GATED LEARNED-DYNAMICS PLANNING FOR EFFICIENT UAV OBSTACLE AVOIDANCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Learned dynamics models let an agent anticipate candidate actions, but invoking model-based planning at every control step can waste computation when reactive behavior is sufficient. We study when short-horizon online imagination is necessary for UAV obstacle avoidance in a controlled cluttered 3D corridor. A compact learned kinematics model supports receding-horizon target-grid planning; a fixed-clearance hybrid invokes this planner only near hazards and otherwise uses behavioral cloning. Across 60 seeded evaluation maps, reactive control and behavioral cloning succeed on 7% and 32% of trials, while always-on learned planning and the gated hybrid succeed on 85% and 87%, respectively. The hybrid reduces mean planning time from approximately 12.0 ms to 6.4 ms, with overlapping success intervals. On the 40 maps withheld from gate selection, success is 33/40 and 34/40. Call-level measurements attribute the time reduction to issuing about half as many of the same rollouts; a lateral boundary override inside the gate is rarely active. Development-holdout attempts to distill planned actions into a reactive policy do not recover online planning. An additional PPO training attempt is inconclusive because its selected checkpoint is the behavioral-cloning warm start, before policy-gradient updates. These results provide a controlled study of selective model-based computation, while exposing the limitation that runtime clearance and imagined-pose costs use privileged geometric information.

1 INTRODUCTION

Model-based agents can improve decision making by imagining the consequences of candidate actions before execution. Yet performing this computation at every control step may be unnecessary: in many states, a simple reactive policy already produces adequate behavior. This raises a practical question: *when is online imagination actually necessary?*

We study this question in cluttered UAV navigation, where locally safe actions can create delayed collisions with towers, hanging beams, or a ceiling. Our setting is intentionally controlled: a seeded 3D corridor, a compact kinematic simulator, and a fixed evaluation contract make it possible to compare reactive control, imitation, and learned-dynamics planning without conflating sensor changes or map changes. The learned forward model predicts six-dimensional kinematics, while a short target-grid rollout scores progress and clearance. The resulting planner is useful but costs computation even in open space.

Our method is a simple hybrid. A behavioral-cloning policy acts when current clearance is above a fixed threshold; otherwise, a learned-dynamics planner is invoked. The threshold is fixed at 3.5 m from a development holdout and is not learned. This design tests selective model-based computation directly: does a controller need to imagine continuously, or mainly near hazards? We also examine attempts to distill planned actions into a reactive network and to train a model-free policy.

Our contributions are: (i) a controlled, reproducible comparison of reactive, imitation, and learned-dynamics planning under a common seeded-map protocol; (ii) a fixed-clearance gated hybrid that retains comparable success to always-on planning while approximately halving mean planning time; and (iii) an analysis of attempted distillation, gate trade-offs, and clearance noise, together with an explicit accounting of privileged geometry. The main Track-A claim is limited to this simulator and protocol; a separate vision track is not ranked against it.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

2 Related Work

Behavioral cloning maps observations to expert actions but can suffer from state-distribution shift (Pomerleau, 1989); DAgger addresses this by collecting labels on learner states (Ross et al., 2011). Learned-dynamics model-predictive control instead uses a forward model to score candidate action sequences, as in shooting-based model-based RL (Chua et al., 2018) and sampling-based predictive control (Williams et al., 2017). World Models and Dreamer learn latent predictive representations and optimize policies in imagination (Ha & Schmidhuber, 2018; Hafner et al., 2020; 2025); our Track-A model is narrower, predicting explicit kinematics rather than pixels or a recurrent latent state.

For aerial navigation, learned reactive controllers can map vision directly to steering commands (Loquercio et al., 2018), while high-performance drone racing combines learned perception and control in a dynamics-rich setting (Kaufmann et al., 2023). Flightmare and AvoidBench provide substantially higher-fidelity simulation and benchmarking than the corridor used here (Song et al., 2021; Yu et al., 2023). We therefore use the corridor as a controlled ablation environment, not as evidence of deployment-level flight performance. Selective execution connects to event-triggered control and event-triggered MPC (Heemels et al., 2012; Li & Shi, 2014), as well as computation-aware planning for airborne systems (Yel et al., 2020). Unlike those works, our trigger is neither stability-derived nor adaptively scheduled: it is a fixed geometric rule used to isolate when a learned-dynamics search is invoked.

3 Problem Setup

The simulator is a 70 m corridor of width 18 m and height 6.2 m containing seeded towers, boxes, and hanging beams. A spherical vehicle starts at $z = 4$ m and must reach $z = 65$ m without collision or boundary violation. Each integer seed defines one Bernoulli trial. The main protocol uses disjoint blocks 101–120 (development holdout), 201–220, and 301–320, pooled as $n = 60$; success intervals are Wilson 95% intervals (Wilson, 1927). The default imagination horizon is $H = 20$ and replanning occurs every four control steps.

The state is $s = (x, y, z, v_x, v_y, v_z) \in \mathbb{R}^6$ and the action is $a = (a_x, a_y, \tau) \in \mathbb{R}^3$. Track A observes state and local features of the five nearest obstacles. A signed clearance probe $c(s)$ gives distance from the vehicle sphere to the nearest geometry. This probe is privileged at runtime in Track A and is therefore a limitation, not an assumption of deployment.

4 Method

4.1 Learned kinematics and behavioral cloning

The learned dynamics model is a residual MLP f trained on logged transitions at the imagination step $\Delta t_{\text{img}} = 0.07$ s:

$$\hat{s}_{t+1} = s_t + f(s_t, a_t, \Delta t_{\text{img}}). \tag{1}$$

It predicts kinematics only; it does not predict collisions. A behavioral-cloning MLP $\pi_{\text{BC}}(o_t)$ is trained by regression on expert actions and is frozen before evaluation.

4.2 Receding-horizon learned-dynamics planning

For each target $(x^\star, y^\star)$ in a fixed 7×4 lateral–vertical grid, the planner rolls f forward for H steps using a clipped proportional controller. It selects the minimum-cost target under a cost combining progress, nominal altitude, lateral deviation, predicted clearance, boundary violations, and collision penalties. Crucially, the predicted poses are scored using the true obstacle geometry in this simulator. The planner commits the first action and caches it for four control steps. Always-on planning uses this procedure at every replan point.

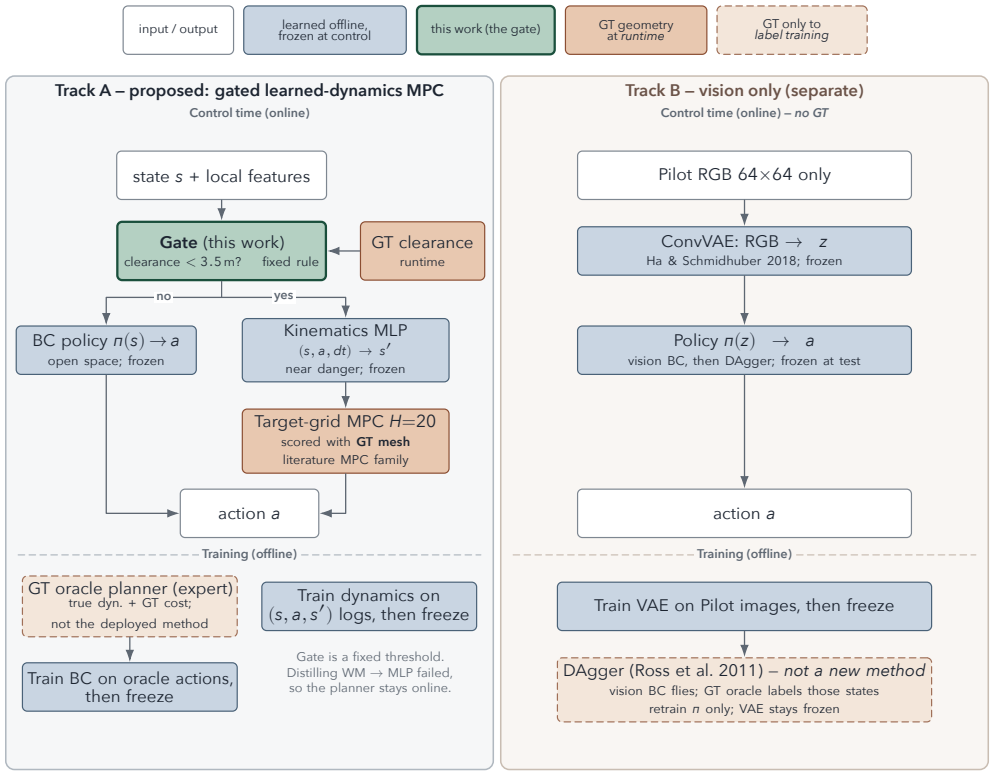

Figure 1: Method overview adapted from the dissertation implementation. Track A uses BC in open space and learned-dynamics planning near a fixed clearance threshold. Ground-truth geometry is used at runtime for Track-A clearance and imagined-pose scoring; this is a deployment limitation.

Table 1: Track-A success on the pooled 60-map protocol. Intervals are Wilson 95%. Seeds 101–120 selected the gate; 201–220 and 301–320 are disjoint replication blocks.

Controller	Success	Mean plan time
Reactive	7% [3,16]	0 ms
BC of oracle	32% [21,44]	0.2 ms
Always-on learned planning	85% [74,92]	12.0 ms
Gated learned planning ($g = 3.5\,\mathrm{m}$)	**87% [76,93]**	**6.4 ms**

4.3 Fixed-clearance gate

The hybrid controller is

$$a_t = \begin{cases} \mathrm{wm_rollout}(s_t), & c(s_t) < g \text{ or } |x_t| > 6.8, \\ \pi_{\mathrm{BC}}(o_t), & \text{otherwise}, \end{cases} \qquad g = 3.5\,\mathrm{m}. \tag{2}$$

The gate is a fixed rule selected on seeds 101–120 and frozen before the two replication blocks. It is not learned, adapted online, or optimized on the pooled test set. Figure 1 summarizes the computation path.

5 Experiments

5.1 Main Track-A comparison

The main result is a computation–success trade-off. Always-on planning succeeds on 51/60 maps, compared with 4/60 for reactive control and 19/60 for BC. The gated controller succeeds on 52/60,

with overlapping Wilson intervals relative to always-on planning, while reducing mean plan time by roughly 47%. The development block favors the gate by 95% versus 85%; on the third block the ordering reverses (85% versus 95%). Overlapping marginal intervals are not an equivalence test, and the gate was selected on one of the three pooled blocks. We therefore interpret the pooled evidence as no observed loss of success under this protocol. Section 5.2 isolates the maps that did not choose the gate and compares the two planners on the same seeds.

5.2 Replication maps and paired outcomes

Seeds 101–120 were used to choose $g = 3.5$ m, so Table 1 mixes that choice with later blocks. On seeds 201–320 ($n = 40$), gated planning succeeds on 33/40 (82.5%, Wilson 95% [68, 91]) and always-on planning on 34/40 (85%, [71, 93]). Locked mean plan times on these maps are 6.5 ms and 12.0 ms.

Pairing the 40 seeds gives 29 maps on which both succeed, 4 on which only the gate succeeds, 5 on which only always-on planning succeeds, and 2 on which both fail. The success difference is -2.5 percentage points. An exact two-sided McNemar test on the nine discordant maps gives $p = 1$: with four and five discordant outcomes, the observed split is among the most probable under a null of no systematic difference. A paired bootstrap 95% interval for the gated-minus-always-on difference, using 10,000 resamples of the seeds, is $[-17.5, +12.5]$ percentage points. The interval covers zero and also covers differences large enough that these 40 maps do not establish equivalence. The same post-hoc procedure on all 60 maps gives counts 46, 6, 5, and 3, a difference of $+1.7$ percentage points, McNemar $p = 1$, and a bootstrap interval of $[-8.3, +13.3]$ percentage points. That pooled difference is carried by the development block, which has two gated-only successes and no always-on-only success. Discordant seeds are listed in Appendix B. These tests were not preregistered.

5.3 Fewer planner calls, same call cost

The 6.4 ms and 12.0 ms entries average controller time over every control step, and cached steps contribute zero. A fresh run of the frozen checkpoints, with horizon 20, replanning every four steps, and the same 60 seeds, separates skipped calls from cheaper calls. Success for always-on planning and for the combined gate matched the locked episode logs on every seed. The timings in this subsection were measured on an Apple M4 with Python 3.10.12 and `time.perf_counter`. The locked means above remain the headline comparison.

On seeds 201–320, always-on planning executes the 28-target rollout on 25.2% of control steps and the combined gate executes it on 13.4%. Mean time per call is 47.2 ms and 47.0 ms. The 50th and 95th percentiles of call time are 47.6 ms and 50.2 ms for always-on planning, and 48.0 ms and 50.4 ms for the gate. Total planner time per episode falls from a mean of 3.0 s (median 3.2 s, 95th percentile 3.4 s) to 1.6 s (median 1.7 s, 95th percentile 2.1 s). Medians sit above means because collisions end some episodes early. Across control steps, latency has median 0 ms for always-on planning and 0.15 ms for the gate, while the 95th percentile is about 49 ms for both: a typical step is a cache hold or a BC evaluation, and the upper tail is one rollout. Step-mean times on this rerun are 11.9 ms and 6.4 ms on seeds 201–320, beside the locked replication means of 12.0 ms and 6.5 ms.

Figure 2 shows where the calls occur. The panels are the first mutual success and the first disagreement in seeds 201–320, read in seed order. In both corridors the gate uses behavioral cloning on the open stretches and the learned-dynamics planner beside towers and beams. Seed 205 also separates the two controllers: always-on planning collides, while the gate reaches the goal plane.

Table 2 splits the locked disjunction on the same replication seeds. Clearance alone reproduces the combined gate: 33/40 successes and planner calls on 13.3% of steps. The lateral override alone reproduces BC on every one of the 60 pooled seeds (11/40 on this subset; 19/60 overall) and, on seeds 201–320, never calls the planner. Of the 2,086 combined-gate calls on all 60 maps, 2,067 are triggered by clearance alone and 19 occur while $|x| > 6.8$. The override changes one development outcome, seed 113, where the combined gate succeeds and clearance alone fails. It does not change the replication success count. The measured saving is the clearance threshold. Definitions, the development block, and the rerun command are in Appendix F.

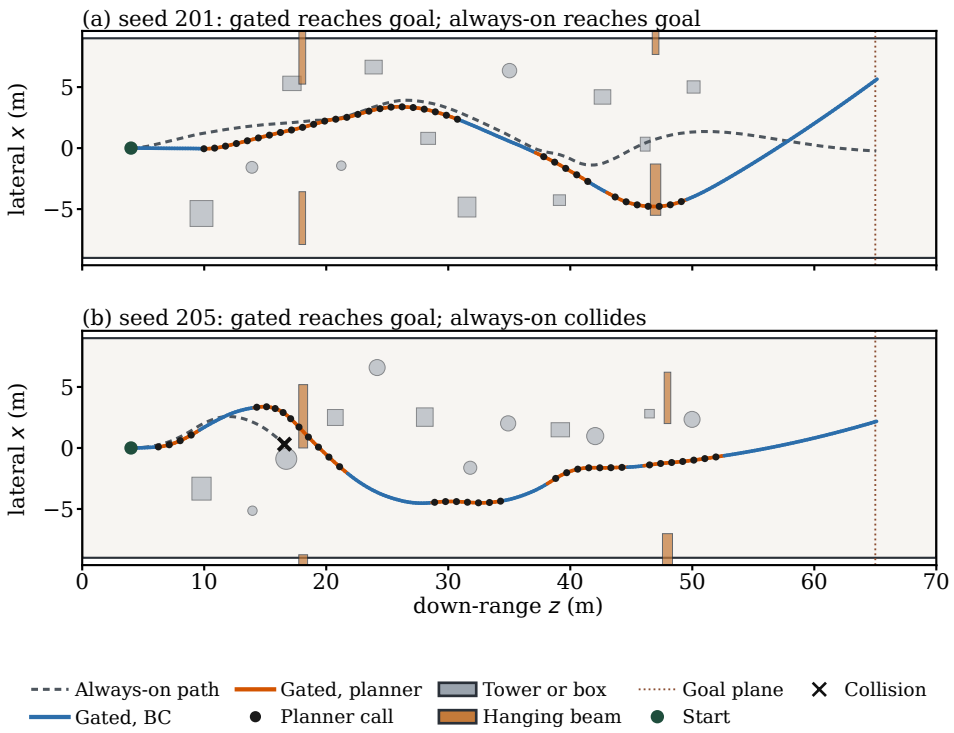

Figure 2: Plan view of two replication corridors. Seeds were fixed from the locked logs before any path was drawn: (a) seed 201 is the smallest seed in 201–320 on which both controllers succeed; (b) seed 205 is the smallest seed in that range on which exactly one succeeds. The grey dashed curve is always-on planning. On the gated path, blue is behavioral cloning and orange is the learned-dynamics planner, including cached repeats of a planner action. Black dots are steps that execute a new rollout. On both drawn trials every such call has clearance below 3.5 m. Grey shapes are towers and boxes; orange bars are hanging beams. The dotted line is the goal plane $z = 65$ m. These are two trials, not a summary of Tables 1 and 2.

Table 2: Gate-component rerun on replication seeds 201–320. A planner call is one 28-target rollout. Episode plan is the mean sum of those calls. Per-call times are means. Lateral-only success matches behavioral cloning on every seed.

Rule	Success	Call fraction	ms/call	Episode plan
Always-on	34/40	25.2%	47.2	3.0 s
Clearance only ($g = 3.5$ m)	33/40	13.3%	46.9	1.6 s
Lateral override only ($\lvert x \rvert > 6.8$)	11/40	0.0%	–	0.0 s
Clearance and lateral	33/40	13.4%	47.0	1.6 s

5.4 Planned computation is state selective

The sweep characterizes the frozen gate. Thresholds of 0, 1.5, 2.5, 3.5, 4.5, 6, and 99 m give pooled success of 32%, 53%, 68%, 87%, 82%, 80%, and 83%, with mean plan time of approximately 0.2, 2.2, 4.1, 6.7, 8.6, 10.0, and 10.5 ms. The 3.5 m rule has the highest point estimate in this sweep. The 6.7 ms sweep timing differs from the 6.4 ms locked main evaluation timing. The apparent 4.5 m success dip on the development block does not repeat in the later blocks. Full split values are in Appendix C.

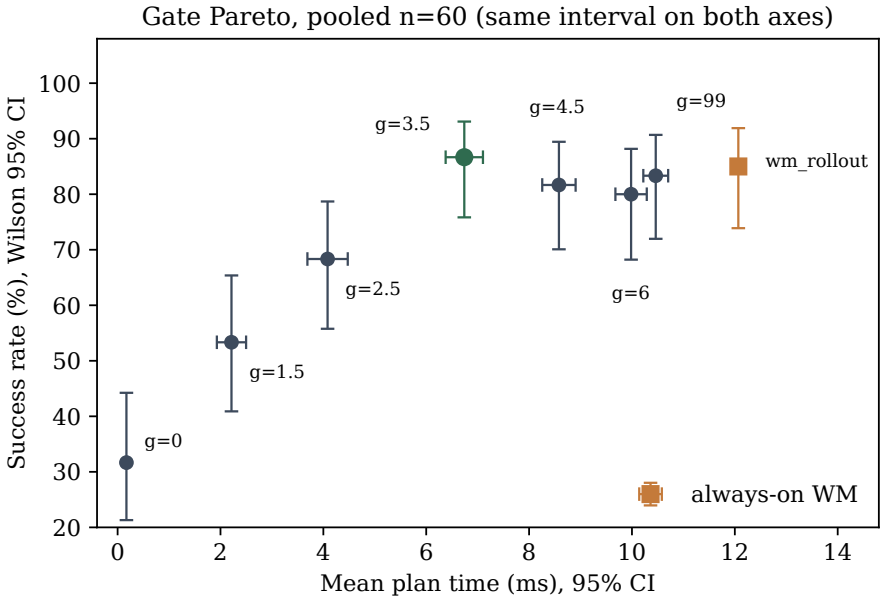

Figure 3: Pooled gate sweep ($n = 60$): success versus mean planning time. Vertical intervals are Wilson 95%; horizontal intervals are 95% intervals on mean planning time. The 3.5 m rule was selected on seeds 101–120 before the replication blocks.

5.5 Can a reactive policy replace online planning?

Action distillation of the learned planner into a reactive MLP did not improve over BC on the development holdout. A separate imagination-trained policy reached 25–35% and a residual BC variant did not exceed 40% there, against 85% for the always-on planner under its locked setting. These are development results rather than replicated findings, so they motivate retaining online search without establishing that every reactive architecture must fail.

We also trained a privileged-state PPO actor initialized from BC on training seeds 1–64, with validation on seeds 41–50. The selected `ppo_best` checkpoint is labeled `bc_warmup`, `step=0`; its 101–120 score is 6/20 (30%, Wilson 95% CI [15%, 52%]). The run reached 200,704 environment steps, but the selected checkpoint remained the pre-update actor. This is an inconclusive PPO attempt, rather than a 30% result for a policy improved by PPO. It is excluded from Table 1 and cannot support a general claim that model-free RL fails here.

5.6 Clearance-noise robustness

At $\sigma = 0$, the ablation recovers 52/60 gated and 51/60 always-on successes. At $\sigma = 0.25$ m, these become 45/60 and 50/60; at 0.5 m, both achieve 50/60; at 1 m, they achieve 51/60 and 48/60. At 2 m, the gated policy falls to 31/60 and always-on planning to 40/60. The gated controller's mean plan time also falls from 6.4 to 2.6 ms, consistent with the noisy gate invoking the planner less often. This additive-noise probe does not simulate structured perception or mapping errors. Exact intervals and timings appear in Appendix D.

6 Discussion and Limitations

The evidence supports a narrow claim: in this seeded corridor, short-horizon learned-dynamics planning is useful, but it need not run continuously to obtain similar observed success. The gate is deliberately simple and fixed, which makes its computation path interpretable. Its benefit is conditional on the BC policy being adequate in high-clearance states and on clearance identifying states in which search is useful. The non-monotone gate sweep shows that invoking the planner more often is not automatically better; switching changes the visited-state distribution as well as computation.

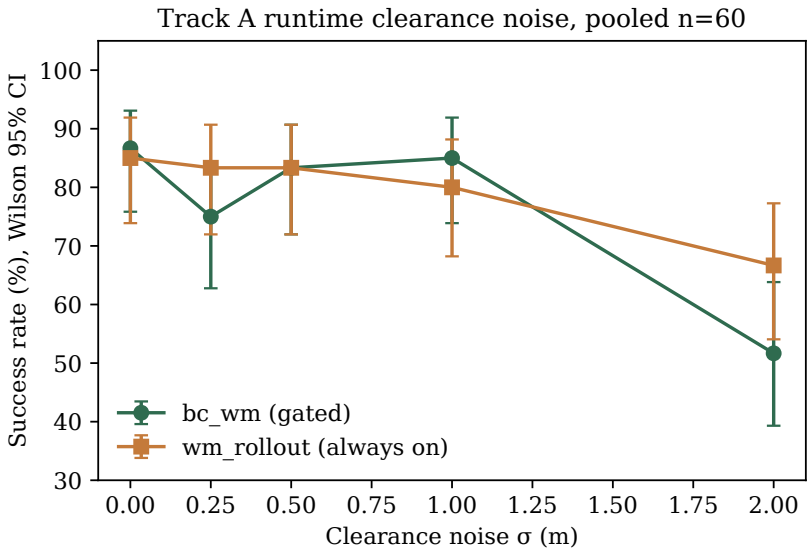

Figure 4: Clearance-noise ablation on the pooled 60-map protocol. Independent Gaussian noise perturbs runtime clearance used by the gate and imagined-pose cost; simulator collision outcomes remain based on true geometry. Whiskers are Wilson 95% intervals.

Several limitations constrain the conclusion. First, the runtime gate and imagined-pose cost both query privileged obstacle geometry. The additive-noise experiment perturbs that scalar query independently and is neither a learned perception pipeline nor a model of correlated mapping error. Second, the simulator uses kinematics rather than rigid-body flight dynamics, and a single corridor generator cannot measure transfer across layouts, vehicle dynamics, or sensor domains. Third, the 3.5 m gate was selected on seeds 101–120, which remain inside the pooled 60-map summary; the untouched comparison is 33/40 versus 34/40 on seeds 201–320. Fourth, that replication comparison still has a post-hoc bootstrap interval from -17.5 to $+12.5$ percentage points. The McNemar test and bootstrap were not preregistered and do not specify an equivalence margin. Finally, wall-clock measurements, including the call-level timings on an Apple M4, use Python `time.perf_counter` on one software and hardware setup and should be read as within-run comparisons.

The PPO attempt is inconclusive and protocol-separated, and the separate vision track is not a sensor-matched comparison with Track A. AvoidBench transfer results are excluded from the headline table because the command channel and simulator interface introduce additional failure modes. A stronger next study would freeze the gate on a separate tuning set, evaluate a measured-clearance input on unseen scene families, and test under vehicle dynamics that expose tracking error.

7 Conclusion

We asked when online imagination is necessary for UAV obstacle avoidance. In the present controlled setting, it is most valuable near hazards: a fixed 3.5 m clearance gate keeps observed success close to always-on learned-dynamics planning while approximately halving mean planning time. On the maps that did not select the threshold, success is 33/40 against 34/40, and the shorter time is the same rollout invoked about half as often. The lateral boundary term in the gate is rarely used. The tested distillation variants did not recover online-search performance on the development holdout. The next step is to replace privileged runtime geometry with a measured or mapped clearance signal and repeat the protocol under that sensor model.

AI USE STATEMENT

An AI assistant supported manuscript drafting and editing, numerical cross-checking, and execution of post-hoc analyses. All reported results were derived from the simulator or locked logs and verified by the authors, who remain responsible for the final manuscript.

REFERENCES

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems*, volume 31, pp. 4754–4765. Curran Associates, Inc., 2018.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems*, volume 31, pp. 2451–2463. Curran Associates, Inc., 2018.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640(8059):647–653, April 2025.

W. P. M. H. Heemels, Karl H. Johansson, and Paulo Tabuada. An introduction to event-triggered and self-triggered control. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, pp. 3270–3285, Maui, HI, USA, 2012. IEEE.

Elia Kaufmann, Leonard Bauersfeld, Antonio Loquercio, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. Champion-level drone racing using deep reinforcement learning. *Nature*, 620(7976):982–987, August 2023.

Huiping Li and Yang Shi. Event-triggered robust model predictive control of continuous-time nonlinear systems. *Automatica*, 50(5):1507–1513, 2014.

Antonio Loquercio, Ana I. Maqueda, Carlos R. del-Blanco, and Davide Scaramuzza. DroNet: Learning to fly by driving. *IEEE Robotics and Automation Letters*, 3(2):1088–1095, 2018.

Dean A. Pomerleau. ALVINN: An autonomous land vehicle in a neural network. In *Advances in Neural Information Processing Systems*, volume 1, pp. 305–313. Morgan Kaufmann, 1989. Conference held in 1988.

Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pp. 627–635. PMLR, 2011.

Yunlong Song, Selim Naji, Elia Kaufmann, Antonio Loquercio, and Davide Scaramuzza. Flightmare: A flexible quadrotor simulator. In *Proceedings of the 2020 Conference on Robot Learning*, volume 155 of *Proceedings of Machine Learning Research*, pp. 1147–1157. PMLR, 2021.

Grady Williams, Nolan Wagener, Brian Goldfain, Paul Drews, James M. Rehg, Byron Boots, and Evangelos A. Theodorou. Information theoretic MPC for model-based reinforcement learning. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1714–1721, Singapore, 2017. IEEE.

Edwin B. Wilson. Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association*, 22(158):209–212, 1927.

Esen Yel, Tony X. Lin, and Nicola Bezzo. Computation-aware adaptive planning and scheduling for safe unmanned airborne operations. *Journal of Intelligent & Robotic Systems*, 100(2):575–596, 2020.

Hang Yu, Guido C. H. E. de Croon, and Christophe De Wagter. AvoidBench: A high-fidelity vision-based obstacle avoidance benchmarking suite for multi-rotors. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9183–9189, London, United Kingdom, 2023. IEEE.

A Main results by evaluation block

Block	Reactive	BC	Always-on	Gated ($g = 3.5$ m)
101–120	3/20	8/20	17/20	19/20
201–220	1/20	5/20	15/20	16/20
301–320	0/20	6/20	19/20	17/20
Pooled	4/60	19/60	51/60	52/60

Seeds 101–120 were used to select the gate threshold; the other two blocks were run as disjoint replications. The pooled row is descriptive and includes the development block.

B Paired outcomes

The replication discordant seeds, gated success with always-on failure, are 205, 206, 211, and 214. The always-on-only seeds are 212, 216, 217, 310, and 315. Both fail on 219 and 303. The development block adds gated-only seeds 113 and 117, and one shared failure, seed 112. It contains no always-on-only seed. The bootstrap draws seeds with replacement, uses generator seed 20260926 for 10,000 replicates, and reports the 2.5 and 97.5 percentiles with linear interpolation. The exact McNemar probability is the two-sided binomial probability of discordant tables at least as extreme as the observed split under a fair coin.

C Gate sweep by evaluation block

Gate (m)	101–120	201–220	301–320	Pooled / plan ms
0	40%	25%	30%	32% / 0.2
1.5	60%	45%	55%	53% / 2.2
2.5	80%	70%	55%	68% / 4.1
3.5	95%	80%	85%	87% / 6.7
4.5	75%	80%	90%	82% / 8.6
6.0	80%	75%	85%	80% / 10.0
99	85%	70%	95%	83% / 10.5

The near-always-on gate at 99 m is still a hybrid controller and remains distinct from the always-on planner.

D Clearance-noise ablation

σ (m)	Gated success	Plan ms	Always-on success	Plan ms
0	87% [76,93]	6.4	85% [74,92]	12.0
0.25	75% [63,84]	6.3	83% [72,91]	11.9
0.5	83% [72,91]	5.6	83% [72,91]	11.6
1.0	85% [74,92]	4.3	80% [68,88]	10.2
2.0	52% [39,64]	2.6	67% [54,77]	6.9

Every cell uses the same 60 seeded maps; success intervals are Wilson 95%.

E Reproducibility details

The corridor is 18 m wide, 6.2 m high, and 70 m long. The simulated vehicle is a sphere of radius 0.42 m. Control runs at $1/30$ s and imagined transitions at 0.07 s. Track-A observations have 41 features, including state, ego distances, and five nearby obstacle descriptions. The learned model predicts a residual six-dimensional state transition. The planner uses 28 lateral–vertical set points, horizon 20, and a four-step replan period. The main evaluation uses fixed checkpoints and seeds 101–120, 201–220, and 301–320, with one trial per seed. The development block was used to choose $g = 3.5$ m. Mean plan time is the episode mean of controller-reported `time.perf_counter`

durations, with cached control steps contributing zero; it excludes simulator stepping and should be compared only within this implementation and machine setup.

The Track-A evaluation used the following command in our implementation:

```
python -m research.run_baselines \
  --controllers reactive bc_oracle bc_wm wm_rollout \
  --n-seeds 20 --difficulty easy --seed-start 101 \
  --horizon 20 --plan-every 4 \
  --out-dir research/results/repro_s101 \
  --bc-model research/checkpoints/bc_mlp_40pct.json \
  --bc-wm-model research/checkpoints/bc_wm_mlp.json \
  --dynamics-model research/checkpoints/dynamics_mlp.json
```

The blocks starting at seeds 201 and 301 used the same command with `--seed-start 201 --out-dir research/results/repro_s201` and `--seed-start 301 --out-dir research/results/repro_s301`. Distinct output directories were used because each invocation writes the same filenames. The three blocks were aggregated with:

```
python -m research.aggregate_eval --episodes \
  research/results/repro_s101/baseline_episodes.json \
  research/results/repro_s201/baseline_episodes.json \
  research/results/repro_s301/baseline_episodes.json \
  --out research/results/repro_pooled.json
```

The same checkpoint files were used for every block. SHA-256 hashes for the audited files are `44cc8861...fa824` (BC), `4ef293ea...98f677` (hybrid BC), and `35348ce0...62bb017` (dynamics). The separate vision analysis is outside the Track-A comparison.

F CALL-LEVEL TIMING AND GATE COMPONENTS

A planner call is a control step that executes the 28-target rollout. Cached repeats of a previous rollout, and BC steps, are not calls. The call fraction divides planner calls by control steps. Per-call time is the `time.perf_counter` duration of that rollout. Episode plan time sums those calls and leaves out BC forward-pass time. The step-mean in Table 1 is different: it averages every control step, so cached steps contribute zero and BC steps contribute a fraction of a millisecond. Step-latency percentiles use that per-step series. On the replication rerun, step-latency medians are 0 ms (always-on) and 0.15 ms (combined gate); 95th percentiles are 49.25 ms and 48.70 ms.

The component rerun uses the same checkpoints and seeds as the locked evaluation. On seeds 101–120 the success counts are 17/20 (always-on), 19/20 (combined), 18/20 (clearance only), and 8/20 (lateral only). Pooled over 60 maps they are 51/60, 52/60, 51/60, and 19/60. Pooled call fractions are 25.2%, 13.3%, 13.2%, and 0.1%. The only success disagreement between clearance alone and the combined gate is seed 113. Lateral-only successes match the locked BC successes on all 60 seeds. The rerun was an Apple M4, macOS arm64, CPython 3.10.12.

The paired analysis and the gate-component rerun used the following command in our implementation:

```
python -m research.analyze_submission_gaps
```

That run recomputes the paired outcomes from the locked episode logs and reruns always-on planning, the combined gate, clearance only, and the lateral override. It checks that the rerun success pattern for always-on planning and the combined gate matches those logs. Figure 2 was produced in the same implementation by `python paper/iclr2027/figures/make_gate_plan.py`, using the seed rule in the caption and the same success check against the locked logs.