either judge, and its gains could not be attributed to the collected data or to the judge that selects them. The improver of all runs trains on the collected episodes alone, at the learning rate with the smallest loss. The loops agree with this: the sealed test kept 12 of the 45 candidates trained this way (mean certified gain $+0.147$) and rolled back the rest.

Effect of refreshing the classifier. We replayed the classifier's test in all 114 generations with the verdicts of the classifier frozen before the loop. The decision differs in 7 (Tab. 16). In 5 of them the refreshed classifier labels more of the candidate's failures as successes than the frozen one, in two generations all of them. In the 4 generations that the classifier decided, the frozen classifier rejects a candidate that the refreshed one leaves undecided, so the run rolls back either way and no kept policy changes. On the 300 collected episodes of a generation the two classifiers disagree on at most one episode in 98/114 generations and on no more than 12/300 in any. On the 300 evaluation episodes of the candidate they disagree on at most one in 69/114 generations and on up to 68/300. The refresh thus changes little on the episodes the classifier is trained on and, in some generations, much on the failures of the next candidate.

Dependence on the start state. A condition is a start state with an environment seed. The seed changes the trajectory but rarely the outcome: 87.6% of 102,600 recorded episodes are distinct action sequences, yet in 93.1% of 17,100 groups of one run and one start state all six seeds end alike. In the sealed records, 86.2% of 4,384 groups of one generation and one start state have the same paired outcome on every seed drawn. The pairs of a test are therefore correlated within a start state; Appendix A measures the error rate under this sampling, and Fig. 14 tests a certified gain on new start states.

G. Limitations

Our loops run in simulation, on three short-horizon manipulation tasks and one driving benchmark, each with one published improver. The driving benchmark replays recorded logs, so other road users do not react to the policy. The field experiment tests one published improvement step on one vehicle, on routes without obstacles. Longer tasks, on-policy reinforcement learning and loops that run on hardware over several generations are left to future work.