AIRPLAYGROUND: BENCHMARKING AERIAL NAVIGATION ACROSS DIVERSE RECONSTRUCTED ENVIRONMENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large-scale aerial resources provide a foundation for learning navigation across diverse environments. However, existing resources remain limited in providing (1) densely sampled flight demonstrations for learning different policy formulations and (2) broad environment coverage for standardized closed-loop evaluation. To address these limitations, we introduce AirPlayground, a unified benchmark for training and evaluating aerial navigation policies across diverse environments. We first construct 94 photorealistic 3D Gaussian Splatting (3DGS) environments spanning indoor, urban outdoor, and vegetation settings, with aligned collision geometry. Building on these environments, we develop a scene-adaptive pipeline for automated expert flight collection, yielding 94K demonstrations and 46.5M synchronized observation–state–action samples at 50 Hz. We further provide an extensible API that supports training different policy formulations on this shared corpus and evaluating them under unified protocols in the corresponding environments. We benchmark seven baselines spanning four policy paradigms across three navigation settings in seen and unseen environments, and further examine scaling behavior with ACT. Scaling ACT's training corpus from 5 to 70 scenes raises median success rates on seen and unseen environments from 30.6% and 16.8% to 41.2% and 29.4%, respectively. Through this shared training and evaluation framework, AirPlayground supports analysis of how offline prediction errors relate to closed-loop outcomes. Our analyses highlight the roles of prediction errors, execution protocols, and scene interactions in closed-loop performance.



Figure 1: **AirPlayground unifies dense flight experience, cross-policy learning, and closed-loop evaluation in reconstructed real-world environments.** AirPlayground pairs executable 3DGS reconstructions with dense expert flight demonstrations collected in the same environments. A unified cross-policy API adapts the shared corpus to different policy formulations, enabling standardized closed-loop evaluation and analysis of how offline prediction errors relate to deployment outcomes.

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.

1 INTRODUCTION

Learning-based aerial navigation depends on large-scale, diverse experience to generalize across environments (Zheng et al., 2024; Liu et al., 2025). Acquiring experience through online interaction can be costly and difficult to scale. Offline demonstrations support scaling policy learning without environment interaction during training (O'Neill et al., 2024). Yet policies trained on demonstrations learn from expert states, whereas deployment exposes them to states shaped by their actions (Seo et al., 2024). Small errors can compound through feedback, so accurate imitation does not guarantee reliable navigation. Assessing reliability therefore requires systematic closed-loop evaluation across diverse environments, where policies must respond to the consequences of their actions.

To meet these learning and evaluation needs, aerial navigation research has produced both flight datasets and interactive environments. Large-scale datasets such as CityNav and OpenFly provide aerial trajectories for navigation and vision-language learning (Lee et al., 2025; Gao et al., 2026), substantially increasing the scale and diversity of reusable flight experience. In parallel, 3DGS-based embodied platforms make reconstructed real-world scenes available as photorealistic, geometrically grounded environments for interactive navigation (Liu et al., 2026b; Miao et al., 2026; Xia et al., 2026). Recent benchmarks further combine reconstructed environments, collision-aware interaction, and flight trajectories for policy learning and evaluation (Guo et al., 2026).

Despite this progress, two limitations remain in the resources for aerial policy learning and evaluation. **(i) Limited dense flight experience.** Densely sampled observation–action demonstrations remain limited across diverse real-world environments. Dense supervision captures continuous flight control for learning different action representations, while coverage across scenes exposes policies to varied visual conditions and spatial structures. **(ii) Limited standardized evaluation across diverse environments.** Standardized closed-loop evaluation remains limited in its coverage of environments with diverse appearances and spatial structures. Indoor spaces, urban outdoor areas, and vegetation present different visibility and obstacle configurations, making evaluation across these settings important for assessing how consistently policies perform beyond a narrow set of scenes.

To address these limitations, we introduce AirPlayground, a unified benchmark for aerial navigation that pairs dense flight experience with photorealistic 3DGS reconstructions and executable geometry. AirPlayground comprises **94 reconstructed scenes** spanning indoor, urban outdoor, and vegetation environments, with **nearly 100K flight demonstrations** and **46.5M RGB frames** under 50 Hz observation–action supervision. To scale collection, we develop a scene-adaptive pipeline, reusable across reconstructed scenes, that models navigable space, samples navigation tasks, and generates expert demonstrations. We further introduce an extensible cross-policy API that adapts the shared corpus to each policy's native representation and maps its outputs to a common control interface under unified evaluation protocols. By linking recorded trajectories to executable scene geometry, AirPlayground also enables analysis of how offline prediction errors relate to closed-loop outcomes.

Specifically, AirPlayground makes three main contributions.

(i) Dense flight experience paired with executable reconstructions of real-world environments. We pair dense flight demonstrations with photorealistic reconstructions and aligned navigation and collision geometry for policy learning and matched closed-loop execution.

(ii) A unified benchmark across navigation policy paradigms. Our extensible API supports shared-data policy training and unified closed-loop evaluation across three navigation settings, while preserving each policy's native representations and inference structure through adapters.

(iii) Cross-policy evaluation and deployment diagnosis. We benchmark seven baselines and analyze offline predictions and closed-loop behavior. These studies show that offline accuracy alone does not fully explain deployment performance and examine how execution horizons and scene geometry affect navigation, demonstrating the benchmark's value beyond performance comparisons.

2 RELATED WORK

Aerial Navigation Datasets. Prior work has expanded navigation-level supervision from constrained simulated settings toward larger urban environments, richer tasks, and broader scene coverage (Liu et al., 2023; Wang et al., 2025b; Lee et al., 2025; Liu et al., 2026c). More recent efforts

scale collection through automated planning and expert trajectory generation (Low et al., 2025; Gao et al., 2026; Adang et al., 2025). Other efforts collect fine-grained flight demonstrations for reactive control and long-horizon navigation (Wang et al., 2025a; Zheng et al., 2026). Despite these advances, temporally dense flight demonstrations remain limited in scale and environmental coverage, constraining the continuous control experience for training diverse navigation policies.

Reconstructed Environments and Navigation Benchmarks. Recent embodied-navigation research uses 3D Gaussian Splatting to reconstruct photorealistic real-world environments, providing a real-to-sim bridge for policy learning and evaluation (Chhablani et al., 2025; Liu et al., 2026a; Low et al., 2025; Quach et al., 2024). To support collision-aware navigation, these representations are augmented with metric geometry, collision models, and navigation structures, enabling interactive navigation across reconstructed environments (Miao et al., 2026; He et al., 2026; Liu et al., 2026b; Xia et al., 2026). Aerial benchmarks also pair 3DGS environments with geometry for collision checking and evaluation: OpenFly spans 18 scenes from multiple rendering sources, while HUGE provides four reconstructed outdoor environments (Gao et al., 2026; Guo et al., 2026). These platforms establish important foundations, but broader coverage of reconstructed indoor, urban outdoor, and vegetation environments is still needed for systematic cross-scene policy evaluation.

Navigation Policy Evaluation. Offline evaluation measures prediction errors against expert demonstrations, while interactive aerial benchmarks assess task success and path efficiency through closed-loop rollouts (Xue et al., 2026; Lou et al., 2026). Studies in autonomous driving and robot manipulation examine this mismatch, including the predictive value of evaluation metrics and task-critical errors (Wang et al., 2026; Huang et al., 2026). This mismatch also depends on how predictions are executed: work on action-chunking policies investigates execution horizons (Zhao et al., 2026), while studies of aerial agents examine how execution mechanisms affect flight outcomes (Xu et al., 2026). However, a systematic understanding of the offline-to-closed-loop gap in aerial navigation remains limited. By pairing flight demonstrations with executable worlds, AirPlayground supports trajectory replay and policy evaluation in corresponding scenes, providing an experimental basis for tracing offline predictions to their closed-loop consequences.

3 AIRPLAYGROUND BENCHMARK

This section describes how AirPlayground builds executable environments and dense expert flight experience from real-world captures, and supports policy training and evaluation through a unified interface. Figure 2 summarizes the construction pipeline, from scene capture and reconstruction to scene-adaptive task generation, expert flight collection, and the resulting shared flight corpus.

3.1 SCENE COLLECTION

Capture hardware. An operator walks each site at a steady pace carrying an XGRIDS Lixel P1 PortalCam, a handheld scanner that pairs a wide-field LiDAR with four synchronized cameras, two fisheye and two forward-facing. A single walk-through therefore yields both metric geometry and multi-view imagery of the scene (Appendix A.1).

Scene selection and diversity. We choose environments spanning three settings with distinct navigation demands: *indoor* spaces such as rooms and corridors, with confined free space and clutter; *urban outdoor* sites such as campuses, streets and landmarks, with regular structures; and *vegetation*-rich environments such as parks, gardens and terrain, with irregular geometry and diffuse obstacle boundaries. Scenes are captured on site or adapted from publicly available assets (Xia et al., 2026), then processed into a common representation (Sec. 3.2). The corpus contains 94 scenes: 18 indoor, 48 urban outdoor and 28 vegetation. Navigable regions total 75,657 m^2, ranging from 14 to 6,674 m^2 per scene (median 567 m^2; mean 805 m^2), spanning over two orders of magnitude.

3.2 EXECUTABLE SCENE RECONSTRUCTION

Photorealistic 3DGS reconstruction. For scenes captured on site, the synchronized LiDAR and camera streams are processed by an end-to-end reconstruction system. It follows the tightly-coupled

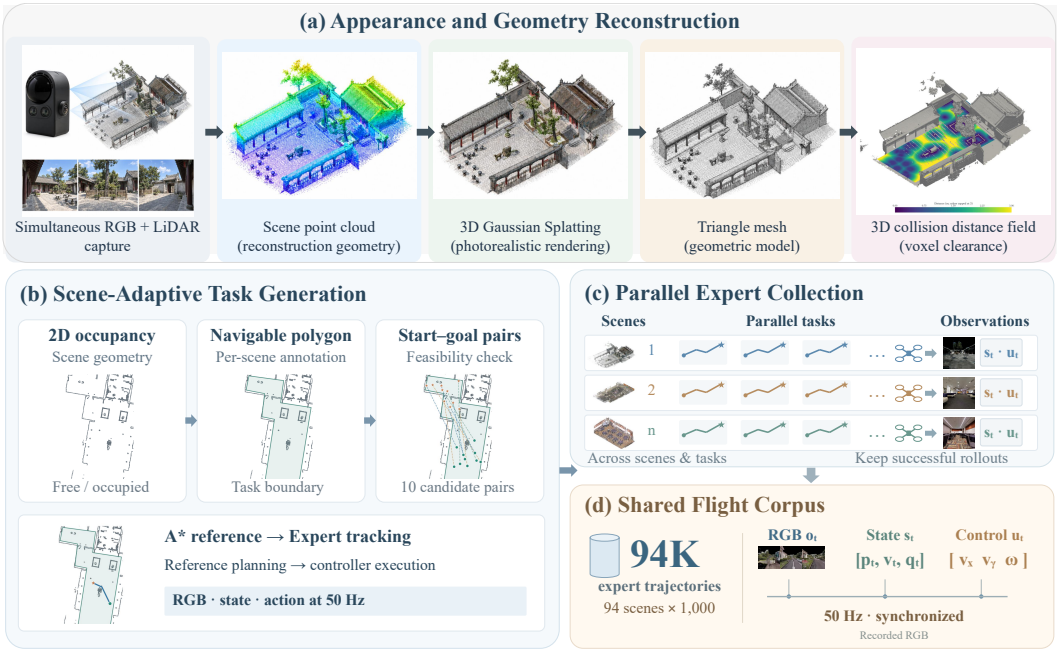


Figure 2: **AirPlayground framework.** Real-world captures become executable 3DGS scenes for closed-loop expert collection. A unified cross-policy API supports training on the resulting demonstrations and evaluation in the corresponding scenes.

LiDAR–visual–inertial route to state estimation (Shan et al., 2021; Lin et al., 2021; Zheng et al., 2025), recovering the sensor trajectory and a point map at metric scale. Fisheye images are undistorted, and a 3D Gaussian Splatting model (Kerbl et al., 2023) is initialized from the point cloud and optimized under depth supervision on the registered images. The pipeline detects dynamic regions and segments them into masks, and the masked regions are held out of the optimization so that transient content does not enter the reconstruction. Additional assets are brought into the same form, so every scene ends up as a 3DGS asset paired with a metric point cloud in one frame.

Executable geometry. A visual asset shows what a scene looks like, but not where a drone may fly or what it would strike. We derive both from the point cloud. A 2D occupancy grid defines the free space for task sampling and path planning. It covers a flight-relevant height band, is restricted to a navigable polygon annotated once per scene, and dilates occupied cells by a per-scene inflation radius so paths keep a margin from obstacles. A 3D clearance field, precomputed once per scene without inflation, answers collision and clearance queries against the robot radius and preserves the vertical structure the 2D grid collapses; GPU queries batch this checking without physics-engine contacts. The 3DGS asset, planning grid, and clearance field share one coordinate frame, so visual observations, task planning, and collision checking refer to the same scene (Appendix A.1).

3.3 Scene-Adaptive Demonstration Generation

Scene-adaptive task generation. A scene specification records each world's occupancy and inflation settings, navigable polygon and flight altitude. Within the navigable region, the pipeline samples start–goal pairs subject to a minimum separation and a clearance margin from obstacles, keeps a pair only if a planned path connects its endpoints, and resamples otherwise. The initial heading is randomized around the goal direction, so the flown trajectory is not fully determined by the planned path. Where a scene offers little natural obstacle interaction, the pipeline adds box-shaped obstacles in free space to diversify trajectories; the released corpus records their count per episode (Appendix A.2).

Closed-loop demonstration collection. For each task, A* plans a reference path over the inflated occupancy grid, and greedy line-of-sight shortcutting then shortens it. The expert follows this path in closed loop: at each 50 Hz control step, a waypoint-tracking expert reads the drone's current simulated state, steers toward the active waypoint, and outputs planar body-frame velocities and a yaw-rate command. Each step synchronously records forward RGB, the UAV state, a local occupancy patch and the executed action. Only successful episodes are retained. Worlds are collected independently and in parallel across GPUs, targeting 1,000 successful demonstrations per scene and yielding the corpus reported in Sec. 3.5.

3.4 Unified Cross-Policy API

Navigation policies differ in observation histories, training targets, and inference and execution mechanisms. AirPlayground handles these differences through training and stateful control adapters, allowing policies to reuse the same flight corpus and evaluation pipeline. The API supports point-goal, image-goal, and goal-free navigation.

Given a trajectory $\tau = \{(o_t, s_t, g_t, a_t)\}_{t=1}^{T}$, a training adapter converts it into the inputs and supervision of policy m, $(\mathbf{x}^m, \mathbf{y}^m) = \mathcal{T}_m(\tau)$, constructing observation histories and policy-specific targets, including single-step actions, action chunks, waypoints, or future observations for predictive models. Future observations serve as training supervision, not deployment inputs. For goal-free policies, the adapter withholds the goal input g_t and the relative-goal entries of the state s_t.

At deployment, a stateful control adapter maps policy outputs, including actions selected by model-based planning, to body-frame velocity commands, $\mathbf{u}_t = \mathcal{C}_m(\hat{\mathbf{y}}_t^m, s_t; \mathbf{h}_t^m)$ with $\mathbf{u}_t = (v_x, v_y, \dot{\psi})$, where $\mathbf{h}_t^m$ tracks cached predictions and progress through the current plan. It executes each policy's outputs on its own execution schedule, standardizing task inputs and control semantics while preserving method-specific prediction and planning mechanisms. New policies implement these adapters while reusing the task generator, simulator, and evaluator.

3.5 Resource Statistics and Validation

Corpus scale and schema. AirPlayground contains 94 executable worlds and 94,000 successful expert trajectories, 1,000 per world. Recorded at 50 Hz, the trajectories comprise 46,530,557 synchronized RGB–state–action samples, 495 per trajectory on average. They amount to roughly 259 hours and 1,800 km of flight, stored in 5.9 TB. Each sample pairs a forward 256×256 RGB image with a 13-dimensional UAV state that embeds the relative goal, a local occupancy patch, and a continuous body-frame control action (u_x, u_y, ω) whose translational components are normalized (Appendix A.5). This schema lets every policy formulation in Sec. 4 read from the same trajectories.

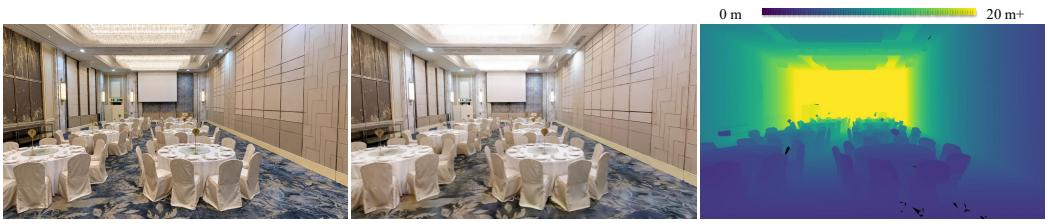
Comparison with existing aerial resources. Table 1 compares environment representations, scale, and flight supervision, which differ in granularity and coverage beyond raw trajectory counts. CityNav and OpenFly provide discrete navigation supervision, with OpenFly providing waypoints. UAV-Flow records flight at 5 Hz in real footage, and FLIGHT at 10 Hz in synthetic environments, but neither uses 3DGS reconstructions. HUGE-Bench pairs 3DGS with collision geometry across four scenes. AirPlayground combines 94 executable worlds with 94K trajectories and continuous body-frame control supervision at 50 Hz.

Trajectory statistics and validation. Planned paths average 19.2 m (median 13.5 m, 95th percentile 55.6 m, maximum 235.1 m), and episodes last 9.9 s on average (median 7.1 s, 95th percentile 27.6 s). Per-scene frame counts range from 43,665 to 2,023,166. All 94,000 released trajectories are successful expert demonstrations, and every one of them is accounted for in storage, with no missing and no orphaned files. All baselines in Sec. 4 use this corpus for training or adaptation and offline evaluation (Appendix A.5).

Visual appearance and geometry. We evaluate rendering fidelity across 20 scenes with 10 reference images per scene, using camera poses estimated from image correspondences and mesh depth. At 1280×720 resolution, the evaluation yields an average PSNR of 19.19 dB, SSIM of 0.655, and LPIPS (AlexNet) of 0.223. Figure 3 illustrates the correspondence between a reference photograph and its 3DGS rendering, preserving the room layout and major furnishings with some differences

Table 1: **Comparison of aerial navigation resources.** AirPlayground pairs executable 3DGS reconstructions with continuous supervision. "–" indicates an absent property or unreported quantity.

	Environment			Scale		Flight Experience	
Benchmark	Source	3DGS	Exec.	Scenes	Traj.	Supervision	Granularity
AerialVLN (Liu et al., 2023)	Simulation	–	✓	25	8.4K	Discrete 4-DoF	∼204 steps/traj.
CityNav (Lee et al., 2025)	Real-city scans	–	✓	2 cities	32.6K	Discrete actions	–
OpenUAV (Wang et al., 2025b)	Simulation	–	✓	22	12.1K	6-DoF trajectory	∼264 steps/traj.
OpenFly (Gao et al., 2026)	Mixed	✓	Partial	18	100K	Discrete + waypoint	∼35 actions/traj.
UAV-Flow (Wang et al., 2025a)	Real + sim.	–	✓	3 real	40.8K	6-DoF state	5 Hz
FLIGHT (Zheng et al., 2026)	Simulation	–	✓	–	10.8K	Continuous 6-DoF	10 Hz
HUGE-Bench (Guo et al., 2026)	Digital twins	✓	✓	4	6.2K	RGB + state	∼1 m spacing
AirPlayground	**Real recon.**	✓	✓	**94**	**94K**	**RGB + state + velocity**	**50 Hz**



(a) Real photograph (b) 3DGS rendering (c) Mesh depth

Figure 3: **Visual appearance and scene geometry.** (a) Reference photograph, (b) 3DGS rendering, and (c) mesh-derived depth at the matched viewpoint.

in fine textures and lighting. Mesh-derived depth depicts the corresponding obstacle surfaces used for collision checking. Together, these results provide evidence of visual fidelity and illustrate how AirPlayground pairs rendered observations with executable scene geometry.

4 Experiments

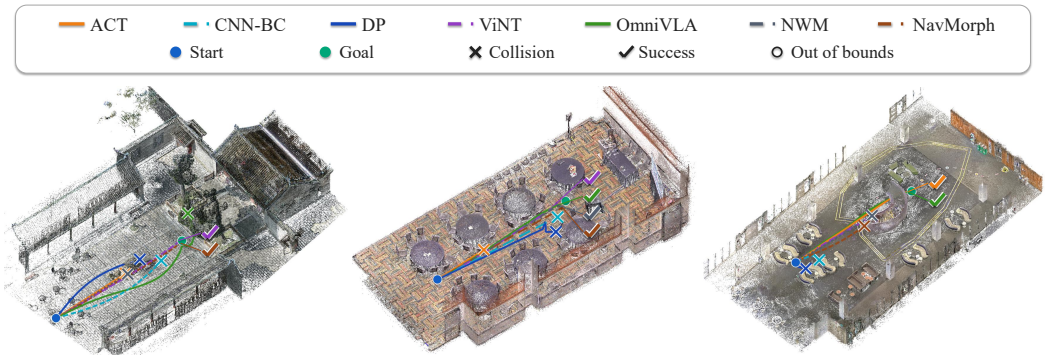
We evaluate AirPlayground through three central research questions: (1) How do aerial navigation policies perform across diverse reconstructed environments under a shared evaluation protocol, and how well do they generalize to unseen scenes? (2) How does increasing available training experience affect closed-loop performance? (3) What explains the observed discrepancies between offline prediction accuracy and closed-loop navigation performance?

4.1 Experimental Setup

Benchmark. We split 94 environments into 70 training, 8 validation, and 16 unseen test scenes, with all three categories in each split. Each scene has 1,000 trajectories; training scenes reserve 100 for seen offline evaluation and use 900 for learning. Closed-loop evaluation uses 50 fixed episodes in each of 16 training scenes and all 16 unseen test scenes: 800 per split and 1,600 per policy and setting. Split scores weight scenes equally. Offline-to-closed-loop comparisons use matched scenes.

Task protocol. Policies navigate at a fixed flight height using forward RGB, flight state, and task-specific goal inputs, and output planar body-frame velocities and yaw rate. PointGoal supplies a relative target; ImageGoal uses successive views along an A* reference route; Goal-free withholds the target, which is retained only for evaluation. Success requires reaching within 0.45 m of the goal without a safety violation. Episodes end on success, collision, leaving the navigable region, or a 100 s timeout; collision checking uses a 0.12 m body radius.

Baselines. We benchmark seven baselines: CNN-BC, ACT (Zhao et al., 2023), Diffusion Policy (DP) (Chi et al., 2025), ViNT (Shah et al., 2023), OmniVLA (Hirose et al., 2025), NWM (Bar et al., 2025), and NavMorph (Yao et al., 2025). All are trained or adapted on the shared corpus and compared primarily in PointGoal mode under unified protocols while preserving their native prediction



(a) Scene 1 (b) Scene 2 (c) Scene 3

Figure 4: **PointGoal rollouts with shared starts and goals in three scenes.**

Table 2: **Offline and closed-loop PointGoal performance on seen/unseen scenes.** Offline errors use a 0.2 s horizon: ADE/FDE/P95 in cm; mean absolute endpoint yaw error in radians. P95 denotes scene-averaged 95th-percentile FDE. Closed-loop protocol: Sec. 4.1.

	Seen Offline				Seen Closed Loop				Unseen Offline				Unseen Closed Loop			
Method	ADE↓	FDE↓	P95↓	Yaw↓	SR↑	A*-SPL↑	CR↓	OB↓	ADE↓	FDE↓	P95↓	Yaw↓	SR↑	A*-SPL↑	CR↓	OB↓
CNN-BC	**0.54**	**0.81**	**2.52**	**0.0022**	21.0	0.21	49.0	30.0	**9.34**	**16.59**	**19.75**	**0.0054**	15.0	0.14	47.5	37.5
ACT	1.67	3.07	16.86	0.0096	40.8	0.41	39.2	20.0	11.84	21.55	26.68	0.0171	29.5	0.30	65.0	5.5
DP	2.88	5.38	32.98	0.0316	20.5	0.19	65.5	14.0	22.15	40.27	58.04	0.1235	11.2	0.10	48.8	40.0
ViNT	7.11	12.81	19.85	0.0360	38.8	0.39	36.2	25.0	15.13	27.50	38.24	0.0422	33.0	0.33	64.5	2.5
OmniVLA	4.99	8.11	15.01	0.0454	47.5	0.47	38.8	13.7	20.64	37.95	45.58	0.0497	**40.5**	0.38	59.5	**0.0**
NWM	8.70	15.81	21.60	0.0320	**50.0**	**0.50**	**27.5**	22.5	13.06	23.72	35.33	0.0386	38.6	**0.39**	54.3	7.1
NavMorph	2.31	3.89	13.91	0.0347	35.5	0.35	59.5	**5.0**	11.98	21.64	28.38	0.0180	22.5	0.22	**46.3**	31.2

and execution mechanisms. Checkpoints and execution intervals are selected on validation scenes and fixed for evaluation (adapter configurations in Appendix B.2).

Metrics. Offline metrics include average and final displacement errors (ADE/FDE), P95 FDE, and mean absolute endpoint yaw error over a common physical horizon. Closed-loop metrics include success rate (SR), A*-SPL, navigation error (NE), collision rate (CR), and out-of-bounds rate (OB). Definitions and additional diagnostics appear in Appendix B.1.

4.2 Unified Policy Benchmark

Unified evaluation reveals complementary policy strengths. Evaluating different policies on the same flight corpus and task reveals complementary strengths across metrics and scene splits (Table 2). NWM achieves the highest seen-scene SR and A*-SPL with the lowest CR, while NavMorph attains the lowest OB. On unseen scenes, OmniVLA leads in SR and OB, NWM in A*-SPL, and NavMorph in CR. Comparable success rates can mask execution behaviors: NWM and OmniVLA achieve seen-scene SRs of 50.0% and 47.5%, yet NWM exhibits lower control variation and path turning (ControlVar 0.003; Turn/m 0.120) than OmniVLA (0.540; 3.919). Across navigation settings, ImageGoal with successive image subgoals yields higher SR than PointGoal for three of seven baselines, while withholding goal input lowers SR for all seven (Table 10), revealing policy-dependent responses to goal guidance. Figure 4 illustrates paths and outcomes in three scenes.

Better offline prediction does not always yield better navigation. CNN-BC achieves the lowest seen-scene ADE and FDE but only 21.0% SR. Averaging seen and unseen scores for each policy, ADE and FDE have Spearman correlations of $\rho = 0.29$ with SR, opposite to the expected negative relationship. Their correlations with CR are weak ($\rho = 0.18$), while FDE correlates negatively with OB ($\rho = -0.43$), contrary to expectation. Thus, lower displacement error does not consistently indicate better closed-loop performance, motivating examination of deployment-relevant error structure.

Scaling flight experience improves closed-loop generalization. We study scaling using ACT trained on nested subsets of 5, 15, 35, and 70 scenes, with 900 trajectories per scene. We keep the

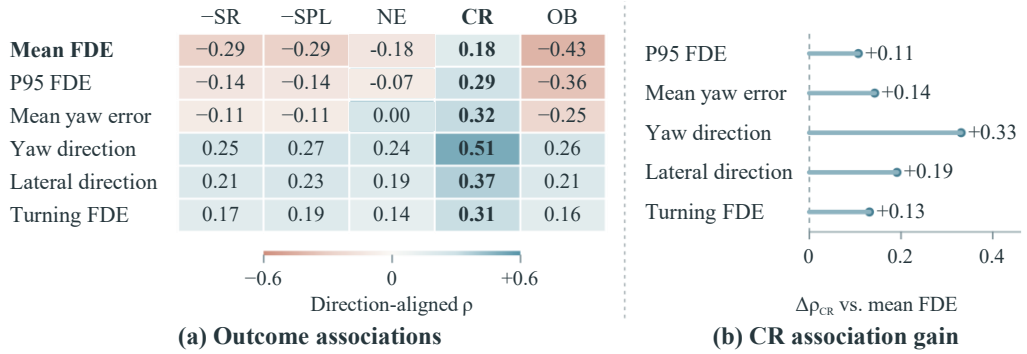


Figure 5: **Offline diagnostics and closed-loop outcomes.** (a) Direction-aligned Spearman correlations: positive correlations link errors to worse outcomes. (b) Signed correlation differences for CR relative to mean FDE. Correlations use seven scores, averaging seen/unseen splits (Appendix B.4.1).

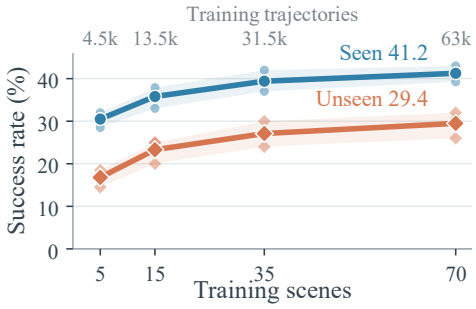


Figure 6: **ACT data scaling.** Three training seeds evaluated on five seen and five unseen scenes, with 100 fixed episodes per scene.

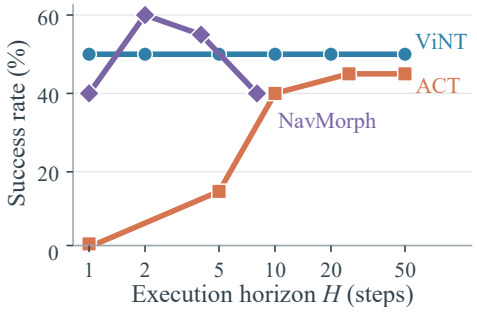


Figure 7: **Execution-horizon sensitivity.** Averaged SR over 5 seen and 5 unseen scenes; 50 episodes per scene, unchanged checkpoints.

optimization settings and epochs fixed and train each configuration with three seeds. The scaling protocol uses 100 fixed episodes per scene in the five seen scenes shared by all training subsets and five fixed unseen scenes (500 episodes per split and seed), with identical evaluation tasks across training scales and seeds. In Figure 6, median success rates across the three seeds rise over the four scales on both splits: unseen-scene SR from 16.8% to 29.4%, seen-scene SR from 30.6% to 41.2%.

4.3 From Prediction to Consequence

Sec. 4.2 shows that aggregate offline errors do not consistently track closed-loop outcomes. Using paired demonstrations and executable scenes, we examine what error structure, execution timing, and local geometry reveal about this mismatch.

Error structure matters for deployment diagnosis. We relate fine-grained offline errors to aggregated closed-loop outcomes across policies (Fig. 5(a)). Mean FDE shows a weak positive association with CR ($\rho = 0.18$), whereas yaw direction error, lateral direction error, and turning-subset FDE show stronger associations ($\rho = 0.51$, 0.37, and 0.31, respectively). Figure 5(b) shows signed correlation differences relative to mean FDE, reaching $+0.33$ for CR. These results suggest that directional and maneuver-conditioned errors retain collision-related information beyond mean FDE, highlighting their potential for offline safety diagnosis.

Execution horizon changes more than query frequency. Offline prediction scores do not capture how predictions are consumed during interaction. We vary the steps H between model queries across five seen and five unseen scenes, using 50 episodes per scene and the same checkpoints and tasks across horizons. For ACT, each query replaces the previous action chunk, and only its first H controls are executed, without temporal ensembling. Figure 7 shows policy-dependent responses:

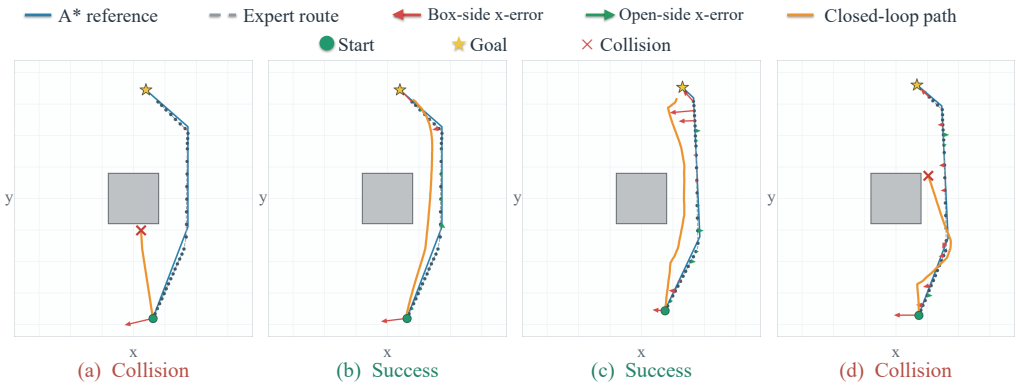


Figure 8: **Offline prediction errors and closed-loop trajectories around an obstacle.**

Table 3: **Outcomes by predicted clearance change.** Group-mean distances; counts out of N.

Change	N	FDE (cm)	Expert → Pred. (cm)	Closed-loop min. (cm)	Collision	Local pass
Decrease	43	8.4	25.6 → 18.9	5.2	16/43 (37.2%)	25/43 (58.1%)
Stable	76	7.9	26.2 → 26.5	10.8	16/76 (21.1%)	56/76 (73.7%)
Increase	61	8.8	25.9 → 32.1	15.7	8/61 (13.1%)	49/61 (80.3%)

ACT's SR rises from 0% at $H = 1$ to 45% at longer horizons, NavMorph performs best at an intermediate horizon, and ViNT's SR remains unchanged across tested settings. More frequent policy queries therefore do not necessarily improve navigation.

Local geometry modulates the safety cost of action deviations. Prediction error measures deviation from an expert trajectory but not its effect on obstacle clearance. Figure 8(a–b) shows opposite outcomes under similar layouts and visually similar offline error patterns; in (d), the rollout initially follows the reference but later collides. These cases highlight the importance of where deviations occur relative to obstacles. We group 180 local obstacle-interaction samples by the difference between predicted and expert minimum clearance over the same prediction window, using ± 2 cm as the stable range. Local passage requires reaching the predefined exit within 5 s without collision or leaving the navigable region. Despite similar group-average prediction errors and expert clearances, the clearance-decrease group has lower closed-loop minimum clearance (5.2 versus 15.7 cm), a higher reference-obstacle collision rate (37.2% versus 13.1%), and a lower local passage rate (58.1% versus 80.3%) than the increase group (Table 3). These associations highlight safety consequences not directly described by displacement error, motivating geometry- and collision-aware policy learning.

5 Conclusion

We introduce AirPlayground, a unified benchmark for aerial navigation that pairs dense flight experience and scene geometry with executable real-world reconstructions. By adapting shared trajectories and unifying deployment controls, our cross-policy API enables different policy formulations to learn from the same flight corpus and undergo standardized evaluation across diverse environments. We benchmark seven baselines spanning four policy paradigms across three navigation settings in seen and unseen scenes, while providing a unified interface for additional policies. Beyond measuring task performance, our analyses reveal an offline-to-closed-loop mismatch involving error structure, execution protocols, and interactions with scene geometry. Together, these analyses demonstrate how AirPlayground combines standardized performance comparison with deployment diagnosis in the corresponding environments. We will publicly release the flight dataset, cross-policy API, and benchmark tools. We hope AirPlayground will serve as a foundation for aerial policy learning and evaluation, helping researchers train policies on dense flight experience, diagnose behavior in executable worlds, and explore geometry- and collision-aware learning.

AI USE STATEMENT

We used generative AI tools solely to improve the grammar and clarity of text originally drafted by the authors. These tools were not used for research ideation, method design, code implementation, experiment execution, or analysis and interpretation of results. The authors reviewed all AI-assisted text to ensure that it accurately reflects their own work and findings. We take full responsibility for the final content of this paper, including all text, claims, and artifacts.

REFERENCES

Maximilian Adang, JunEn Low, Ola Shorinwa, and Mac Schwager. SINGER: An onboard generalist vision-language navigation policy for drones. *arXiv preprint arXiv:2509.18610*, 2025.

Amir Bar, Gaoyue Zhou, Danny Tran, Trevor Darrell, and Yann LeCun. Navigation world models. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15791–15801. IEEE, 2025.

Gunjan Chhablani, Xiaomeng Ye, Muhammad Zubair Irshad, and Zsolt Kira. EmbodiedSplat: Personalized real-to-sim-to-real navigation with Gaussian splats from a mobile device. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 25431–25441. IEEE, 2025.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

Yunpeng Gao, Chenhui Li, Zhongrui You, Junli Liu, Zhen Li, Pengan Chen, Qizhi Chen, Zhonghan Tang, Liansheng Wang, Penghui Yang, Yiwen Tang, Yuhang Tang, Shuai Liang, Songyi Zhu, Ziqin Xiong, Yifei Su, Xinyi Ye, Jianan Li, Yan Ding, Dong Wang, Xuelong Li, Zhigang Wang, and Bin Zhao. OpenFly: A comprehensive platform for aerial vision-language navigation. In *International Conference on Learning Representations*, 2026.

Jingyu Guo, Ziye Chen, Ziwen Li, Zhengqing Gao, Jiaxin Huang, Hanlue Zhang, Fengming Huang, Yu Yao, Tongliang Liu, and Mingming Gong. HUGE-Bench: A benchmark for high-level UAV vision-language-action tasks. *arXiv preprint arXiv:2603.19822*, 2026.

Honglin He, Yukai Ma, Brad Squicciarini, Wayne Wu, and Bolei Zhou. From seeing to experiencing: Scaling navigation foundation models with reinforcement learning. In *International Conference on Learning Representations*, 2026.

Noriaki Hirose, Catherine Glossop, Dhruv Shah, and Sergey Levine. OmniVLA: An omni-modal vision-language-action model for robot navigation. *arXiv preprint arXiv:2509.19480*, 2025.

Haoxu Huang, Tongsam Zheng, Yifan Chen, Jiacheng You, and Yang Gao. Critical interval MSE: Toward reliable offline validation for robot manipulation policies. *arXiv preprint arXiv:2606.29898*, 2026.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics*, 42(4):139:1–139:14, 2023. doi: 10.1145/3592433.

Jungdae Lee, Taiki Miyanishi, Shuhei Kurita, Koya Sakamoto, Daichi Azuma, Yutaka Matsuo, and Nakamasa Inoue. CityNav: A large-scale dataset for real-world aerial navigation. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 5912–5922. IEEE, 2025.

Jiarong Lin, Chunran Zheng, Wei Xu, and Fu Zhang. R^2LIVE: A robust, real-time, LiDAR-inertial-visual tightly-coupled state estimator and mapping. *IEEE Robotics and Automation Letters*, 6(4): 7469–7476, 2021.

Jiahang Liu, Yuanxing Duan, Jiazhao Zhang, Minghan Li, Shaoan Wang, Zhizheng Zhang, and He Wang. NavGSim: High-fidelity Gaussian splatting simulator for large-scale navigation. *arXiv preprint arXiv:2603.15186*, 2026a.

Shubo Liu, Hongsheng Zhang, Yuankai Qi, Peng Wang, Yanning Zhang, and Qi Wu. AerialVLN: Vision-and-language navigation for UAVs. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 15384–15394. IEEE, 2023.

Xinhao Liu, Jintong Li, Yicheng Jiang, Niranjan Sujay, Zhicheng Yang, Juexiao Zhang, John Abanes, Jing Zhang, and Chen Feng. CityWalker: Learning embodied urban navigation from web-scale videos. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6875–6885. IEEE, 2025.

Xinhao Liu, Jiaqi Li, Youming Deng, Ruxin Chen, Yingjia Zhang, Yifei Ma, Li Guo, Yiming Li, Jing Zhang, and Chen Feng. Wanderland: Geometrically grounded simulation for open-world embodied AI. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1041–1052, 2026b.

Xu Liu, Yu Liu, Hanshuo Qiu, Yang Qirong, and Zhouhui Lian. IndoorUAV: Benchmarking vision-language UAV navigation in continuous indoor environments. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(28):23864–23872, 2026c. doi: 10.1609/aaai.v40i28.39562.

Jiabin Lou, Haopeng Wang, Yuanshuai Wang, Xinyu Liu, Xuxin Lv, Yuxin Guo, Lei Huang, Rongye Shi, and Wenjun Wu. VoLN: Vision-only long-horizon navigation—paradigm, benchmark, and method. *arXiv preprint arXiv:2607.21400*, 2026.

JunEn Low, Maximilian Adang, Javier Yu, Keiko Nagami, and Mac Schwager. SOUS VIDE: Cooking visual drone navigation policies in a Gaussian splatting vacuum. *IEEE Robotics and Automation Letters*, 10(5):5122–5129, 2025.

Bingchen Miao, Rong Wei, Zhiqi Ge, Xiaoquan Sun, Shiqi Gao, Jingzhe Zhu, Renhan Wang, Siliang Tang, Jun Xiao, Rui Tang, and Juncheng Li. Towards physically executable 3D Gaussian for embodied navigation. In *International Conference on Learning Representations*, 2026.

Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open X-Embodiment: Robotic learning datasets and RT-X models. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6892–6903. IEEE, 2024.

Alex Quach, Makram Chahine, Alexander Amini, Ramin Hasani, and Daniela Rus. Gaussian splatting to real world flight navigation transfer with liquid networks. *arXiv preprint arXiv:2406.15149*, 2024.

Seokin Seo, Byung-Jun Lee, Jongmin Lee, HyeongJoo Hwang, Hongseok Yang, and Kee-Eung Kim. Mitigating covariate shift in behavioral cloning via robust stationary distribution correction. *Advances in Neural Information Processing Systems*, 37:109177–109201, 2024.

Dhruv Shah, Ajay Sridhar, Nitish Dashora, Kyle Stachowicz, Kevin Black, Noriaki Hirose, and Sergey Levine. ViNT: A foundation model for visual navigation. *arXiv preprint arXiv:2306.14846*, 2023.

Tixiao Shan, Brendan Englot, Carlo Ratti, and Daniela Rus. LVI-SAM: Tightly-coupled lidar-visual-inertial odometry via smoothing and mapping. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5692–5698, 2021.

Xiangyu Wang, Donglin Yang, Yue Liao, Wenhao Zheng, Wenjun Wu, Bin Dai, Hongsheng Li, and Si Liu. UAV-Flow Colosseo: A real-world benchmark for flying-on-a-word UAV imitation learning. *Advances in Neural Information Processing Systems*, 38, 2025a.

Xiangyu Wang, Donglin Yang, Ziqin Wang, Hohin Kwan, Jinyu Chen, Wenjun Wu, Hongsheng Li, Yue Liao, and Si Liu. Towards realistic UAV vision-language navigation: Platform, benchmark, and methodology. In *International Conference on Learning Representations*, 2025b.

Yiru Wang, Anqing Jiang, Shuo Wang, Yuwen Heng, Hai Yang, Yang Chen, and Hao Sun. Do open-loop metrics predict closed-loop driving? a cross-benchmark correlation study of NAVSIM and Bench2Drive. *arXiv preprint arXiv:2605.00066*, 2026.

Ziyuan Xia, Jingyi Xu, Chong Cui, Yuanhong Yu, Jiazhao Zhang, Qingsong Yan, Tao Ni, Junbo Chen, Xiaowei Zhou, Hujun Bao, Ruizhen Hu, and Sida Peng. Habitat-GS: A high-fidelity navigation simulator with dynamic Gaussian splatting. *arXiv preprint arXiv:2604.12626*, 2026.

Yiqing Xu, Haifeng Lin, Yujin Yang, Ji'An Xia, and Zidong Han. A lightweight vision–language–action policy with progress-aware hybrid execution for UAV waypoint navigation in AirSim. *Sensors*, 26(14):4655, 2026. doi: 10.3390/s26144655.

Fanfu Xue, En Yu, Yantian Shen, Zhikun Hu, Hongjun Wang, Yang Yang, Xindi Wang, and Jiande Sun. See-and-Reach: Precise vision-language navigation for UAVs within the field of view. *arXiv preprint arXiv:2606.20045*, 2026.

Xuan Yao, Junyu Gao, and Changsheng Xu. NavMorph: A self-evolving world model for vision-and-language navigation in continuous environments. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 5536–5546. IEEE, 2025.

Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

Yuchi Zhao, Miroslav Bogdanovic, Arjun Sohal, Liyu Tao, Kourosh Darvish, Alán Aspuru-Guzik, Florian Shkurti, and Animesh Garg. Dynamic execution horizon prediction for chunk-based robot policies. *arXiv preprint arXiv:2606.11408*, 2026.

Chunran Zheng, Wei Xu, Zuhao Zou, Tong Hua, Chongjian Yuan, Dongjiao He, Bingyang Zhou, Zheng Liu, Jiarong Lin, Fangcheng Zhu, Yunfan Ren, Rong Wang, Fanle Meng, and Fu Zhang. FAST-LIVO2: Fast, Direct LiDAR-Inertial-Visual Odometry. *IEEE Transactions on Robotics*, 41:326–346, 2025.

Duo Zheng, Shijia Huang, Lin Zhao, Yiwu Zhong, and Liwei Wang. Towards learning a generalist model for embodied navigation. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13624–13634. IEEE, 2024.

Xiangyi Zheng, Xiangyu Wang, Qinan Liao, Zimu Tang, Yue Liao, Dongyue Lyu, Guodong Wang, Junjie Liu, and Si Liu. Think like a pilot: Fine-grained long-horizon UAV navigation. *arXiv preprint arXiv:2606.06836*, 2026.

A Demonstration Collection Protocol

AirPlayground generates navigation demonstrations by sampling tasks in reconstructed scenes, planning reference paths, and executing a feedback controller in the simulator. The collection unit is a scene-specific PointGoal episode. Each episode contains observations and control commands from the executed flight, together with the task configuration and reference path. The following description details the base collection pipeline and separately documents its optional diversity mechanisms. Table 4 lists the reference implementation settings; scene configuration files and launch arguments determine the settings of an individual collection run.

A.1 Scene Configuration and Navigable Space

Each scene is associated with a rendered 3DGS asset, geometric point clouds, and an annotated navigable polygon. A scene configuration specifies the geometric transformation, the height interval used for obstacle extraction, the obstacle inflation radius, and the initial flight height or height range. The collection code resolves these scene-specific settings before constructing the planning grid, so a scene's planning grid is built the same way in every run. Settings that vary between collection runs, such as the number of virtual obstacles, are supplied on the command line.

Capture device. Scenes captured on site are recorded with an XGRIDS Lixel P1 PortalCam. Its LiDAR is a Class 1 940 nm unit with a $180^\circ \times 180^\circ$ field of view and a rate of 856,000 points per second, ranging to 30 m against 10% reflectivity and to 60 m against 90%. Its camera array comprises two fisheye cameras ($200^\circ \times 200^\circ$) and two forward-facing cameras ($100^\circ \times 85^\circ$), each at 4000×3000 resolution. The unit carries no RTK receiver.

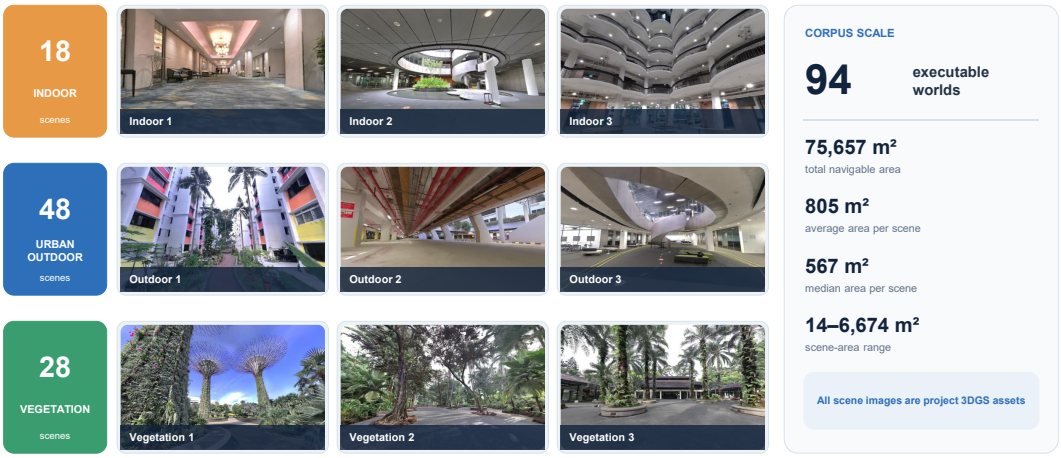


Figure 9: **Scene diversity and corpus scale.** Representative views across indoor, urban outdoor, and vegetation environments.

Scene assets and frames. A scene ships as a USD package holding the renderable asset together with a triangle mesh of the same reconstruction, alongside a separate point-cloud file. Because all of these come from one reconstruction, they already share a frame; the collection code applies a single fixed map transform to bring them into the simulation frame, and no pairwise registration between appearance and geometry is performed. The packaged mesh is exposed to the environment as a ray-cast target for depth queries. The clearance field described below is what supplies collision geometry, and it is derived from the point cloud rather than from that mesh.

Occupancy construction. Point-cloud coordinates are scaled, rotated, and translated into the simulation frame. Points within the configured height interval are projected onto a horizontal grid with cell width δ. A cell is initially occupied when its point count reaches the configured threshold; an optional vertical-extent threshold further filters the projected points. Small connected components are removed before obstacle inflation. Occupied cells are then dilated by a discrete circular neighborhood with radius $\lceil r_{\text{inf}}/\delta \rceil$, where r_{inf} is the configured inflation distance. Finally, cells whose centers lie outside the navigable polygon are marked occupied. The resulting grid defines the feasible domain for sampling and path planning.

The planning grid describes horizontal traversability. A feasible grid path is followed by an actual closed-loop rollout before the episode is accepted, so grid feasibility and successful execution are separate checks in the collection procedure.

Collision distance field. Because the planning grid is horizontal, it cannot distinguish an obstacle that is dangerous only at ceiling height from one that is dangerous at flight height. A separate 3D clearance field supplies that distinction. We first rebuild the footprint above without inflation, then voxelize the height-filtered, polygon-cropped point cloud on the same horizontal grid extended by a vertical axis at the same resolution, and keep only voxels whose (x, y) column survives the cleaned footprint. This retains the two-dimensional denoising while recovering height structure, which a 3D connected-component filter applied directly would fragment. A Euclidean distance transform over the remaining free voxels, scaled by the voxel size, yields the field, stored with its origin and resolution.

The bake applies no inflation. At run time a query point is mapped to its voxel by flooring, the stored clearance is read in a vectorized gather on the simulation device, and the robot radius is subtracted; a non-positive result signals collision. Points outside the baked volume are treated as free, and departures from the scene are handled separately by out-of-bounds and height checks. The bake reuses the simulator's map transform so that the clearance field, the planning grid and the rendered asset share one frame; this correspondence has to be maintained whenever a scene overrides that transform.

A.2 Task Sampling and Initialization

Candidate generation. The sampler draws horizontal positions within the polygon's bounding box and rejects positions outside the polygon. For endpoint sampling, it further dilates the planning occupancy by an endpoint-clearance margin. Candidate positions are mapped to nearby free cells in this sampling mask and checked again for polygon membership. Duplicate cells are removed, yielding a set of up to N candidate locations.

Start–goal selection. For candidate locations $\{\mathbf{c}_i\}_{i=1}^{N}$, the sampler constructs eligible unordered pairs

$$\mathcal{E} = \{(i, j) : i < j,\ \|\mathbf{c}_i - \mathbf{c}_j\|_2 \geq \max(d_{\min}, 2\delta)\}, \tag{1}$$

where $d_{\min}$ is the minimum start–goal separation. A pair is selected uniformly from this set, and its start and goal roles are randomly exchanged. If sampling or subsequent planning fails, the collector resamples the task. The path-length distribution this produces is measured on the collected episodes (Sec. 3.5).

Initial pose. The robot and goal share an episode-specific initial height drawn from the scene configuration. Planning operates in the horizontal plane, and the action interface supplies horizontal body-frame velocity and yaw rate, with zero commanded body-frame vertical velocity. The robot is initialized at rest. In the default goal-facing initialization, its yaw points toward the goal with a uniformly sampled offset in $[-45°, 45°]$. This produces variation in the initial heading while keeping the task endpoints fixed.

Virtual obstacle placement. Beyond the geometry recovered from the scene itself, episodes may include virtual box obstacles placed in the free space of the planning grid. The number of virtual obstacles is fixed within a collection run. Given the sampled start and goal, their centers are drawn by rejection sampling over free grid cells: a candidate is accepted when its box footprint, expanded by a planning clearance, contains no occupied cells, and when it lies at least a configured distance from the start, from the goal, and from every previously accepted virtual obstacle. Sampling stops once the requested number of centers is accepted; an episode whose virtual obstacles cannot be placed within the attempt budget is rejected and resampled. Each accepted obstacle is assigned a random yaw, and its footprint, expanded by a planning clearance, is written into the planning occupancy grid before path search. Episode metadata records the number of placed obstacles and their positions; in the released corpus, 65,000 of the 94,000 episodes (69.1%) contain virtual obstacles.

A.3 Reference Path Planning

A* searches an eight-connected occupancy grid. Horizontal and vertical moves have unit cost, diagonal moves have cost $\sqrt{2}$, and the heuristic is Euclidean distance in grid coordinates. A diagonal transition is allowed only when both adjacent axial cells are free, preventing corner cutting during grid search. Endpoints are associated with nearby free cells within a bounded snapping radius, and the search runs under a bounded expansion budget; a failed endpoint association, an exhausted budget, or an unsuccessful search causes the task to be rejected.

After search, a greedy shortcutting pass reduces the number of intermediate waypoints. Starting from the current anchor, it selects the farthest subsequent waypoint whose connecting segment passes the grid's discrete free-space check, and repeats this operation until the goal is reached. Grid points are converted back to scene coordinates, and the path endpoints are set to the sampled start and goal. For the resulting polyline $P = (\mathbf{p}_0, \ldots, \mathbf{p}_K)$, the reference length is

$$L(P) = \sum_{k=0}^{K-1} \|\mathbf{p}_{k+1} - \mathbf{p}_k\|_2. \tag{2}$$

The collector stores both the waypoint sequence and its length. Across the released corpus, the mean ratio of executed path length to straight-line start–goal distance is 1.06.

A.4 Closed-Loop Expert Execution

Waypoint tracking. The expert maintains an index into the reference polyline. At each control step, it advances past intermediate waypoints within a capture radius r_{wp} of the current horizontal position. The active waypoint is lifted to the robot's current height and transformed into the body frame using the current pose. Let $\mathbf{e}_t = (e_{x,t}, e_{y,t})$ denote its horizontal body-frame displacement and $d_{g,t}$ the horizontal distance to the final goal. The base controller uses the goal-dependent speed fraction

$$b_t = \min\left(1, \max\left(0.25, \frac{d_{g,t}}{r_{\mathrm{slow}}}\right)\right) \tag{3}$$

and produces normalized translational commands

$$\mathbf{u}_t = b_t \frac{\mathbf{e}_t}{\|\mathbf{e}_t\|_2}, \qquad \omega_t = \mathrm{clip}\left(k_\psi \,\mathrm{wrap}\left(\mathrm{atan2}(e_{y,t}, e_{x,t})\right), -\omega_{\max}, \omega_{\max}\right). \tag{4}$$

Here, wrap maps angles to $[-\pi, \pi)$, and a near-zero waypoint displacement produces a zero action. Translational components are clipped to the action bounds. The controller recomputes these commands from the current simulated pose at every step; the reference path remains fixed during a base rollout.

Action semantics. The stored action is $\mathbf{a}_t = (u_{x,t}, u_{y,t}, \omega_t)$. The wrapper inserts a zero vertical component and passes the command to the simulator. With configured translational scale v_{scale}, the simulator applies

$$\mathbf{v}_t^b = v_{\mathrm{scale}} \frac{(u_{x,t}, u_{y,t}, 0)}{\max(1, \|(u_{x,t}, u_{y,t}, 0)\|_2)}. \tag{5}$$

Yaw rate is passed directly as the body-frame angular velocity about the vertical axis. The simulator rotates the linear and angular velocities into the world frame and writes them to the robot state. Thus, the first two stored action components require the configured velocity scale to recover physical units. This is a velocity-controlled simulation interface; the demonstrations describe navigation under that interface.

Episode acceptance. An episode is accepted when the environment reports success or the robot enters the collector's horizontal goal-radius threshold. Episodes that terminate without success are excluded from the standard demonstration output, and the task is resampled. Collection continues until the requested number of successful demonstrations has been saved.

Discarded rollouts are recorded in the collection logs. The most recent collection run and its resumption together saved 65,000 successful episodes, of which 64,000 enter the released corpus; 1,332 rollouts terminated without reaching the goal and two aborted on a simulator error, so 98.0% of the rollouts flown in that run were kept. These counts cover rollouts only; tasks rejected at sampling or planning time are resampled before any rollout and are not logged as failures.

A.5 Recorded Modalities and Episode Storage

The collector records the current observation and selected action before advancing the simulator. A record contains forward RGB, a 13-dimensional state vector, a local occupancy patch, and a three-dimensional action. RGB is stored as an unsigned 8-bit array in height–width–channel order. State and action arrays use 32-bit floating-point values.

The state vector concatenates body-frame linear velocity (three entries), body-frame angular velocity (three), the goal position relative to the robot in the body frame (three), and the world-frame orientation quaternion in (w, x, y, z) order (four). The relative goal is therefore embedded in the state array. The local occupancy patch is centered at the robot and aligned with its yaw, with forward along decreasing row indices and body-left along increasing column indices. Occupied cells have value one, free cells have value zero, and samples outside the grid are marked occupied. Its cell size is recorded per episode; at the reference resolution of $0.10\,\mathrm{m}$ the 64×64 patch covers $6.4 \times 6.4\,\mathrm{m}$. It is computed from scene geometry and constitutes geometric supervision; its presence in the corpus does not prescribe which inputs a benchmark policy receives.

Variable-length episodes are stored individually or concatenated along the time dimension in compressed NPZ shards. Shards include episode identifiers, start offsets, lengths, and success flags, so

Table 4: Reference settings of the collection implementation. Scene files and launch arguments must be consulted when reproducing a particular collection run.

Parameter	Reference value
Navigable space and task sampling	
Planning-grid cell width δ	$0.10\,\mathrm{m}$
Endpoint candidates N	128
Minimum endpoint separation $d_{\min}$	$2.0\,\mathrm{m}$
Additional endpoint-clearance margin	$0.40\,\mathrm{m}$
Endpoint distance-quantile filter	0.0
Task resampling attempts	20
Virtual obstacles	
Box footprint	$0.8 \times 0.8\,\mathrm{m}$
Planning clearance	$0.20\,\mathrm{m}$
Clearance from start	$0.80\,\mathrm{m}$
Clearance from goal	$1.00\,\mathrm{m}$
Minimum spacing	$1.20\,\mathrm{m}$
Placement attempts	2048
Planning and control	
Planner snapping radius	$0.80\,\mathrm{m}$
Waypoint capture radius r_{wp}	$0.18\,\mathrm{m}$
Goal slowdown radius r_{slow}	$0.80\,\mathrm{m}$
Collector goal radius	$0.45\,\mathrm{m}$
Yaw proportional gain k_ψ	$2.5\,\mathrm{s}^{-1}$
Yaw-rate bound $\omega_{\max}$	$1.0\,\mathrm{rad/s}$
Velocity scale v_{scale}	$2.0\,\mathrm{m/s}$
Simulation step / control interval	$0.01\,\mathrm{s}$ / $0.02\,\mathrm{s}$
Recording	
RGB resolution	256×256
Local occupancy shape	64×64
Local occupancy resolution	$0.10\,\mathrm{m}$

episode boundaries can be recovered exactly. Companion JSON metadata records task endpoints, initial pose information, planning configuration, reference waypoints, path length, episode length, and available camera and collection settings. Optional diagnostic videos are stored separately from the training arrays.

Integrity checks. The released corpus is reconciled against its own index in both directions. Every one of the 94,000 episodes recorded in metadata resolves to a shard file that is present, non-empty and opens as a valid archive; every one of the 940 shard files is referenced by that metadata; and no missing or orphaned files remain on either side. Shard metadata reports the expected count of successful episodes throughout, and the frame totals reported above are summed from the per-episode lengths it records. Beyond this reconciliation, the corpus has been consumed end to end: the seven baselines of Sec. 4 read these shards directly for training and offline evaluation, so the recorded arrays load, stay aligned across modalities, and support policy learning.

A.6 Behavior Diversity Mechanisms

Beyond the base episodes, the collector provides mechanisms that extend task and behavior coverage. All of them record into the same observation–action schema as base episodes.

Obstacle- and corner-aware speed. An optional multiplier reduces speed according to nearby occupied cells and upcoming path bends. Clearance is measured from the center of the local occupancy patch to the closest occupied cell in a forward-facing region that includes a small rear margin. A clipped clearance ratio determines one speed factor. A second factor decreases as the robot approaches a waypoint whose turning angle reaches a specified threshold. The controller uses the smaller factor, bounded below by the configured minimum speed fraction, together with the base goal slowdown.

Near-obstacle starts. The hard-start sampler selects free cells from a configurable clearance band around obstacles. It samples a separate feasible goal and varies the initial heading using uniform, obstacle-facing, or goal-facing modes. The sampled start must remain inside the scene polygon, and the resulting task must pass the same planning and execution checks as a base episode.

Perturbation and recovery. During selected episodes, the collector temporarily holds an earlier action or adds a bounded horizontal command bias. After the disturbance ends, it replans from the current robot position to the original goal and resumes waypoint tracking. The action array stores the command actually selected during the disturbance, while event metadata additionally records the contemporaneous expert commands, event timing, and replanning outcome. These fields distinguish disturbance segments from subsequent expert recovery when constructing training samples.

Planning and camera variation. An episode-level inflation offset changes the planning margin, while the local occupancy observation continues to use the base geometry representation. This keeps observation geometry consistent across planning-margin draws. Camera variation changes horizontal field of view at episode boundaries while retaining the square RGB resolution. The collector records the drawn settings and available runtime camera metadata for later analysis.

A.7 Collection across Scenes

Scene collection processes operate independently, each loading its own scene assets, sampling tasks, executing the expert, and writing episode shards. An orchestration layer runs one process per scene and packs several processes onto each GPU, so that a single launch fills every available GPU up to a configured per-GPU process count and queues the remaining scenes. Collection writes to fast local storage and promotes a scene to durable network storage only once its shard metadata reports the requested episode count and every archive opens as a valid file, which keeps the step loop off the network-storage path. A reference count derived from already-collected scenes sets the per-scene target, a resumable staging layout supports continuation after interruptions, and a watchdog frees the GPU slot of a collector that reports a large, configurable number of attempts without a success. Failed task attempts are resampled within a scene, and process-level failures do not halt the remaining queue. The common recording schema permits outputs from independently collected scenes to be aggregated for policy training.

B Experimental Details

B.1 Evaluation Protocol and Metrics

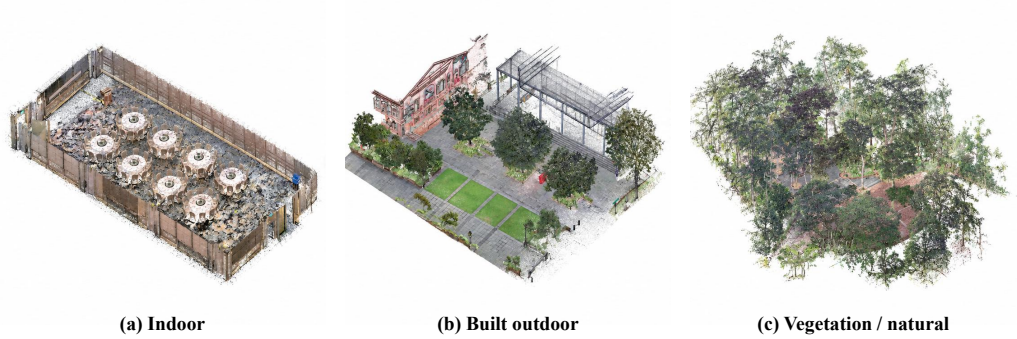
Data split. All policy training follows the dataset partitions in Table 5, which reports scene counts, stored trajectories, and scene polygon areas. Each partition covers indoor, urban outdoor, and vegetation environments. Each training scene provides 900 trajectories for training and 100 held-out trajectories for seen offline evaluation. Validation scenes are used for checkpoint selection; test scenes are excluded from training and model selection.

Figure 10 illustrates reconstructed scenes from the three categories.

Table 5: **Dataset splits.**

	Scenes				Flight data	Area (m^2)
Split	Indoor	Urban outdoor	Vegetation	Total	Trajectories	Min–max
Training	11	38	21	70	70,000	23–6,674
Validation	3	3	2	8	8,000	14–1,640
Unseen test	4	7	5	16	16,000	118–4,543
Total	18	48	28	94	94,000	14–6,674

Evaluation episodes. The full closed-loop evaluation protocol specifies 16 seen scenes from the training partition and all 16 unseen test scenes, with 50 fixed episodes per scene (800 per split;



(a) Indoor (b) Built outdoor (c) Vegetation / natural

Figure 10: **Reconstructed scenes across three environment categories.** Indoor, urban outdoor, and vegetation examples.

1,600 per policy and navigation setting). The seen subset covers the three environment categories and a range of spatial structures and areas (Table 6). Exact scene IDs, episode configurations, and sampling seeds will accompany the dataset release. Episode sampling follows the same scene-adaptive procedure used for data collection (Sec. 3.3). The protocol fixes scene IDs, initial poses, goals, geometry, and task seeds across policies and navigation settings; goal conditioning changes between settings. Seen evaluation tasks are held out from training demonstrations. Evaluation uses the scenes' existing static obstacles, without adding synthetic obstacles.

Table 6: **Evaluation scenes.**

		Seen		Unseen	
Environment	Spatial structures	Scenes	Area range (m^2)	Scenes	Area range (m^2)
Indoor	Rooms, corridors, open halls	4	80–1,079	4	118–863
Urban outdoor	Covered passages, streets, plazas	7	200–6,674	7	502–4,543
Vegetation	Wooded paths, hillside gardens, coastal parks	5	141–1,191	5	128–1,678

Scaling evaluation. The scaling protocol retains the five seen scenes shared by the 5-, 15-, 35-, and 70-scene training subsets and uses five fixed unseen test scenes. Each scene contributes 100 fixed episodes: 500 seen and 500 unseen per training seed. The same tasks are reused across training scales and the three seeds. Metrics are averaged equally over scenes within each split for each seed. The figure reports individual seed results; the main-text endpoint values refer to the median across the three seeds. Scaling therefore uses a different scene subset and episode count from the main benchmark.

Termination. An episode succeeds when the drone enters the goal's 0.45 m horizontal radius without a safety violation; no stop action is required. Failure occurs on collision, leaving the navigable polygon, or reaching the 5,000-step limit (100 s). Collision is detected when obstacle distance in the scene-aligned 3D distance field is at most the 0.12 m body radius (Sec. 3.2).

Offline metrics. We select 20 held-out trajectories of at least 152 steps per scene by a fixed seeded hash ordering (seed 20260910). All policies share up to six distinct expert anchors per trajectory: steps 0 and 50 (0 and 1 s), and the rounded 10/35/65/90% positions within $[101, T-51]$, where T is the trajectory length in 50 Hz steps. Duplicate indices after rounding are removed, and all policies use the same valid anchors. This retains at least 1 s of future data at each anchor. Predicted and expert trajectories share the initial pose and are compared on a $\Delta t = 0.02$ s grid. For $K = h/\Delta t$

future positions $\hat{\mathbf{p}}_k, \mathbf{p}_k \in \mathbb{R}^2$,

$$\mathrm{ADE}_h = \frac{1}{K}\sum_{k=1}^{K} \|\hat{\mathbf{p}}_k - \mathbf{p}_k\|_2, \qquad \mathrm{FDE}_h = \|\hat{\mathbf{p}}_K - \mathbf{p}_K\|_2. \tag{6}$$

We use $h = 0.2\,\mathrm{s}$. Control predictions are integrated into kinematic trajectory proxies; waypoint positions are linearly interpolated between native timestamps onto the evaluation grid. Additional queries within the window use recorded expert observations rather than observations from predicted states. CNN-BC is queried at every expert state in the window. For OmniVLA and NavMorph, the 0.2 s window joins eight controls from the initial query with two from a new expert-state query at step 8; missing query coverage is unavailable. These are teacher-forced offline predictions. Table 7 defines additional diagnostics over the same 0.2 s window, with subsets determined by expert behavior. Position errors are computed in meters and converted to centimeters for display in Table 2; angles are in radians, and lower errors are better.

Table 7: **Offline diagnostics.**

Metric	Definition
Motion Direction Error	Angle between predicted and expert net displacements; undefined for zero displacement.
Band-limited DTW	Minimum cumulative position distance over monotone alignments, divided by K; band $\lvert i-j\rvert \le \max(1, \lceil 0.1K\rceil)$.
Lateral / Yaw Direction Error Rate	Opposite-sign fraction for expert lateral displacement $\lvert d_y\rvert \ge 0.02\,\mathrm{m}$ / yaw change $\lvert\Delta\psi\rvert \ge 0.02\,\mathrm{rad}$. Lateral displacement uses the initial body frame.
Yaw Error	Absolute wrapped difference between predicted and expert yaw changes at the 0.2 s endpoint, averaged over valid anchors within trajectories, then equally across trajectories within scenes and across scenes.
P95 FDE	95th percentile of anchor-level FDE within each scene, then averaged equally across scenes.
Corner FDE	FDE for a $\ge 20^\circ$ angle between the first and last 0.1 s displacements of the next second; both lengths must exceed 0.02 m.
Turning-segment FDE	FDE with 0.2 s expert history and 1 s future, where $\lvert\omega\rvert \ge 0.1$ rad/s in either adjacent 0.2 s window.
Turn-onset / Turn-end Yaw Error	With 0.2 s expert history and 1 s future, yaw error at $\lvert\bar{\omega}^-\rvert < 0.1 \le \lvert\bar{\omega}^+\rvert$ / $\lvert\bar{\omega}^+\rvert < 0.1 \le \lvert\bar{\omega}^-\rvert$, where $\bar{\omega}^-, \bar{\omega}^+$ are mean expert yaw rates (rad/s) in the preceding/following 0.2 s windows.
Start / Mid-trajectory FDE	FDE at $t < 2$ s / $t \ge 2$ s; mid includes all post-start states.
Maneuver-balanced FDE	Equal-weight mean of FDE across three expert-demand bins.

Demand bins use the larger of the bins for mean absolute lateral speed and yaw rate over the next second. Cutoffs are the 50th and 90th percentiles of 4,000 training windows from four scenes: approximately $(5.858712\times10^{-5}, 0.03238329)$ m/s and $(1.464765\times10^{-4}, 0.08099765)$ rad/s, respectively. Values at a cutoff enter the lower bin. Each scene-level mean requires 5 valid samples, P95 requires 20, and balanced FDE requires 5 per bin; otherwise the value is missing.

Offline means are computed over valid anchors within each trajectory, then averaged equally across trajectories within each scene; split summaries weight scenes equally. Cross-policy correlations use one sample per policy after averaging over common scenes, so n counts valid policies. Within-policy correlations instead use one sample per scene. Spearman's ρ is the Pearson correlation of ranks, averaging tied ranks; constant inputs yield an undefined correlation. For offline error e and deployment metric y, direction-aligned correlation is

$$\rho_{\mathrm{align}}(e,y) = \begin{cases} -\rho(e,y), & y \in \{\mathrm{SR}, \mathrm{A^*\text{-}SPL}, \mathrm{OSR}\}, \\ \rho(e,y), & y \in \{\mathrm{NE}, \mathrm{CR}, \mathrm{OB}, \mathrm{Timeout}, \mathrm{TTS}\}. \end{cases} \tag{7}$$

Positive values indicate that larger offline errors accompany worse deployment outcomes; negative values indicate the reverse. Paired differences subtract the reference diagnostic's aligned correlation using identical policies, scene coverage, and deployment outcomes. Missing entries are excluded jointly from both correlations.

Execution characteristics. ControlVar is the mean Euclidean change between consecutive normalized control vectors after command limiting. Turn/m is the accumulated absolute change in planar path direction per meter traveled, measured in rad/m.

Closed-loop metrics. For N completed episodes, SR, CR, OB, and timeout are the percentages ending in each corresponding outcome. NE is mean terminal 3D goal distance; OSR is the percentage whose trajectory ever enters the goal's 1 m 3D radius. With success indicator S_i, executed horizontal path length P_i, reference length L_i of the A* path after line-of-sight shortcutting, and control-step count T_i,

$$\text{A}^*\text{-SPL} = \frac{1}{N}\sum_{i=1}^{N} S_i \frac{L_i}{\max(L_i, P_i)}, \qquad \text{TTS} = \frac{\Delta t \sum_i S_i T_i}{\sum_i S_i}. \tag{8}$$

A*-SPL uses $[0, 1]$; TTS averages simulation seconds over successes and is missing if none succeed. The reference ends at the goal point, whereas execution stops within its success radius. Thus $P_i \leq L_i$ can saturate the score at 1; A*-SPL measures efficiency relative to the grid-based planner, not an exact continuous-space optimum. Under the full protocol, closed-loop metrics other than TTS are computed within each scene and then averaged equally across scenes within each split. With 50 episodes in every scene, this equals pooling episodes for SR, CR, OB, timeout, NE, and A*-SPL. TTS remains pooled over successful episodes.

The matched perturbation protocol and its local safety metrics are defined in Appendix B.4.4.

B.2 Baseline Configurations

Shared interface. AirPlayground shares data, controls, and evaluation while preserving native policy interfaces. In $\tau = \{(o_t, s_t, g_t, a_t)\}_{t=1}^{T}$, o_t, s_t, g_t, a_t denote RGB, flight state, goal, and recorded control (u_x, u_y, ω). The conversion from stored translational controls to physical velocity follows Appendix A and the collection velocity scale; yaw is already in rad/s. Policy-native target normalization is separate from this storage convention. The training adapter constructs native inputs and targets: $(\mathbf{x}_t^m, \mathbf{y}_t^m) = \mathcal{T}_m(\tau, t)$.

At query q, $\mathcal{O}_m$ builds $\mathbf{x}_q^m$ from causal RGB/state/action histories, the goal, and policy memory, yielding $\hat{\mathbf{y}}_q^m = \pi_m(\mathbf{x}_q^m)$. Within the controller $\mathcal{C}_m$ of Sec. 3.4, decoder $\mathcal{D}_m$ produces a control sequence or metric plan and executor $\mathcal{E}_m$ applies it:

$$\mathbf{p}_q^m = \mathcal{D}_m(\hat{\mathbf{y}}_q^m), \qquad \mathbf{u}_t = \mathcal{E}_m(\mathbf{p}_q^m, s_t; \mathbf{h}_t^m, H_m), \quad q \leq t < q + H_m.$$

Context $\mathbf{h}_t^m$ stores plans and policy memory. Execution uses state feedback where required and queries every H_m steps. Horizon experiments fix weights, decoding, and history sampling while varying H_m. Query-triggered state updates, including NavMorph's CEM memory update, follow the resulting schedule.

Training. The API supports PointGoal, ImageGoal, and Goal-free conditioning where applicable; each adapter declares its supported modes. The main benchmark uses PointGoal: observations, state, and relative goals, without maps, A* paths, or simulator goal RGB. Evaluation follows Appendix B.1; pretraining, optimization, and inference compute are not equalized. CNN-BC, ACT, and DP initialize visual encoders from ImageNet; other methods use released weights. Tables 8–9 summarize adapters and source training configurations for the evaluated checkpoints.

Execution configuration. Execution settings use the official implementations as references where applicable, with adaptations to the shared flight-control interface. We explore nearby execution intervals around these adapted configurations on validation scenes and freeze the selected settings for evaluation. The horizons in Table 8 describe our deployed adapters rather than universal defaults of the original methods. The execution-horizon study varies these intervals separately with model weights fixed.

RGB history denotes frame count/stride, including the current frame. Stride and execution horizon H count 0.02 s steps. Prediction span includes the final control interval for action outputs. Controls are clipped to ± 2 m/s per planar axis and ± 1 rad/s in yaw.

Table 8: **Policy adapters.**

Method	RGB history	Native output	Pred. span (s)	Exec. H	Deployment adapter
CNN-BC	1/1	1 control	0.02	1	Direct / hold
ACT	1/1	50 controls	1.00	50	Prefix
DP	2/1	32 controls	0.64	16	Prefix
ViNT	6/5	5 waypoints	0.50	10	Tracker
OmniVLA	1/1	8 poses	0.16	8	Pose differences / prefix
NWM	4/5	Future view	0.80	40	Rank / hold
NavMorph	30/1	8 poses	0.16	8	Pose differences / prefix

Table 9: **Training configurations.**

Method	Updates	Batch	Peak LR	Training scope
CNN-BC	5,280	256	$(1–3)\times10^{-4}$	Full policy
ACT	34,650	32	10^{-5}	Full policy
DP	54,450	32	10^{-4}	Full policy
ViNT	8,418	256	10^{-6}–10^{-4}	Goal modules + decoder
OmniVLA	500	32	$(2–5)\times10^{-5}$	LoRA + adapters
NWM	604	64	10^{-5}	Full predictor
NavMorph	10,000	4	10^{-5}	RGB backbone + adapters

Updates refer to the selected checkpoint's final training stage. Batch includes all GPUs and accumulation (NWM: contexts, four targets each). LR ranges span parameter groups; exact assignments and stage configurations accompany the checkpoints.

All methods use AdamW; `configs/baselines/` specifies schedules, weight decay, group LRs, and frozen modules. Earlier selected stages comprise 2,806 ViNT goal-adapter updates, 504 OmniVLA head/projector updates, and 302 NWM conditioning updates.

Direct-control policies. CNN-BC holds its predicted control until refresh. ACT and Diffusion Policy (DP) execute H-step prefixes and discard unused tails. DP uses 100 denoising steps per decision; ACT disables temporal ensembling by default.

Waypoint policies. Our ViNT adaptation adds learned point-goal conditioning and predicts five query-relative $(x, y, \cos\psi, \sin\psi)$ waypoints at 0.1 s spacing and 0.2 m XY scale. Its cached third waypoint defines a world-frame position/heading target; each control step divides body-frame position and wrapped heading errors by 0.3 s. OmniVLA adds a pose-goal projector, LoRA, and eight-pose head. Both leave visual-goal slots blank in PointGoal mode.

NavMorph adds a goal projector and eight-pose head to its RGB backbone, uses aligned state/action histories, and resets CEM to its initial memory each episode. OmniVLA and NavMorph decode cumulative query-relative poses (0.1 m XY scale) into control prefixes: difference consecutive poses, starting from the query-frame origin; rotate displacements into the preceding body frame; divide displacement and wrapped yaw changes by 0.02 s and clip.

World-model policy. Our NWM PointGoal adapter converts body-frame goal g into a terminal relative pose $(g_{\text{local}}, \theta)$, where $g_{\text{local}} = g\min(1, 1.6/\|g\|)$ and $\theta = \operatorname{atan2}(g_y, g_x)$. XY follows NWM's 0.17 m normalization; yaw remains in radians. The temporal condition follows NWM's native normalization and corresponds to a 0.8 s prediction horizon. Using observed RGB history, 10 DDIM steps generate this view. This view is model-generated, never rendered or retrieved from the simulator at the target location.

The 120 candidates comprise the mean control and 119 diagonal-Gaussian samples, clipped to $[-1, 1]^3$. Mean and standard deviation are fixed from training-control statistics. Sampling uses a deterministic seed derived from the episode identifier and query index. NWM predicts 0.8 s end-

point views with 10 DDIM steps and shared noise across candidates. LPIPS ranks them against the generated goal view; the best five controls are averaged and held for 40 steps.

New policies implement these adapters while reusing the simulator and evaluator.

B.3 Additional Navigation Results

Goal specifications. AirPlayground's unified API supports baseline training and evaluation with PointGoal, ImageGoal, and Goal-free conditioning. PointGoal supplies a relative target. Here, ImageGoal uses successive A* subgoal views, denoted *Image Subgoal*; the API also supports a single terminal goal image. Goal-free withholds goal input and relative-goal state entries, retaining the target only for evaluation. Table 10 reports navigation under these guidance settings.

Image subgoals are rendered at the episode's flight height along the A* route, sampled every 0.5 m starting 0.5 m from the start. For route length L, the final view is placed at arc length $\max(L/2, L - 1\,\mathrm{m})$. Each view faces the next subgoal, and the final view faces the task goal. Before each policy step, the subgoal index advances monotonically while the current subgoal is within 3 m Euclidean distance of the drone, stopping at the first farther subgoal or the final view. The selected image is supplied through the goal-image interface. This provides route guidance without requiring arrival at every subgoal.

Figure 11 compares example paths under the three goal settings.

Shared evaluation tasks. The full three-setting protocol uses the same 16 seen and 16 unseen scenes and 50 start–goal episodes per scene for each supported policy–setting pair (800 per split; 1,600 total). PointGoal, ImageGoal, and Goal-free reuse initial poses, target locations, and scene geometry while changing goal conditioning. Except for TTS, metrics are first averaged within scenes, then equally across scenes within each split; a combined score gives seen and unseen equal weight. TTS pools successful episodes. Table 10 reports navigation scores.

Task completion and safety. These results demonstrate the API's support for learning and executing policies under different forms of goal guidance. Image Subgoal yields higher SR than PointGoal for DP, ViNT, and OmniVLA, but lower SR for the other four methods. Successive visual subgoals provide intermediate route guidance, yet their benefits depend on the policy. Withholding goal input lowers SR for all seven methods. Safety still requires examining multiple failure modes: Goal-free CNN-BC has 19.8% CR but 73.3% OB, showing that low collision rate alone does not imply reliable navigation.

B.4 Offline-to-Closed-Loop Analysis

B.4.1 Offline Diagnostics

Main-text aggregation. The main-text cross-policy analysis uses seven baselines. For each policy and metric shown in the main-text correlation figure, we first average over common valid scenes within each split, then take the arithmetic mean of the seen and unseen split scores and compute Spearman correlations across the seven resulting policy scores; we do not average split-specific correlations. The full protocol specifies 16 seen and 16 unseen scenes with 50 closed-loop episodes per scene. The sample unit is a policy, giving seven observations for cross-policy correlations. Direction alignment follows Appendix B.1. For collision-rate comparisons, the signed difference is $\Delta\rho = \rho(e, \mathrm{CR}) - \rho(\mathrm{FDE}, \mathrm{CR})$, evaluated on identical valid policies and scene coverage. This is a descriptive cross-policy association, not an estimate of predictive accuracy. The evaluation protocol is specified in Appendix B.1.

B.4.2 Execution Horizon

Main-text evaluation protocol. The main-text execution-horizon study uses five seen and five unseen scenes with 50 fixed episodes per scene, totaling 500 episodes per policy and horizon. All horizons reuse the same tasks, checkpoint, prediction length, decoder, and history sampling for each policy. Scores are averaged within scenes and then equally within each split; combined scores weight seen and unseen equally. Each control step corresponds to 0.02 s.

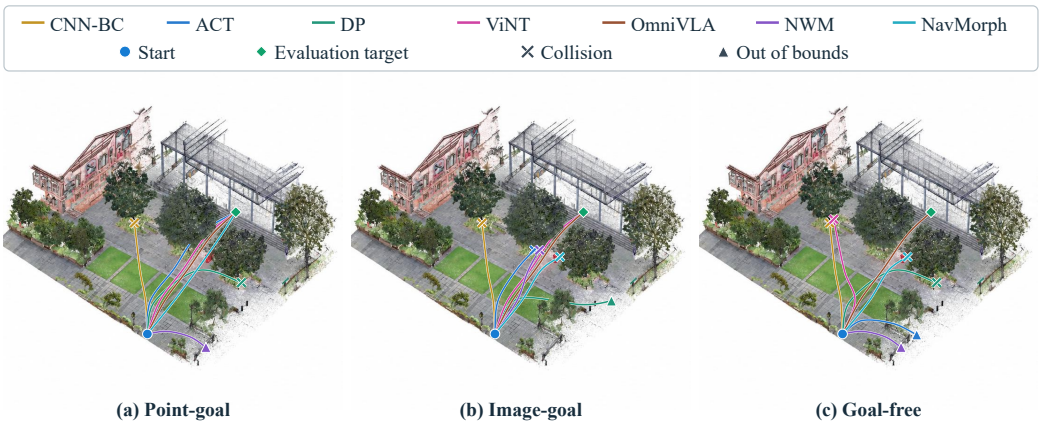


Figure 11: **Navigation under different goal guidance.** Paths share a start and evaluation target; the target is withheld from Goal-free policies.

Table 10: **Closed-loop navigation across goal modes.** The full protocol reuses 50 tasks per scene across settings in 16 seen and 16 unseen scenes; see Appendix B.3 for aggregation and coverage.

Method	PointGoal							ImageGoal							Goal-free						
	NE↓	OSR↑	SR↑	SPL↑	CR↓	OB↓	TTS↓	NE↓	OSR↑	SR↑	SPL↑	CR↓	OB↓	TTS↓	NE↓	OSR↑	SR↑	SPL↑	CR↓	OB↓	TTS↓
CNN-BC	6.44	20.8	18.0	0.18	48.2	33.8	7.21	5.60	10.2	8.3	0.07	59.2	32.5	5.49	5.84	10.5	6.9	0.06	**19.8**	73.3	5.46
ACT	4.38	39.1	35.1	0.35	52.1	12.7	4.74	4.08	26.7	24.8	0.22	54.3	20.9	3.21	4.34	16.4	12.1	0.10	63.5	**24.4**	2.42
DP	7.17	26.5	15.9	0.14	57.1	27.0	5.88	4.13	24.4	20.2	0.16	52.7	27.1	3.58	4.43	16.4	13.1	0.11	60.1	26.8	3.75
ViNT	4.94	39.0	35.9	0.36	50.4	13.7	8.05	2.20	52.1	51.1	**0.48**	**39.7**	9.2	6.01	4.36	17.1	13.4	0.11	60.8	25.8	3.45
OmniVLA	3.81	46.5	44.0	0.43	49.1	**6.9**	6.83	**2.00**	**53.8**	**52.3**	0.47	42.8	4.9	5.44	4.08	**32.6**	**19.6**	**0.15**	49.4	31.0	3.74
NWM	**2.94**	**50.0**	**44.3**	**0.44**	**40.9**	14.8	**3.92**	2.60	44.2	36.4	0.33	46.8	16.8	3.92	4.73	19.1	11.4	0.09	57.5	31.1	2.75
NavMorph	5.17	36.5	29.0	0.28	52.9	18.1	5.19	2.57	30.1	28.8	0.25	71.2	**0.0**	**2.57**	**4.01**	19.6	15.1	0.13	57.2	27.7	**2.29**

Supplementary two-scene diagnostic sweeps. We vary H on ten fixed tasks per scene in an indoor laboratory and a hill garden, retaining each checkpoint, prediction length, decoder, and history sampling (Fig. 12). Each step is 0.02 s; execution follows Appendix B.2. ViNT retains its cached tracking target beyond the prediction span when H is longer. Evaluation processes use seed 20260914, with one run per task and interval. ACT improves beyond $H = 1$, NavMorph favors intermediate intervals, and ViNT retains the same SR and CR through $H = 50$. These matched-task results show policy-dependent responses over the tested intervals.

Turning behavior. A separate six-task ACT diagnostic compares $H = 50$ and $H = 1$ from matched starts. For each run, let $\bar{\omega}$ be the mean executed yaw-rate command over the first ten steps. We retain $|\bar{\omega}| \geq 0.1$ rad/s and define decay time as the start of the first subsequent five-step interval satisfying $\text{sign}(\bar{\omega})\omega_t \leq |\bar{\omega}|/2$ throughout. Four tasks have measurable decay in both runs; one lacks an initial turn and one terminates early. All four decay times increase under $H = 1$, including 0.40 to 1.28 s in the displayed task (Fig. 13). This is a conditional maneuver diagnostic: one task still succeeds despite delayed decay, and early-terminated runs have no measured decay time.

B.4.3 Clearance-Based Diagnostics

Table 3 summarizes 180 independent local rollouts across seven methods and ten scenes, with one rollout per sample. Starting states are selected near obstacles along expert trajectories, without conditioning on policy failures. At each start, the reference obstacle is selected in advance as the obstacle nearest to the expert's future path and remains fixed throughout the rollout, regardless of predictions or collision outcomes. Clearance is the distance from the drone center to the obstacle surface minus the body radius: positive values indicate remaining safety margin, and negative values indicate intrusion into the safety boundary. Expert and predicted minimum clearances refer to the same reference obstacle over a 0.2 s prediction window (ten control steps at 50 Hz); their difference is $\Delta c = c_{\min}^{\text{pred}} - c_{\min}^{\text{expert}}$. The decrease, stable, and increase groups use $\Delta c < -0.02$ m, $|\Delta c| \leq$

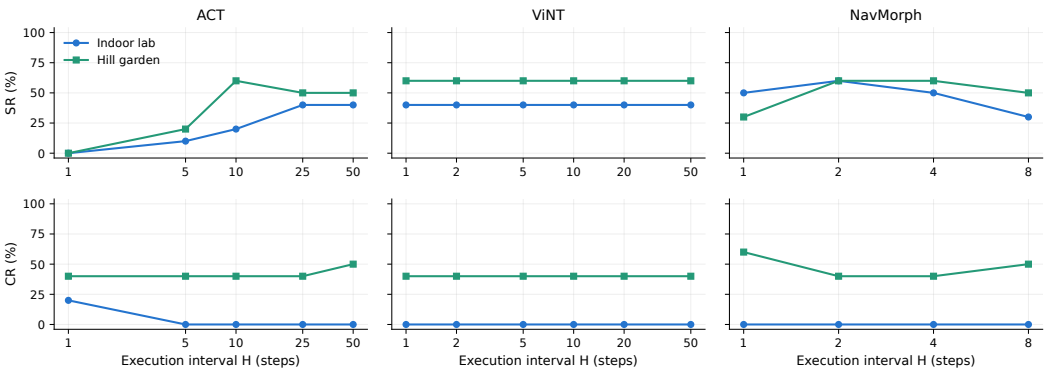


Figure 12: **Execution-interval sweeps.** Supplementary two-scene evaluation, separate from the main-text sweep. Every point uses the same ten tasks in its scene; each method retains its checkpoint across intervals. The two rows show SR and CR. The horizontal axis lists all evaluated intervals on a logarithmic scale.

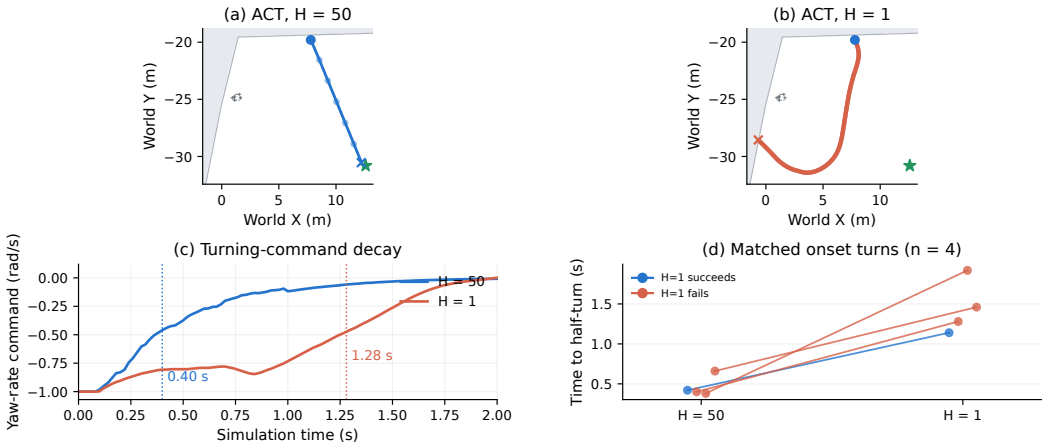


Figure 13: **How execution changes a maneuver.** A separate matched-task ACT diagnostic. (a–b) Matched-start ACT trajectories on a flight-height occupancy slice: dark gray denotes obstacles and light gray lies outside the navigable polygon. Dots mark queries, stars goals, and crosses episode ends. (c) Executed yaw commands. (d) All four measurable turn-decay pairs, colored by $H = 1$ outcome. States evolve independently after the shared start.

$0.02\,\mathrm{m}$, and $\Delta c > 0.02\,\mathrm{m}$, yielding 43, 76, and 61 samples, respectively. FDE is the endpoint position error over that window. Distance columns report group means, with the closed-loop column averaging each rollout's local minimum clearance. Local passage requires reaching a predefined exit within 5 s without collision or leaving the navigable region. The exit lies 0.5 m beyond the reference obstacle's far boundary along the expert's direction of travel. Collision counts concern the reference obstacle; passage requires avoiding all collisions. Both counts use the full group size N as their denominator, including other termination outcomes.

B.4.4 LOCAL GEOMETRY

Local intervention. ACT ($H = 50$) and NavMorph ($H = 8$) use nominal rollouts in an indoor laboratory and a courtyard. We select ten anchors per policy–scene–clearance group from collision-free states at step 30 or later, requiring speed 0.2–2.05 m/s, goal distance at least 1.5 m, and boundary clearance above 0.4 m. Directions must pass a local SDF probe: moving ± 0.2 m toward/away changes distance by at least 0.04 m in the expected direction. Shared clearance cutoffs use quantiles of the four equally weighted candidate-frame pools. Branches share the recorded observation prefix and use live observations after intervention. We apply a world-frame planar velocity disturbance of 0.5 m/s for up to 0.4 s, stopping on episode termination. Toward uses the normalized XY vector $n = (n_x, n_y)$ from the anchor to its nearest occupied-voxel center; Away uses $-n$, and Tangential uses $\sigma(-n_y, n_x)$, with $\sigma \in \{-1, 1\}$ fixed in the anchor manifest. Directions remain fixed during the pulse; the added increment is removed afterward. The disturbance is added after native velocity limiting without reclipping; it models an external disturbance, not a bounded policy output. Initial clearance subtracts the 0.12 m body radius; shared bin boundaries are 0.520312 and 0.780000 m. Directions are controlled within anchors; clearance bins compare different states.

Reference region. The reference patch comprises occupied voxels within 2 m geodesic distance of the initially nearest occupied voxel, using edge-weighted 26-neighbor connectivity. All branches and strengths share an anchor-centered 1 m-radius observation sphere. Observation ends at first exit, termination, or 5.4 s; later events are excluded. Of 120 anchors, 93 have valid patches; 24 have isolated seeds and three have ambiguous nearest surfaces.

Local outcomes. Local CR is the reference-collision fraction over retained anchors, averaged equally across scenes; earlier non-reference terminations remain in the denominator without a reference-collision event. This is an observed local event fraction, not a censoring-adjusted risk. Minimum reference clearance $d_{\min}^{\mathrm{ref}}$ is the smallest native-grid distance to the reference patch minus body radius. Table 11 uses branch-specific windows; paired clearance contrasts use the shortest window shared by all four branches. Intervals use 2,000 task-cluster bootstrap draws within scenes, preserving branch pairs and conditioning on these two scenes.

Table 11: **Local disturbance outcomes.** Disturbance magnitude: 0.5 m/s. $\mathrm{CR}^{\mathrm{ref}}$ is in percent and $d_{\min}^{\mathrm{ref}}$ in meters; N counts accepted anchors. Results weight the two scenes equally.

			Clean		Toward		Tangential		Away	
Method	Margin	N	$\mathrm{CR}^{\mathrm{ref}} \downarrow$	$d_{\min}^{\mathrm{ref}} \uparrow$	$\mathrm{CR}^{\mathrm{ref}} \downarrow$	$d_{\min}^{\mathrm{ref}} \uparrow$	$\mathrm{CR}^{\mathrm{ref}} \downarrow$	$d_{\min}^{\mathrm{ref}} \uparrow$	$\mathrm{CR}^{\mathrm{ref}} \downarrow$	$d_{\min}^{\mathrm{ref}} \uparrow$
ACT	Low	17	30.0	0.221	30.0	0.192	32.1	0.202	20.0	0.254
ACT	Medium	15	20.0	0.451	25.0	0.368	10.0	0.480	15.0	0.476
ACT	High	15	5.0	0.836	5.0	0.766	0.0	0.850	5.0	0.867
NavMorph	Low	14	35.0	0.194	35.0	0.178	35.0	0.185	30.0	0.211
NavMorph	Medium	17	15.0	0.476	15.0	0.431	10.0	0.497	15.0	0.514
NavMorph	High	15	0.0	0.831	5.0	0.781	0.0	0.835	0.0	0.870

Directional effects. Toward reduces common-window clearance relative to Clean and Away increases it in all six policy–bin groups (Fig. 14), while collision contrasts vary. On the same 46 NavMorph anchors, reducing the increment to 0.25 m/s increases Toward clearance by 1.2/4.2/3.5 cm in low/medium/high bins over windows common to both strengths, with unchanged local CR. Thus, binary collision counts can conceal a consistent loss of local safety margin.

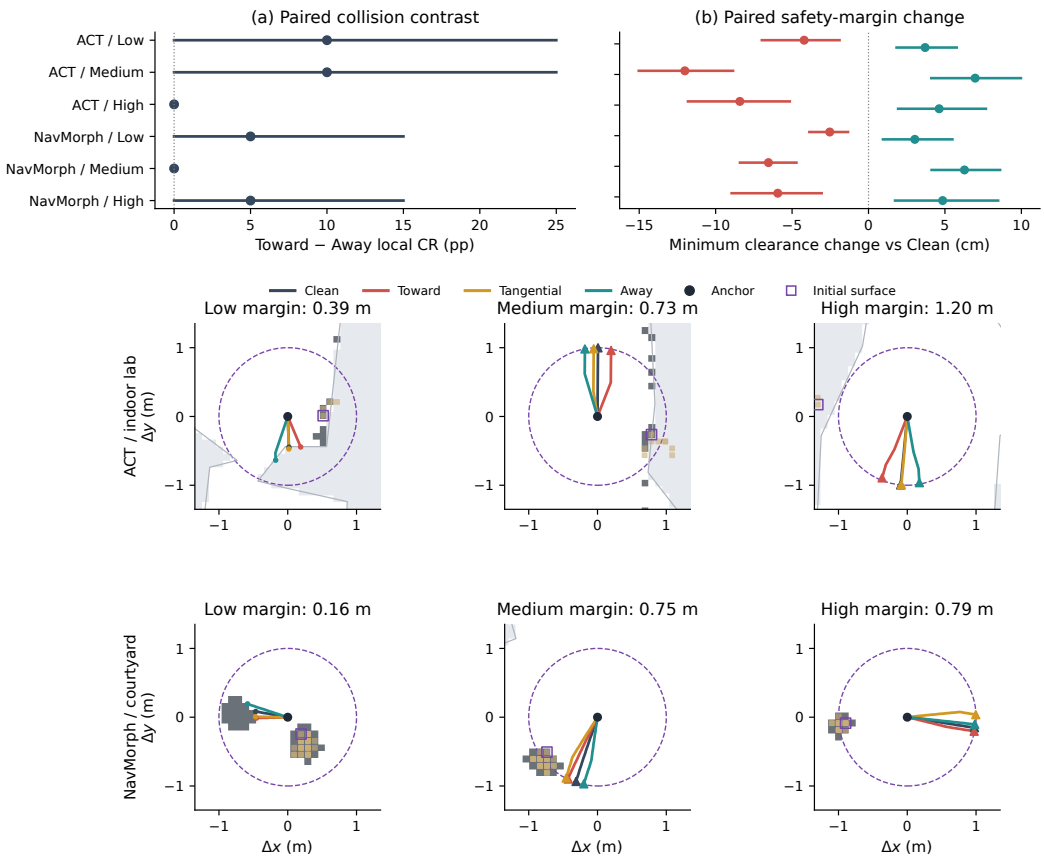


Figure 14: **Paired effects within a fixed local region.** Controlled disturbances at 93 anchors, separate from the 180-sample clearance grouping. Top: local CR and common-window clearance contrasts with 95% task-bootstrap intervals. Bottom: the first eligible anchor in each displayed policy–scene–clearance group. Dashed circles delimit the 1 m observation region; tan cells show the reference patch. Crosses mark reference collisions and triangles region exits. Coordinates are anchor-relative. Zero-width CR intervals arise from all-zero paired differences and do not establish equivalence.