# VISION-LANGUAGE-ACTION MODEL ALREADY HAS ATTENTION HEADS FOR PATH DEVIATION DETECTION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Vision-Language-Action (VLA) models have demonstrated strong potential in navigation tasks by predicting semantic actions and reasoning over complex linguistic and visual contexts. However, they are fundamentally hindered by visual-reasoning hallucinations causing trajectory deviations. Addressing this issue has conventionally required training external critic modules or relying on complex uncertainty heuristics. In this work, we discover that monitoring specific attention heads within a frozen VLA model detects path deviations without incurring additional computational overhead. We term these Navigation Heads, as they inherently capture the spatiotemporal correlations between historical visual sequences and linguistic instructions. Using these heads, we propose an intuitive, training-free anomaly-detection framework that monitors their signals to detect hallucinations in real time. Surprisingly, among over a thousand attention heads, a combination of just three is sufficient to achieve a 65.1 % anomaly detection rate (recall) and a 76.4 % F1 score in unseen environments. Furthermore, we couple this high-level detection logic with a low-level Reinforcement Learning (RL) policy for local obstacle avoidance, ensuring stable navigation and executing a direct safe recovery upon anomaly detection. Ultimately, deploying this entire framework zero-shot onto our customized physical robot demonstrates its practical robustness. All source code will be publicly available.

## 1 INTRODUCTION

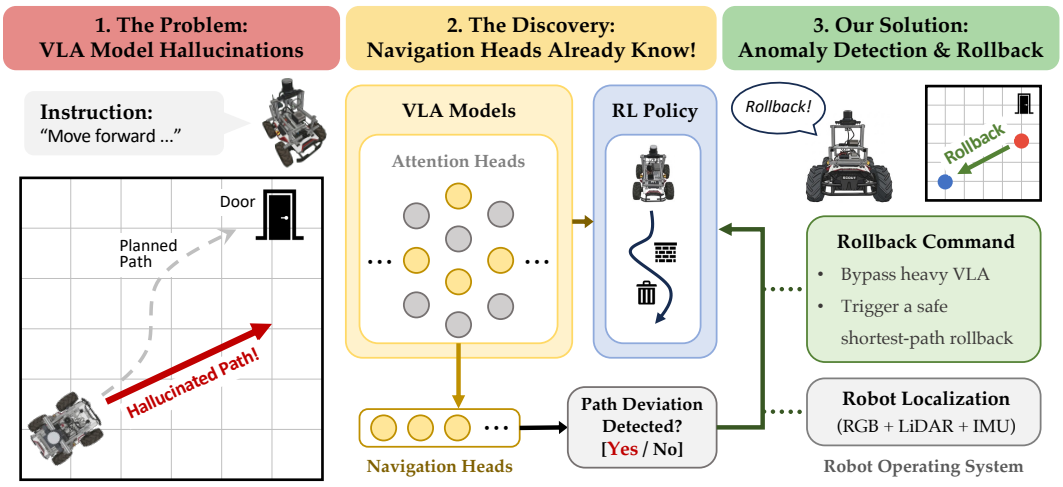



Figure 1: Conceptual overview of our approach to mitigating hallucinations in VLA-based navigation. **1. The Problem:** VLA models are highly vulnerable to vision-language reasoning hallucinations, causing severe trajectory deviations. **2. The Discovery:** We identify Navigation Heads ($H_{\text{nav}}$) that are highly sensitive to navigation states, serving as zero-overhead internal anomaly detectors. **3. Our Solution:** We implement real-time, training-free path deviation detection that instantly triggers a low-level RL policy for safe collision avoidance and rollback.



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.

Driven by recent advances in Large Vision-Language Models (LVLMs), vision-language navigation (VLN) has progressed to Transformer-based policies capable of reasoning over complex multimodal instructions and rich visual observations (Anderson et al., 2018b; Fried et al., 2018; Tan et al., 2019; Hong et al., 2021; Chen et al., 2021b; 2022). More recently, Vision-Language-Action (VLA) models have extended this paradigm by directly mapping sequential visual observations and language instructions to executable navigation actions, enabling robots to operate in continuous, open-world environments without predefined topological constraints (Krantz et al., 2020; Zhang et al., 2024; Cheng et al., 2025; Wang et al., 2024; Marafioti et al., 2025).

However, VLA models fundamentally inherit the chronic hallucination issues of their underlying LLM backbones. As visual histories accumulate, failures in visual reasoning and language grounding often lead to trajectory drift, incorrect sub-goal selection, and ultimately catastrophic path deviations (Tang et al., 2025; Chen et al., 2024b). To address this, prior work has highlighted the importance of detecting uncertainty and initiating backtracking when an agent loses its path (An et al., 2024; Chen et al., 2021a; Ma et al., 2019b; Wang et al., 2023). For instance, current path deviation detection primarily relies on post-facto physical heuristics (e.g., collisions or stagnation) (Hornung et al., 2014; Wellhausen et al., 2020; An et al., 2024), requires computationally expensive fine-tuning of auxiliary networks (Ma et al., 2019a; Ji et al., 2022; Taioli et al., 2024), or invokes latency-heavy external LLMs via APIs (Hu et al., 2023; Rajvanshi et al., 2024). Consequently, these approaches pose critical computational bottlenecks for real-time, high-frequency deployment.

In this work, we ask a different question: Does a frozen VLA model already contain internal mechanisms that reveal whether navigation remains properly grounded? We show that the answer is yes. Specifically, we discover Navigation Heads ($H_{\texttt{nav}}$), a sparse subset of attention heads that consistently aligns spatiotemporal visual observations with navigation instructions. Their structured attention progression tracks instructions during successful navigation, but degrades characteristically as the robot deviates. This enables training-free detection of grounding failures directly from the model's internal computations, eliminating the need for auxiliary detectors.

Building on this insight, we propose a training-free spatiotemporal grounding diagnostic framework that formalizes $H_{\texttt{nav}}$ dynamics. This approach is model-agnostic (Zhang et al., 2024; Cheng et al., 2025) and robustly prevents overfitting by zero-shot evaluating parameters optimized solely on the training split. As illustrated in Fig. 1, our method produces a continuous signal of the perceptual grounding state via real-time attention monitoring, seamlessly identifying path deviations and triggering a rollback to a safe state without additional computational overhead.

Furthermore, to overcome the slow inference speeds inherent to massive VLA models (Chen et al., 2024a; Du et al., 2025; Krantz & Lee, 2022; Elnoor et al., 2024; Liu et al., 2024; Shukor et al., 2025), we integrate a custom low-level Reinforcement Learning (RL) controller trained specifically for obstacle avoidance. Crucially, to minimize the sim-to-real gap, we trained this policy in simulation by precisely replicating the kinematics and exact sensor suite of the physical robot. Finally, we deployed this integrated system directly on-device, validating that the optimal parameters derived from the VLN-CE virtual environment seamlessly transfer to real-world robot deployments without modifications (zero-shot transfer). To the best of our knowledge, we are the first to identify navigation-specific attention heads and successfully leverage them for real-world robot navigation.

To summarize, our main contributions are as follows:

- We identify a sparse subset of Navigation Heads ($H_{\texttt{nav}}$) in a frozen LVLM that aligns spatiotemporal observations with navigation intent, showing they serve as core anomaly indicators for language-guided navigation.
- We propose a training-free anomaly detection framework that monitors attention dynamics in real time using only three heads. Without any additional trainable parameters, it achieves a 65.1 % recall and a 76.4 % F1 score in R2R unseen environments.
- We construct a robust navigation-and-recovery pipeline by integrating the anomaly-detection signal with a lightweight RL controller, maintaining stability during normal driving and ensuring safe, immediate rollbacks upon path deviation.
- We successfully deploy the integrated system on-device, demonstrating that parameters optimized in a simulation featuring a highly accurate model of our physical robot robustly transfer to the real world, fully validating its sim-to-real performance and practical viability.

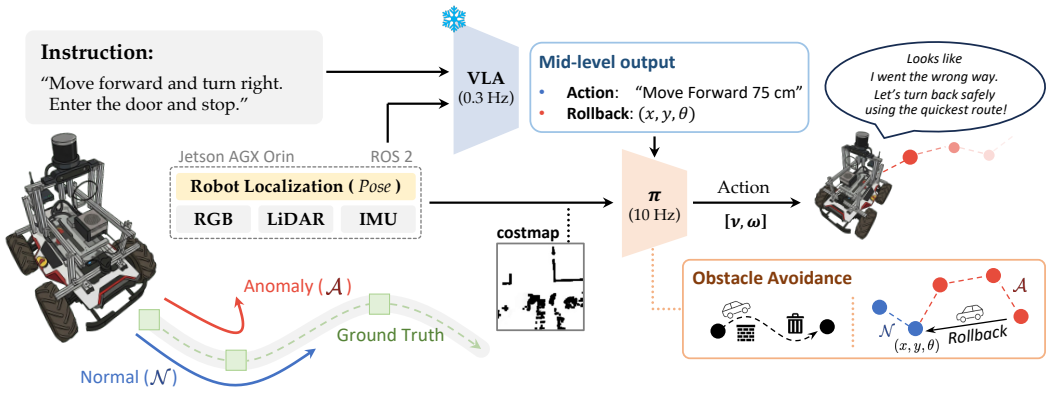



Figure 2: **Overview of the path recovery framework.** Deployed via ROS 2, our hierarchical system integrates a high-level VLA model (0.3 Hz) and a low-level RL policy (10 Hz). Using RGB, instructions, and pose data, the VLA detects real-time path deviations ($\mathcal{N} \rightarrow \mathcal{A}$) and triggers a rollback to the last verified checkpoint $(x, y, \theta)$. Concurrently, the RL policy ($\pi$) utilizes LiDAR costmaps to output collision-free velocity commands $[v, \omega]$. This design enables the robot to dynamically avoid obstacles and safely return to the normal path via the quickest route during recovery.

## 2 Related Work

**Motion-based heuristics detection.** As a fundamental approach, this method relies on rule-based mechanisms to detect a post-facto deadlock, such as physical collisions or position stagnation. Beginning with the DWA (Fox et al., 2002), which detects entrapment in local minima, subsequent robotic systems have utilized various hardware inputs, including wheel odometry (Marder-Eppstein et al., 2010), simple sensor thresholds (Hornung et al., 2014), and multi-modal sensor fusion (Wellhausen et al., 2020), to reactively identify physical entrapments and execute recovery behaviors like map initialization or in-place rotations. This same logic has been recently applied to VLN. For example, An et al. (2024) propose ETPNav, which registers a deadlock if the agent's coordinates remain unchanged after a forward action, subsequently proposing an in-place rotation heuristic.

**High-overhead semantic detection.** To proactively detect trajectory deviations beyond physical deadlocks, recent paradigms leverage semantic understanding but introduce significant computational overhead. One line of research trains auxiliary detection modules alongside the main navigation backbone. For instance, Ma et al. (2019a) introduce a progress monitor (SMNA), while Fried et al. (2018) and Taioli et al. (2024) train separate networks to score trajectory-instruction alignment or localize errors. In physical robot navigation, approaches predict failure probabilities (Ji et al., 2022), leverage self-supervised data for collision prediction (Kahn et al., 2021a;b), request external assistance via RL (Zhu et al., 2021), or measure reconstruction errors on sensor streams (Park et al., 2018; Malhotra et al., 2016). To avoid such expensive fine-tuning, another line of research invokes external massive LLMs or VLMs zero-shot. Hierarchical frameworks (Hu et al., 2023) leverage massive VLMs to guide low-level policies, and Rajvanshi et al. (2024) query external LLMs with scene-graph prompts to refine plans. Consequently, whether relying on fine-tuned auxiliary adapters or auto-regressive text generation via APIs for massive models, these approaches inherently suffer from severe inference latency that prohibits real-time on-device monitoring.

## 3 Method

### 3.1 Preliminaries

**NaVILA Backbone.** NaVILA (Cheng et al., 2025) is a navigation-specific VLA model fine-tuned from VILA-8B (Lin et al., 2024). To capture continuous navigation history, it uniformly samples $N$ past frames at a $384 \times 384$ resolution, encoding them into $27 \times 27$ vision tokens for the underlying LLM (Grattafiori et al., 2024). Processed through 32 layers with 32 self-attention heads each, the model jointly attends to all visual history and instruction tokens to autoregressively predict mid-level actions (e.g., "Turn left 30 degrees") for a low-level controller to execute.

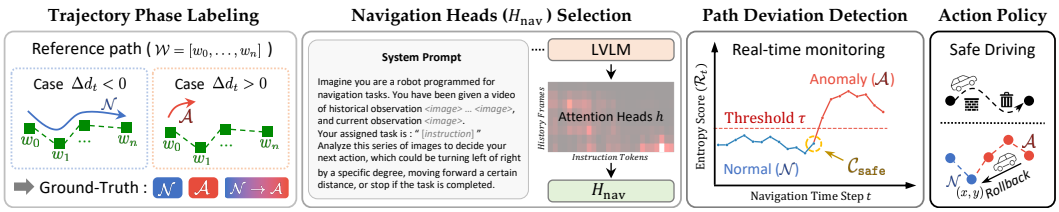



Figure 3: Our core approach consists of four consecutive stages: (i) **Phase Labeling:** Formulating ground-truth labels based on path deviation. (ii) **Head Selection:** Identifying a subset of Navigation heads $H_{\text{nav}}$ sensitive to these state transitions. (iii) **Anomaly Detection:** Monitoring the entropy ($\mathcal{R}_t$) of $H_{\text{nav}}$ to detect real-time failures and preserve safe checkpoints ($\mathcal{C}_{\text{safe}}$). (iv) **Action Policy:** Deploying an RL policy for collision-free navigation and rollbacks triggered by detected anomalies.

**Distinct Characteristics of Attention Heads.** Although Multi-Head Self-Attention (MHSA) is designed for parallel computation, attention heads spontaneously diverge during training to acquire distinct functional roles. For instance, specific heads capture unique linguistic features (Clark et al., 2019), and Voita et al. (2019) demonstrated that only a sparse subset performs the core reasoning, allowing the majority to be pruned without performance degradation (Michel et al., 2019). This fundamental specialization in LLMs (Zheng et al., 2024b) extends directly to LVLMs, where Kang et al. (2025) showed that specific heads govern the semantic alignment between visual and textual tokens. Building on these insights, we analyze internal attention distributions to identify and leverage a sparse subset of heads inherently optimized for spatial navigation tasks.

### 3.2 Trajectory Phase Categorization

To rigorously evaluate attention heads and establish a baseline for anomaly detection, we first construct a dataset by categorizing the agent's trajectories into Normal ($\mathcal{N}$) and Anomaly ($\mathcal{A}$) phases. Given a reference path $\mathcal{W} = [w_0, \ldots, w_n]$ representing a sequence of navigable waypoints for each VLN-CE episode, the navigation state evaluation is conducted through a three-step process:

(i) *Target Tracking:* At each step $t$, we update the target $w_t^*$ to the closest forward waypoint along the reference path. To prevent regression, we enforce a non-decreasing index constraint ($j_t \geq j_{t-1}$).

(ii) *Progress Evaluation:* We evaluate progress by calculating the distance delta $\Delta d_t$. If the agent reaches a new waypoint, $\Delta d_t = 0$. Otherwise, $\Delta d_t$ is the distance change $d_t - d_{t-1}$ (where $d_t$ is the Euclidean distance to $w_t^*$). Consequently, a positive $\Delta d_t$ intuitively indicates a path deviation.

(iii) *Phase Classification:* We employ a one-way state machine with patience $p$. Starting in $\mathcal{N}$, if $p$ consecutive translational steps yield $\Delta d_t > 0$, the state irreversibly transitions to $\mathcal{A}$, retroactively labeling from the first deviation. Rotations are excluded in $\mathcal{N}$ but included in $\mathcal{A}$ as disorientation. Crucially, if $p$ consecutive on-track steps occur in $\mathcal{A}$, episodes truncate to exclude noisy corrections.

Consequently, episodes are categorized as purely $\mathcal{N}$, purely $\mathcal{A}$, or $\mathcal{N} \to \mathcal{A}$ transitions (Fig. 3).

### 3.3 Navigation Heads Selection

Leveraging the distinct functional roles of LVLM attention heads, we identify a navigation-critical subset ($H_{\text{nav}}$). This subset is defined by two core criteria: (i) explicit spatio-temporal alignment, and (ii) high sensitivity to the cognitive anomalies defined in Sec. 3.2.

**Spatio-Temporal Alignment.** We hypothesize specific heads $h$ capture spatiotemporal correlation between $T$ history frames $\mathcal{V} = \{F_k\}_{k=0}^{T-1}$ ($F_0$ is the oldest, $F_{T-1}$ the current frame) and sequential instructions. To isolate them, we extract Instruction-to-Frame attention weights into a 2D alignment matrix $A \in \mathbb{R}^{T \times N}$ for $N$ instruction tokens. We evaluate these heads using a Spatio-Temporal alignment score $I_{\text{diag}}(h)$ capturing observation-level grounding and dynamic spatiotemporal shifts.

First, observation-level grounding is validated via frame energy uniformity ($S_{\text{uniform}}$) and focus sharpness ($S_{\text{peak}}$). To ensure continuous visual grounding without ignoring any frame in the temporal sequence, $S_{\text{uniform}}$ penalizes vanishing attention by measuring the ratio of each frame's energy against the peak episode energy. Concurrently, to verify the agent targets a specific instruction seg-

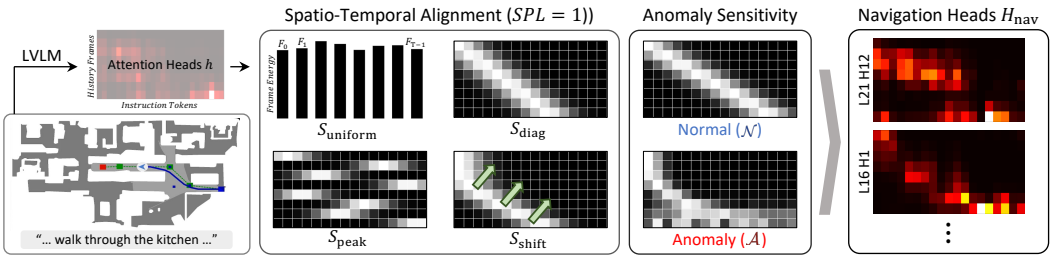



Figure 4: Training-free spatiotemporal grounding framework for navigation. To select robust navigation heads, candidates are evaluated via $I_{\texttt{diag}}(h)$, which comprises even frame energy ($S_{\texttt{uniform}}$), focused instruction attention ($S_{\texttt{peak}}$), diagonal alignment ($S_{\texttt{diag}}$), and smooth transition ($S_{\texttt{shift}}$), and assessed for sensitivity to attention changes between normal and anomalous navigation.

ment rather than dispersing its focus, $S_{\texttt{peak}}$ enforces a sharp, unimodal peak. Given the attention weight $A_{k,j}$ from frame $k$ to token $j$, we formulate frame energy $E_k = \sum_{j=0}^{N-1} A_{k,j}$, normalized distribution $P_{k,j} = A_{k,j}/E_k$, and temporal center of mass $c_k = \sum_{j=0}^{N-1} j \cdot P_{k,j}$. Let $\sigma_k$ and $m_k$ be the standard deviation and local window mass of $P_k$ around $c_k$. We compute:

$$S_{\texttt{uniform}} = \frac{1}{T}\sum_{k=0}^{T-1} \frac{E_k}{\max_{k'} E_{k'}}, \quad S_{\texttt{peak}} = \frac{1}{T}\sum_{k=0}^{T-1} \frac{1}{2}\left(\left(1 - \frac{\sigma_k}{\sigma_{\max}}\right) + m_k\right), \tag{1}$$

where $k' \in \{0, \ldots, T-1\}$ indexes the frames for maximum energy, and $\sigma_{\max}$ represents the standard deviation of a uniform distribution over instruction length.

Second, to evaluate dynamic spatiotemporal shifts, we define the ideal attention peak $i_k^*$, forming a diagonal across $A$. We compute spatial proximity to this diagonal ($S_{\texttt{diag}}$) and penalize erratic jumps to enforce smooth temporal shifts toward the instruction's end as the navigation progresses ($S_{\texttt{shift}}$):

$$\begin{aligned} S_{\texttt{diag}} &= 1 - \frac{1}{T}\sum_{k=0}^{T-1} \frac{|c_k - i_k^*|}{N-1}, \quad \text{where } i_k^* = \left(\frac{k}{T-1}\right)\cdot(N-1), \\ S_{\texttt{shift}} &= \frac{1}{2(T-1)}\sum_{k=1}^{T-1}\left(\mathbb{I}(\Delta c_k > 0) + \exp\left(-\frac{1}{2}\left(\frac{\Delta c_k}{\Delta i_k^*} - 1\right)^2\right)\right), \end{aligned} \tag{2}$$

where $\Delta c_k = c_k - c_{k-1}$ and $\Delta i_k^* = i_k^* - i_{k-1}^*$, with $\mathbb{I}(\cdot)$ as the indicator function.

Ultimately, the alignment score $I_{\texttt{diag}}(h)$ is calculated as the expectation over $\mathcal{E}_{\text{ideal}}$, a set of optimal episodes with a perfect Success weighted by Path Length (SPL) (Anderson et al., 2018a) of 1.0:

$$I_{\texttt{diag}}(h) = \mathbb{E}_{\tau\sim\mathcal{E}_{\text{ideal}}}\left[S^{(h)}_{\texttt{uniform}} \cdot S^{(h)}_{\texttt{peak}} \cdot \left(\lambda S^{(h)}_{\texttt{diag}} + (1-\lambda) S^{(h)}_{\texttt{shift}}\right)\right], \tag{3}$$

where $\lambda$ balances spatial precision with temporal smoothness. Heads consistently yielding high $I_{\texttt{diag}}(h)$ are selected as initial candidates for $H_{\texttt{nav}}$.

**Cognitive Anomaly Sensitivity.** While the alignment score $I_{\texttt{diag}}$ identifies instruction-following heads, robust navigation heads must also exhibit distinct attentional shifts during failures. Therefore, we evaluate candidate sensitivity by quantifying the divergence in focus sharpness ($S_{\texttt{peak}}$) distributions between normal ($\mathcal{N}$) and anomalous ($\mathcal{A}$) phases via Cohen's $d$ (Lakens, 2013):

$$d(h) = \frac{|\mu_{\mathcal{N}} - \mu_{\mathcal{A}}|}{\sigma_{\text{pooled}}}, \quad \text{where } \sigma_{\text{pooled}} = \sqrt{\frac{(n_{\mathcal{N}}-1)\sigma^2_{\mathcal{N}} + (n_{\mathcal{A}}-1)\sigma^2_{\mathcal{A}}}{n_{\mathcal{N}} + n_{\mathcal{A}} - 2}} \tag{4}$$

Here, $\mu$ and $\sigma$ represent the mean and standard deviation of focus sharpness within each partition, with $n$ denoting the sample count. A higher $d(h)$ signifies a substantial shift or collapse in attention focus when the agent becomes stuck. Consequently, we rank candidate heads by $d(h)$ to form $H_{\texttt{nav}}$.

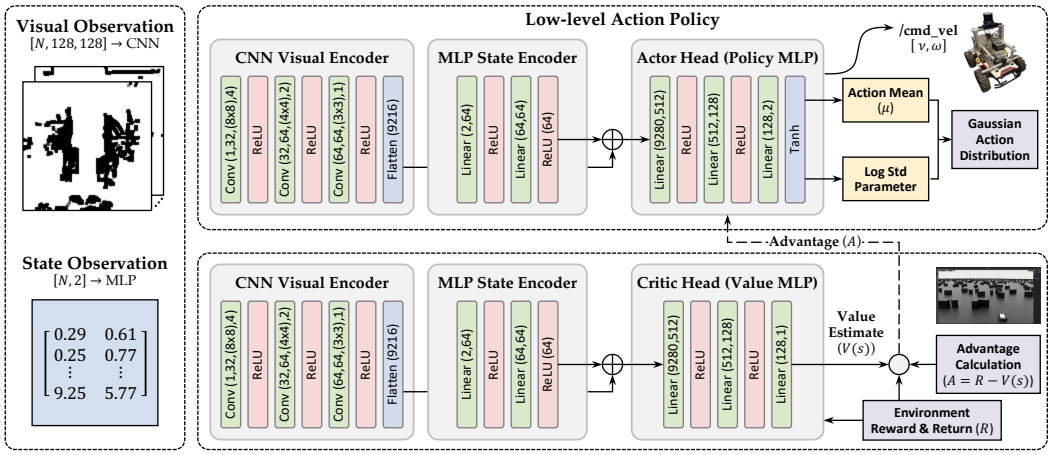



Figure 5: Architecture of the low-level RL action policy. The network fuses CNN-encoded $128 \times 128$ costmaps and MLP-encoded relative subgoals $(x, y)$ to feed two specialized heads: an **Actor** outputting Gaussian distributions for velocity commands $(\upsilon, \omega)$, and a **Critic** estimating state-value $V(s)$. This enables the agent to learn reactive obstacle avoidance and robust subgoal navigation.

### 3.4 Attention-Based Path Deviation Detection

To detect deviations, we compute the mean attention entropy $E_t$ at step $t$ using the top-$K$ heads in $H_{\text{nav}}$. We monitor relative entropy $\mathcal{R}_t$, comparing $E_t$ against its rolling window average of size $W$:

$$\mathcal{R}_t = \frac{E_t}{\frac{1}{W}\sum_{i=1}^{W} E_{t-i} + \epsilon}, \quad \text{where } E_t = \frac{1}{|H_{\text{nav}}|} \sum_{h \in H_{\text{nav}}} \mathcal{H}(A_t^{(h)}). \tag{5}$$

Here, $\mathcal{H}(\cdot)$ is the normalized Shannon entropy averaged over visual frames for scale invariance. High $\mathcal{R}_t$ indicates a sudden dispersion of attention relative to the agent's recent history, serving as a strong precursor to navigation failure. Mirroring ground-truth patience, an anomaly ($\mathcal{A}$) triggers only if $\mathcal{R}_t$ exceeds a predefined threshold $\tau$ for $P$ consecutive steps. In normal phases ($\mathcal{N}$), we continuously update a safe checkpoint $\mathcal{C}_{\text{safe}} = \{s_t, \mathcal{V}_t, \mathcal{M}_t, A_t\}$, preserving the agent's pose $s_t = (x, y, z, \theta)$, visual history $\mathcal{V}_t$, the entropy buffer $\mathcal{M}_t = \{E_{t-W}, \ldots, E_{t-1}\}$, and attention state $A_t$ for recovery. Hyperparameters $(K, W, \tau, P)$ are grid-optimized on the training split, as detailed in Sec. 4.

### 3.5 Collision-Free Low Level Action Policy

To achieve safe collision avoidance and low-level control, we train a local action policy via Proximal Policy Optimization (PPO) (Schulman et al., 2017). To ensure the policy remains agnostic to specific LiDAR hardware configurations, we employ a 2D occupancy grid costmap as an intermediate observation space rather than using raw point clouds. Specifically, we apply ego-motion compensation to spatially align the map with the robot's motion, and we couple it with a temporal decay mechanism that gradually fades historical obstacle data. This formulation enables the system to robustly handle dynamic obstacles and mitigate the impact of temporary sensor blind spots.

As illustrated in Fig. 5, a decoupled actor-critic architecture fuses CNN-extracted spatial costmap features with MLP-encoded relative goals to predict Gaussian velocity commands. The policy is optimized using a dense reward formulation targeting goal-reaching, safety, and smoothness, alongside a curriculum learning strategy for robust long-horizon evasion. Ultimately, this RL policy serves a crucial dual purpose: it not only empowers the robot with immediate reactive obstacle avoidance, but also ensures safe trajectory recovery during anomaly rollbacks. When a path deviation is detected, the robot navigates back to the saved safe point via a direct path, and the robust collision avoidance of our RL policy safely handles any newly encountered or previously unidentified obstacles along this recovery route without relying on high-level VLA reasoning.

For zero-shot real-world deployment, we meticulously model the physical robot and sensors in simulation, with all hyperparameter, reward, and curriculum details provided in Appendix A.2.

# 4 EXPERIMENTS

## 4.1 EXPERIMENTAL SETUP

**Simulation and Training.** Experiments and RL training used one NVIDIA RTX 6000 Ada GPU. We adopt the default NaVILA (Cheng et al., 2025) configuration ($384 \times 384$ resolution) as our primary baseline. To demonstrate generalizability, we evaluate on NaVid (Zhang et al., 2024). The results and IsaacLab implementation details are provided in Appendices A.1 and A.2, respectively.

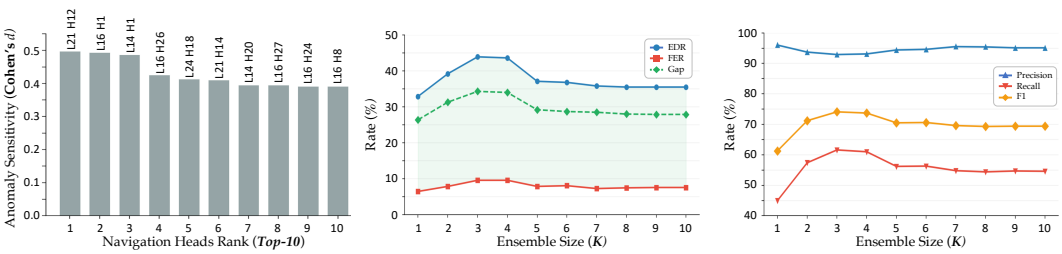
**Real-world Deployment.** The framework was deployed on an AgileX Scout 2.0 mobile robot equipped with an NVIDIA Jetson AGX Orin (64 GB), a ZED 2i camera, and an RS-LiDAR. Remote control is managed via a host PC over 5G, following the setup in (Jeong et al., 2025a). To provide precise odometry for safe point recording and the RL policy, we adopt Fast-LIVO2 (Zheng et al., 2024a), a CPU-based framework. Ultimately, all real-robot deployments succeeded zero-shot using the exact frozen anomaly detection parameters derived from the R2R training split (Sec. 4.2), strictly without any environment-specific tuning. Detailed communication protocols, calibration parameters, and comprehensive state estimation analyses are available in Appendix A.3.

## 4.2 EVALUATING NAVIGATION HEADS FOR ANOMALY DETECTION

Table 1: Ground Truth analysis of trajectory phases (Sec. 3.2). On NaVILA trajectories in VLN-CE R2R, Cohen's $d$ ($> 1.2$) confirms separation in waypoint distances between $\mathcal{N}$ and $\mathcal{A}$ steps. The Val-Unseen 'Only $\mathcal{N}$' ratio (56.0 %) matches the baseline Success Rate, verifying GT reliability.

| Split | Episode Category | | | | Path Distance & Effect Size | | |
|---|---|---|---|---|---|---|---|
| | **Total** | **Only $\mathcal{N}$** | **Only $\mathcal{A}$** | **$\mathcal{N} \to \mathcal{A}$** | **$\mathcal{N}$ Median** | **$\mathcal{A}$ Median** | **Cohen's $d$** |
| Train | 1,000 | 617 (61.7 %) | 79 (7.9 %) | 304 (30.4 %) | 0.13 m | 2.38 m | 1.54 |
| Val-Seen | 778 | 435 (55.9 %) | 91 (11.7 %) | 252 (32.4 %) | 0.14 m | 2.97 m | 1.60 |
| Val-Unseen | 1,839 | 1,029 (56.0 %) | 263 (14.3 %) | 547 (29.7 %) | 0.16 m | 2.53 m | 1.27 |

**Validation of Ground Truth Labels.** We validate our behavioral Ground Truth (GT) labels using NaVILA (Cheng et al., 2025) trajectories from 1,000 random Train episodes and all Val episodes of VLN-CE R2R (Anderson et al., 2018b) (Table 1). Step-level analysis shows that without explicit spatial thresholds, $\mathcal{N}$ steps tightly adhere to the optimal path (0.13–0.16 m median deviation), while $\mathcal{A}$ steps diverge significantly (2.38–2.97 m). A large effect size (Cohen's $d > 1.2$) confirms our criteria isolate severe deviations from minor steering noise. Moreover, the 'Only $\mathcal{N}$' ratio in Val-Unseen (56.0 %) matches baseline Success Rate, justifying GT validity. Finally, $\mathcal{N} \to \mathcal{A}$ transitions show a median $\mathcal{N}$ phase of 6–7 steps preceding an $\mathcal{A}$ phase of 9–10 steps. This distribution provides sufficient history for normal contexts and an adequate window for anomaly detection. Generalization experiments on additional models and datasets are provided in Appendix A.1.




(a) $H_{\text{nav}}$ Discriminability (b) Episode-level $K$ ablation (c) Step-level $K$ ablation

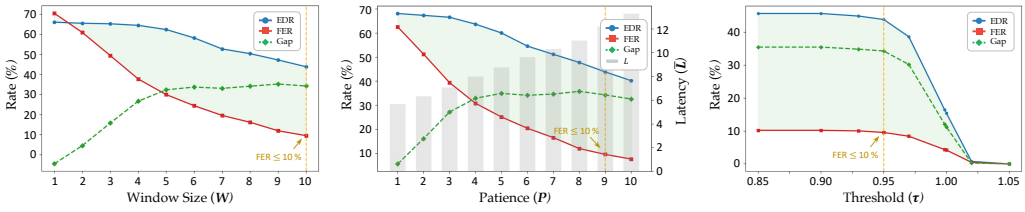
Figure 6: Ablation of navigation heads ($H_{\text{nav}}$) and ensemble size ($K$) on the R2R train split. (a) Head discriminability (Cohen's $d$), where higher values denote greater sensitivity to anomalies. (b) Episode-level performance across $K$. Thresholds are tuned to maximize Episode Detection Rate (EDR) within 5–10 % False Episode Rate (FER) constraint (Gap = EDR − FER). (c) Step-level classification on $\mathcal{N} \to \mathcal{A}$ episodes. Ultimately, an optimal $K = 3$ ensemble yields 43.9 % EDR (at 9.6 % FER) alongside 92.9 % Precision, 61.6 % Recall, and 74.1 % F1 at the step level.

Table 2: Comparison of training-free anomaly detection on R2R validation splits. Using configurations derived solely from the train split, we evaluate against rule-based heuristics to ensure a fair assessment with near-zero inference overhead. Our method significantly outperforms these baselines in episode-level Gap and step-level F1. Notably, while heuristics suffer from severe recall bottlenecks (missing >60 % of anomalies), our approach robustly captures navigation failures.

| Method | R2R Val-Seen | | | | | | R2R Val-Unseen | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | All Episodes | | | $\mathcal{N} \rightarrow \mathcal{A}$ Step-level | | | All Episodes | | | $\mathcal{N} \rightarrow \mathcal{A}$ Step-level | | |
| | EDR ↑ | FER ↓ | Gap ↑ | Prec. ↑ | Rec. ↑ | F1 ↑ | EDR ↑ | FER ↓ | Gap ↑ | Prec. ↑ | Rec. ↑ | F1 ↑ |
| Stagnation | 24.5 % | 4.8 % | 19.7 % | 99.7 % | 31.3 % | 47.7 % | 29.6 % | 5.9 % | 23.7 % | 95.6 % | 38.7 % | 55.1 % |
| Act.Failure | 0.3 % | 6.0 % | −5.7 % | 70.6 % | 2.8 % | 5.5 % | 1.5 % | 6.3 % | −4.8 % | 77.9 % | 5.6 % | 10.4 % |
| Uncertainty | 1.5 % | **0.0 %** | 1.5 % | **100.0 %** | 0.8 % | 1.6 % | 1.7 % | **0.0 %** | 1.7 % | **100.0 %** | 2.1 % | 4.1 % |
| **Ours** | **44.6 %** | 11.7 % | **32.9 %** | 91.3 % | **68.6 %** | **78.3 %** | **41.9 %** | 9.6 % | **32.2 %** | 92.5 % | **65.1 %** | **76.4 %** |

**Path Deviation Detection Performance.** To verify the robustness and generalizability of our mathematically formalized, data-driven framework, we derive optimal configurations strictly from the R2R train split and evaluate them zero-shot on validation splits. As shown in Fig. 6a, we extract a subset of $H_{\text{nav}}$ and conduct a grid search over 9,000 combinations, comprising ensemble size ($K \in \{1, \ldots, 10\}$), patience ($P \in \{1, \ldots, 10\}$), sliding window ($W \in \{1, \ldots, 10\}$), and 9 values for activation threshold ($\tau$) (individual trends for $P$, $W$, and $\tau$ are detailed in Fig. 7). Optimization results on the train split (Fig. 6b) indicate that, under a 10 % False Episode Rate (FER) constraint, the $K = 3$ ensemble achieves the peak Episode Detection Rate (EDR) and detection Gap. At the step level on $N \rightarrow A$ transitions (Fig. 6c), this configuration captures anomalies with 92.9 % Precision (avoiding false alarms), 61.6 % Recall (identifying path deviations), and a 74.1 % F1 score.

Table 2 compares our approach against zero-overhead heuristics (Marder-Eppstein et al., 2010; An et al., 2024) and an uncertainty baseline. Notably, Position Stagnation (Marder-Eppstein et al., 2010) (fixed $(x, y)$ displacement thresholds), Action Failure (An et al., 2024) (coordinate shifts during forward actions), and the uncertainty baseline (action probability $< 50\,\%$) universally suffer from critically low recall. Because LVLMs remain overconfident during hallucinations and persistently output movement commands, coordinate- or logit-based triggers fail to detect deviations. Conversely, our method monitors intrinsic semantic misalignment to robustly capture navigation failures. Despite a marginal FER increase, our approach achieves superior recall (68.6 % Val-Seen, 65.1 % Val-Unseen) and step-level F1 scores (78.3 %, 76.4 %). Qualitative results and validation on another VLA model (Zhang et al., 2024) are detailed in Fig. 9 and Appendix A.1.3, respectively.




(a) Effect of Window size (b) Effect of Patience (c) Effect of Threshold

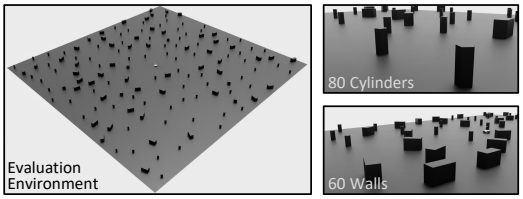
Figure 7: Hyperparameter grid search for the proposed anomaly detector ($K = 3$) on the R2R train split. (a) Both EDR and detection Gap (EDR − FER) saturate near $W = 7$. (b) Higher $P$ suppresses FER but linearly increases mean detection latency. (c) The detector is highly sensitive near $\tau = 1.0$, where EDR and FER drop rapidly. Constrained by FER $< 10\,\%$, we select $(W = 10, P = 9, \tau = 0.95)$, achieving 43.9 % EDR while preventing unnecessary interruptions.

### 4.3 RL Obstacle Avoidance Performance

**Evaluation Setup.** We evaluate the RL policy in IsaacLab (Mittal et al., 2025a). The testbed (Fig. 8b) comprises a 60 m × 60 m terrain with 80 cylindrical and 60 wall-shaped obstacles (2.0 m minimum clearance). Across 2,048 episodes (512 per 5, 10, 15, 20 m target), the policy maps local costmaps to velocities bounded to $v \in [-0.5, 0.5]$ m/s and $\omega \in [-1.0, 1.0]$ rad/s. Episodes terminate upon reaching the target (< 0.5 m, SR), timeout (> 150.0 s, TR), or heavy collision (> 200 N, CR).

| Metric | SR ↑ | CR ↓ | TR ↓ |
|---|---|---|---|
| APF (Khatib, 1986) | 47.4 % | 8.6 % | 43.9 % |
| DWA (Fox et al., 2002) | 17.8 % | 79.6 % | **2.6 %** |
| MPPI (Williams et al., 2017) | 2.5 % | 91.2 % | 6.3 % |
| TEB (Rösmann et al., 2012) | 47.2 % | 43.6 % | 9.2 % |
| **Ours** | **87.2 %** | **5.3 %** | 7.5 % |

(a) RL Policy Performance




(b) Evaluation Environments

Figure 8: **Evaluation of the low-level RL obstacle avoidance policy.** (a) Comparison of average metrics (SR, CR, TR) against classical baselines. Our policy significantly outperforms traditional methods, achieving the highest Success Rate (87.2 %). (b) The 60 m × 60 m IsaacLab testbed, populated with procedural cylindrical and wall-shaped obstacles to assess collision-free navigation.

**Evaluation Analysis.** Table 8a compares our policy against four classical local planners using purely local observations (a 128 × 128 costmap and relative goal position). While baselines (Khatib, 1986; Rösmann et al., 2012; Fox et al., 2002; Williams et al., 2017) heavily struggle with local minima or dense collisions (CR up to $> 79\,\%$), our RL policy achieves the highest SR (87.2 %), outperforming the best baseline by $\sim$40 %, with minimal CR (5.3 %) and TR (7.5 %). This demonstrates our tailored RL framework effectively equips the policy with robust spatial reasoning for dense navigation. Detailed per-baseline analyses and distance-wise metrics are provided in Appendix A.2.5.

### 4.4 Real-world Evaluation

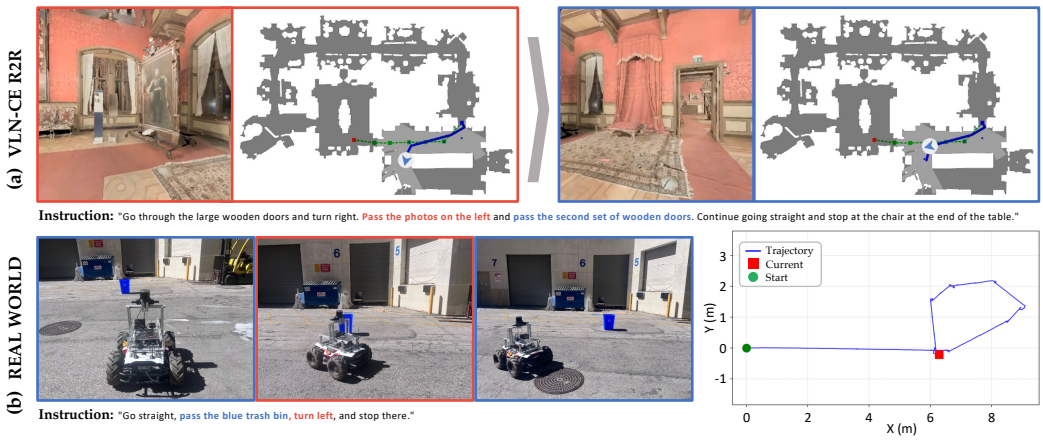



Figure 9: (a) Hallucination-induced path deviation and positional reset in the VLN-CE simulation. (b) Real-world demonstration of VLA-RL integrated path recovery on a custom on-device robot.

Fig. 9 demonstrates our framework on a VLA-driven robot. Upon detecting deviations, the RL policy safely executes trajectory rollbacks. These real-world deployments strictly use parameters derived from R2R without environment-specific tuning. Further experiments are provided in the Appendix.

## 5 Conclusion

We present a training-free anomaly detection and recovery framework for frozen VLAs. By analyzing the internals of a frozen VLA to isolate Navigation Heads ($H_{\text{nav}}$) that capture spatiotemporal alignment between vision and instructions, our system detects real-time path deviations via attention dynamics without computational overhead. Integrating this signal with a low-level RL policy ensures safe physical recovery and reactive obstacle avoidance. Ultimately, this mitigates VLA hallucinations, providing a practical solution for deploying reliable, self-monitoring embodied agents. However, simply resetting the robot's physical position without resolving the model's internal state can trigger repetitive errors. Future work will explore active replanning or re-prompting to align these internal states and break these error loops. Additionally, our identification of $H_{\text{nav}}$ offers a strategic pathway for model compression via attention pruning (Michel et al., 2019). By selectively retaining only these task-essential heads and pruning the redundant ones, future frameworks can significantly accelerate on-device inference without sacrificing navigation capabilities.

### AI USE STATEMENT

In this work, we used generative AI tools for proofreading and refining the grammatical structure of the manuscript. We have not used generative AI tools for data generation, conducting physical robot experiments, or formulating the core theoretical frameworks, and the generation of quantitative experimental results is not applicable to this work. We have thoroughly reviewed all AI-assisted work. Specifically, all LLM-refined text was verified by the authors to ensure scientific accuracy and strict alignment with our original intentions, and all source code was manually reviewed and empirically tested on physical hardware. We take full responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

### ETHICS STATEMENT

This work adheres to the ICLR Code of Ethics. Our research focuses on improving the safety and reliability of autonomous navigation systems by detecting trajectory deviations and executing collision-free rollbacks. This research does not involve human subjects, and all experiments were conducted using publicly available simulated datasets and safe, controlled real-world robot deployments. We acknowledge the broad potential dual-use nature of autonomous navigation technologies. However, our primary contribution is fundamentally defensive and safety-oriented, designed to prevent catastrophic failures and mitigate hallucinations in embodied agents leveraging LLM backbones. We do not anticipate any harmful societal impacts directly arising from this methodology.

### REPRODUCIBILITY STATEMENT

We are committed to ensuring the full reproducibility of our proposed training-free anomaly detection and path recovery framework. For the simulated environments, we provide detailed descriptions of the IsaacLab testbed, procedural obstacle generation, and the Reinforcement Learning (RL) reward and curriculum formulations in the Supplementary Material. For real-world deployments, the exact communication protocols, hardware calibration parameters, and state estimation details for the custom physical robot are extensively documented in the Appendix. Furthermore, to facilitate future research, all source code, including the proposed anomaly detection module, the RL policy, and the trajectory phase categorization logic, will be made publicly available as an anonymous downloadable repository upon publication.

## REFERENCES

Dong An, Hanqing Wang, Wenguan Wang, Zun Wang, Yan Huang, Keji He, and Liang Wang. Etpnav: Evolving topological planning for vision-language navigation in continuous environments. IEEE TPAMI, 2024.

Peter Anderson, Angel Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, et al. On evaluation of embodied navigation agents. arXiv preprint arXiv:1807.06757, 2018a.

Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton Van Den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In CVPR, pp. 3674–3683, 2018b.

Till Beemelmanns. ros2_calib: A Multi-Sensor Calibration Tool for ROS2. `https://github.com/ika-rwth-aachen/ros2_calib`, 2025. Accessed: 2026-03-11.

Kevin Chen, Junshen K Chen, Jo Chuang, Marynel Vázquez, and Silvio Savarese. Topological planning with transformers for vision-and-language navigation. In CVPR, pp. 11276–11286, 2021a.

Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, Junyang Lin, Chang Zhou, and Baobao Chang. An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models. In ECCV, pp. 19–35, 2024a.

Shizhe Chen, Pierre-Louis Guhur, Cordelia Schmid, and Ivan Laptev. History aware multimodal transformer for vision-and-language navigation. In NeurIPS, volume 34, pp. 5834–5847, 2021b.

Shizhe Chen, Pierre-Louis Guhur, Makarand Tapaswi, Cordelia Schmid, and Ivan Laptev. Think global, act local: Dual-scale graph transformer for vision-and-language navigation. In CVPR, pp. 16537–16547, 2022.

Zhaorun Chen, Zhuokai Zhao, Hongyin Luo, Huaxiu Yao, Bo Li, and Jiawei Zhou. HALC: Object hallucination reduction via adaptive focal-contrast decoding. arXiv preprint arXiv:2403.00425, 2024b.

An-Chieh Cheng, Yandong Ji, Zhaojing Yang, Zaitian Gongye, Xueyan Zou, et al. Navila: Legged robot vision-language-action model for navigation. In RSS, 2025.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. What does BERT look at? an analysis of BERT's attention. In BlackboxNLP, pp. 276–286, 2019.

Yi Du, Taimeng Fu, Zhuoqun Chen, Bowen Li, Shaoshu Su, Zhipeng Zhao, and Chen Wang. VL-Nav: Real-time vision-language navigation with spatial reasoning. arXiv preprint arXiv:2502.00931, 2025.

Mohamed Elnoor, Kasun Weerakoon, Gershom Seneviratne, Ruiqi Xian, Tianrui Guan, Mohamed Khalid M Jaffar, Vignesh Rajagopal, and Dinesh Manocha. Robot navigation using physically grounded vision-language models in outdoor environments. arXiv preprint arXiv:2409.20445, 2024.

Dieter Fox, Wolfram Burgard, and Sebastian Thrun. The dynamic window approach to collision avoidance. IEEE Robotics & Automation Magazine, 4(1):23–33, 2002.

Daniel Fried, Ronghang Hu, Volkan Cirik, Anna Rohrbach, Jacob Andreas, et al. Speaker-follower models for vision-and-language navigation. In NeurIPS, 2018.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. arXiv preprint arXiv:2407.21783, 2024.

Yicong Hong, Qi Wu, Yuankai Qi, Cristian Rodriguez-Opazo, and Stephen Gould. Vln bert: A recurrent vision-and-language bert for navigation. In CVPR, pp. 1643–1653, 2021.

Rachel Hornung, Holger Urbanek, Julian Klodmann, Christian Osendorfer, and Patrick van der Smagt. Model-free robot anomaly detection. In IROS, pp. 3676–3683, 2014.

Yingdong Hu, Fanqi Lin, Tong Zhang, Li Yi, and Yang Gao. Look before you leap: Unveiling the power of gpt-4v in robotic vision-language planning. arXiv preprint arXiv:2311.17842, 2023.

Jaehwan Jeong, Tuan-Anh Vu, Mohammad Jony, Shahab Ahmad, Md Mukhlesur Rahman, Sangpil Kim, and M Khalid Jawed. Agrichrono: A multi-modal dataset capturing crop growth and lighting variability with a field robot. arXiv preprint arXiv:2508.18694, 2025a.

Jaehwan Jeong, Tuan-Anh Vu, Radha Lahoti, Jiawen Wang, Vivek Alumootil, Sangpil Kim, and M Khalid Jawed. Vision-guided targeted grasping and vibration for robotic pollination in controlled environments. arXiv preprint arXiv:2510.06146, 2025b.

Tianchen Ji, Arun Narenthiran Sivakumar, Girish Chowdhary, and Katherine Driggs-Campbell. Proactive anomaly detection for robot navigation with multi-sensor fusion. IEEE RA-L, 7(2): 4975–4982, 2022.

Gregory Kahn, Pieter Abbeel, and Sergey Levine. BADGR: An autonomous self-supervised learning-based navigation system. IEEE RA-L, 6(2):1312–1319, 2021a.

Gregory Kahn, Pieter Abbeel, and Sergey Levine. LaND: Learning to navigate from disengagements. IEEE RA-L, 6(2):1872–1879, 2021b.

Seil Kang, Jinyeong Kim, Junhyeok Kim, and Seong Jae Hwang. Your large vision-language model only needs a few attention heads for visual grounding. In CVPR, pp. 9339–9350, 2025.

Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. The International Journal of Robotics Research, 5(1):90–98, 1986.

Jacob Krantz and Stefan Lee. Sim-2-sim transfer for vision-and-language navigation in continuous environments. In ECCV, pp. 588–603, 2022.

Jacob Krantz, Erik Wijmans, Arjun Majumdar, Dhruv Batra, and Stefan Lee. Beyond the nav-graph: Vision-and-language navigation in continuous environments. In ECCV, pp. 104–120, 2020.

Alexander Ku, Peter Anderson, Roma Patel, Eugene Ie, and Jason Baldridge. Room-across-room: Multilingual vision-and-language navigation with dense spatiotemporal grounding. In EMNLP, pp. 4392–4412, 2020.

Daniël Lakens. Calculating and reporting effect sizes to facilitate cumulative science: A practical primer for $t$-tests and ANOVAs. Frontiers in Psychology, 4:863, 2013.

Ji Lin, Hongxu Yin, Wei Ping, Yao Lu, Pavlo Molchanov, Andrew Tao, Huizi Mao, Jan Kautz, Mohammad Shoeybi, and Song Han. Vila: On pre-training for visual language models. In CVPR, 2024.

Zhijian Liu, Ligeng Zhu, Baifeng Shi, Zhuoyang Zhang, Yuming Lou, Shang Yang, et al. Nvila: Efficient frontier visual language models. arXiv preprint arXiv:2412.04468, 2024.

Chih-Yao Ma, Jiasen Lu, Zuxuan Wu, Ghassan AlRegib, Zsolt Kira, Richard Socher, and Caiming Xiong. Self-monitoring navigation agent via auxiliary progress estimation. arXiv preprint arXiv:1901.03035, 2019a.

Chih-Yao Ma, Zuxuan Wu, Ghassan AlRegib, Caiming Xiong, and Zsolt Kira. The regretful agent: Heuristic-aided navigation through progress estimation. In CVPR, pp. 6732–6740, 2019b.

Pankaj Malhotra, Anusha Ramakrishnan, Gaurangi Anand, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff. LSTM-based encoder-decoder for multi-sensor anomaly detection. arXiv preprint arXiv:1607.00148, 2016.

Andrés Marafioti, Orr Zohar, Miquel Farré, Merve Noyan, Elie Bakouch, Pedro Cuenca, Cyril Zakka, Loubna Ben Allal, Anton Lozhkov, Nouamane Tazi, et al. SmolVLM: Redefining small and efficient multimodal models. arXiv preprint arXiv:2504.05299, 2025.

Eitan Marder-Eppstein, Eric Berger, Tully Foote, Brian Gerkey, and Kurt Konolige. The office marathon: Robust navigation in an indoor office environment. In ICRA, pp. 300–307, 2010.

Paul Michel, Omer Levy, and Graham Neubig. Are sixteen heads really better than one? In NeurIPS, 2019.

Mayank Mittal, Pascal Roth, James Tigue, Antoine Richard, Octi Zhang, Peter Du, Antonio Serrano-Munoz, Xinjie Yao, René Zurbrügg, Nikita Rudin, et al. Isaac Lab: A GPU-accelerated simulation framework for multi-modal robot learning. arXiv preprint arXiv:2511.04831, 2025a.

Mayank Mittal, Pascal Roth, James Tigue, Antoine Richard, Octi Zhang, Peter Du, et al. Isaac lab: A gpu-accelerated simulation framework for multi-modal robot learning. arXiv preprint arXiv:2511.04831, 2025b.

Edwin Olson. AprilTag: A robust and flexible visual fiducial system. In ICRA, pp. 3400–3407, 2011.

Daehyung Park, Yuuna Hoshi, and Charles C Kemp. A multimodal anomaly detector for robot-assisted feeding using an LSTM-based variational autoencoder. IEEE RA-L, 3(3):1544–1551, 2018.

Abhinav Rajvanshi, Karan Sikka, Xiao Lin, Bhoram Lee, Han-Pang Chiu, and Alvaro Velasquez. SayNav: Grounding large language models for dynamic planning to navigation in new environments. In ICAPS, volume 34, pp. 464–474, 2024.

Christoph Rösmann, Wendelin Feiten, Thomas Wösch, Frank Hoffmann, and Torsten Bertram. Trajectory modification considering dynamic constraints of autonomous robots. In ROBOTIK 2012; 7th German Conference on Robotics, pp. 1–6, 2012.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347, 2017.

Antonio Serrano-Muñoz, Dimitrios Chrysostomou, Simon Bøgh, and Nestor Arana-Arexolaleiba. skrl: Modular and flexible library for reinforcement learning. Journal of Machine Learning Research, 24(254):1–9, 2023. URL http://jmlr.org/papers/v24/23-0112.html.

Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, et al. Smolvla: A vision-language-action model for affordable and efficient robotics. arXiv preprint arXiv:2506.01844, 2025.

Francesco Taioli, Stefano Rosa, Alberto Castellini, Lorenzo Natale, Alessio Del Bue, Alessandro Farinelli, Marco Cristani, and Yiming Wang. Mind the error! detection and localization of instruction errors in vision-and-language navigation. In IROS, pp. 12993–13000, 2024.

Hao Tan, Licheng Yu, and Mohit Bansal. Learning to navigate unseen environments: Back translation with environmental dropout. In NAACL, pp. 2610–2621, 2019.

Feilong Tang, Chengzhi Liu, Zhongxing Xu, Ming Hu, Zile Huang, Haochen Xue, Ziyang Chen, Zelin Peng, Zhiwei Yang, Sijin Zhou, et al. Seeing far and clearly: Mitigating hallucinations in MLLMs with attention causal decoding. In CVPR, pp. 26147–26159, 2025.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In ACL, pp. 5797–5808, 2019.

Hanqing Wang, Wei Liang, Luc Van Gool, and Wenguan Wang. Dreamwalker: Mental planning for continuous vision-language navigation. In ICCV, pp. 10873–10883, 2023.

Zihan Wang, Xiangyang Li, Jiahao Yang, Yeqi Liu, Junjie Hu, Ming Jiang, and Shuqiang Jiang. Lookahead exploration with neural radiance representation for continuous vision-language navigation. In CVPR, pp. 13753–13762, 2024.

Lorenz Wellhausen, René Ranftl, and Marco Hutter. Safe robot navigation via multi-modal anomaly detection. IEEE RA-L, 5(2):1326–1333, 2020.

Grady Williams, Nolan Wagener, Brian Goldfain, Paul Drews, James M Rehg, Byron Boots, and Evangelos A Theodorou. Information theoretic MPC for model-based reinforcement learning. In ICRA, pp. 1714–1721, 2017.

Jiazhao Zhang, Kunyu Wang, Rongtao Xu, Gengze Zhou, Yicong Hong, Xiaomeng Fang, Qi Wu, Zhizheng Zhang, and He Wang. Navid: Video-based vlm plans the next step for vision-and-language navigation. arXiv preprint arXiv:2402.15852, 2024.

Chunran Zheng, Wei Xu, Zuhao Zou, Tong Hua, Chongjian Yuan, Dongjiao He, Bingyang Zhou, Zheng Liu, Jiarong Lin, Fangcheng Zhu, et al. FAST-LIVO2: Fast, direct LiDAR–inertial–visual odometry. IEEE Transactions on Robotics, 41:326–346, 2024a.

Zifan Zheng, Yezhaohui Wang, Yuxin Huang, Shichao Song, Mingchuan Yang, Bo Tang, Feiyu Xiong, and Zhiyu Li. Attention heads of large language models: A survey. arXiv preprint arXiv:2409.03752, 2024b.

Yi Zhu, Yue Weng, Fengda Zhu, Xiaodan Liang, Qixiang Ye, Yutong Lu, and Jianbin Jiao. Self-motivated communication agent for real-world vision-dialog navigation. In ICCV, pp. 1594–1603, 2021.

# A Appendix

## A.1 Expanding Path Deviation Detection

### A.1.1 Computational Overhead Analysis

Table 3: Computational overhead analysis on a single NVIDIA RTX 6000 Ada GPU. The integration of our path deviation detection module introduces a marginal latency increase of less than 20 ms per step, with no additional GPU memory required.

| Configuration | Total Time | Alloc VRAM | Peak VRAM |
|---|---|---|---|
| NaVILA (Baseline) | 583.5 ms | 17,145.6 MB | 18,730.3 MB |
| NaVILA + Ours | 603.3 ms | 17,145.6 MB | 18,730.3 MB |
| **Difference** | **+19.8** ms | **0.0** MB | **0.0** MB |

Expanding upon the performance discussions in Sec. 4.2 of the main paper, Table 3 details the precise computational requirements of our path deviation detection mechanism on an NVIDIA RTX 6000 Ada GPU. The table demonstrates that our module introduces a near-zero computational overhead, adding less than 20 ms of latency per step without requiring additional peak GPU memory. Additionally, the computational overhead on the NVIDIA Jetson AGX Orin edge device can be found in Table 14. During inference, the computational logic remains highly streamlined. Provided that the task-relevant navigation heads ($H_{\texttt{nav}}$) and the optimal natural threshold ($\tau$) have been pre-identified through offline statistical analysis on the training data, online real-time inference merely requires a simple $\mathcal{O}(1)$ lookup. This allows the system to instantly extract the attention distributions specific to $H_{\texttt{nav}}$. The subsequent anomaly detection logic comprises calculating the Shannon entropy over the visual tokens, computing the relative difference against a rolling-window mean, and comparing it against the predefined threshold. Because these steps consist entirely of lightweight scalar operations, the entire detection mechanism operates without introducing any new neural network computations beyond what the base model already performs.

### A.1.2 Additional Experiments on the RxR Dataset

Table 4: **Path Deviation Detection Performance on the RxR Dataset.** Extending the primary evaluation on the R2R dataset (Table 2 of the main paper), we further validate our proposed method on the RxR dataset (Ku et al., 2020). The detection mechanism, configured with $H_{\texttt{nav}}$ and hyperparameters derived exclusively from the training set, demonstrates robust generalizability across RxR validation splits. Most importantly, it successfully maintains an overall F1 score of over 50 % in both environments by balancing exceptionally high Precision with robust Recall.

| Method | RxR Val-Seen | | | | | | RxR Val-Unseen | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | All Episodes | | | $\mathcal{N} \rightarrow \mathcal{A}$ Step-level | | | All Episodes | | | $\mathcal{N} \rightarrow \mathcal{A}$ Step-level | | |
| | EDR ↑ | FER ↓ | Gap ↑ | Prec. ↑ | Rec. ↑ | F1 ↑ | EDR ↑ | FER ↓ | Gap ↑ | Prec. ↑ | Rec. ↑ | F1 ↑ |
| **Ours** | 36.1 % | 6.7 % | 29.4 % | 96.9 % | 39.8 % | 56.5 % | 32.7 % | 9.7 % | 23.0 % | 93.2 % | 34.4 % | 50.3 % |

Following the primary results presented in Table 2 of the main paper, we conducted additional evaluations on the RxR (Room-across-Room) dataset (Ku et al., 2020) to demonstrate the dataset generalizability and robustness of our approach. To identify the navigation heads ($H_{\texttt{nav}}$), we strictly followed the exact same offline analysis protocol, deriving them exclusively from the RxR training set to prevent any potential data leakage. Once identified, these head configurations and their corresponding hyperparameters were completely frozen and evaluated on both the Val-Seen and Val-Unseen splits, with the results presented in Table 4. Considering that the RxR dataset is characterized by significantly higher linguistic complexity and longer, more intricate spatial trajectories compared to R2R, these results sufficiently prove the data-level robustness of our methodology. In particular, successfully maintaining an F1 score over 50 % across both the Val-Seen and Val-Unseen splits confirms that our attention-based deviation detection serves as a robust, dataset-agnostic indicator of navigational confidence.

### A.1.3 Additional Experiments on the NaVid Model

Table 5: Extending our evaluation beyond the primary NaVILA backbone, we validate our training-free anomaly detection pipeline on NaVid (Zhang et al., 2024). Configured with heads and hyperparameters optimized strictly on the R2R Train split, our method demonstrates robust architecture generalizability. Most importantly, it significantly outperforms rigid heuristics and uncertainty baselines, achieving the highest Recall and overall F1 scores.

| **Baseline** | **R2R Val-Seen** | | | | | | **R2R Val-Unseen** | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| (Navid[1]) | **All Episodes** | | | $\mathcal{N} \rightarrow \mathcal{A}$ **Step-level** | | | **All Episodes** | | | $\mathcal{N} \rightarrow \mathcal{A}$ **Step-level** | | |
| **Method** | **EDR** ↑ | **FER** ↓ | **Gap** ↑ | **Prec.** ↑ | **Rec.** ↑ | **F1** ↑ | **EDR** ↑ | **FER** ↓ | **Gap** ↑ | **Prec.** ↑ | **Rec.** ↑ | **F1** ↑ |
| Stagnation | 22.4 % | **6.3 %** | 16.1 % | 91.7 % | 30.6 % | 45.9 % | 18.1 % | **7.0 %** | 11.1 % | 95.6 % | 23.9 % | 38.2 % |
| Act.Failure | 3.1 % | 6.8 % | -3.7 % | 80.6 % | 3.3 % | 6.3 % | 2.6 % | 8.8 % | -6.2 % | 70.5 % | 3.7 % | 7.1 % |
| Uncertainty | **37.0 %** | 31.1 % | 5.9 % | **100.0 %** | 17.3 % | 29.5 % | **28.9 %** | 27.1 % | 1.8 % | **96.1 %** | 18.5 % | 31.0 % |
| **Ours** | 32.3 % | 8.1 % | **24.2 %** | 79.7 % | **57.3 %** | **66.7 %** | 27.9 % | 12.1 % | **15.8 %** | 77.8 % | **53.2 %** | **63.2** % |

To rigorously verify the architecture-agnostic generalizability of our proposed framework, we extended our evaluation to NaVid (Zhang et al., 2024). Despite having fundamentally different backbones compared to NaVILA (Cheng et al., 2025)—specifically, employing Vicuna-7B and EVA-CLIP rather than VILA-8B and CLIP—we strictly applied the protocol from Sec. 4.2 of the main paper without any heuristic modifications. During this validation, the specific head combinations and hyperparameters derived exclusively from the R2R Train split remained completely frozen. As shown in Table 5, the empirical results exhibit a similar trend to Table 2 in the main paper. While conventional heuristics still suffer from low failure detection rates, our methodology successfully identified attention heads exhibiting analogous spatiotemporal alignment patterns within NaVid. Consequently, our approach achieved superior Recalls of 57.3 % and 53.2 %, alongside the highest overall F1 scores of 66.7 % and 63.2 % in the Val-Seen and Val-Unseen environments, respectively. These findings strongly confirm that our framework operates independently of specific VLA parameters, successfully translating internal model interpretability into a robust robot control signal.

## A.2 Obstacle Avoidance RL Action Policy

### A.2.1 IsaacLab Simulation Setup

To facilitate robust sim-to-real transfer, we meticulously designed the RL training environment within IsaacLab (Mittal et al., 2025b), as outlined in Fig. 10. Crucially, this simulation is engineered to precisely mirror our real-world physical setup, as demonstrated in Fig. 13. The comprehensive list of physical and environmental parameters utilized for this configuration is detailed in Table 6.

**Robot Configuration and Kinematics.** To minimize the reality gap, we constructed a custom USD model of our mobile robot, the AgileX Scout 2.0, and imported it into the simulation at a 1:1 scale. Because the physical robot operates on a skid-steer differential drive mechanism, we explicitly engineered the left and right wheel velocities to be governed by the kinematic equations $v_{\text{left}} = (v - \omega \cdot L/2)/R$ and $v_{\text{right}} = (v + \omega \cdot L/2)/R$, where $v$ and $\omega$ denote the linear and angular velocities, respectively. We strictly applied the real-world specifications: a wheelbase of $L = 0.582\,\text{m}$ and a wheel radius of $R = 0.165\,\text{m}$. The RL policy utilizes a $\tanh$ activation to output continuous actions bounded within $[-1, 1]$, which we explicitly mapped to a maximum linear velocity of $0.5\,\text{m/s}$ and a maximum angular velocity of $1.0\,\text{rad/s}$. To define the physical collision boundaries during navigation, we adopted a conservative rectangular robot dimension of $1.06\,\text{m} \times 0.90\,\text{m}$, incorporating an empirical safety margin over the bare chassis measured directly on the physical robot. To balance rapid reactive obstacle avoidance with computational efficiency while maintaining physical realism, we designed the policy to operate at a control frequency of $15\,\text{Hz}$ (*i.e.*, $\approx 0.067\,\text{s}$ per step), allowing a maximum translational displacement of $\approx 0.033\,\text{m}$ per action step, while the underlying physics engine simulates dynamics at $60\,\text{Hz}$.

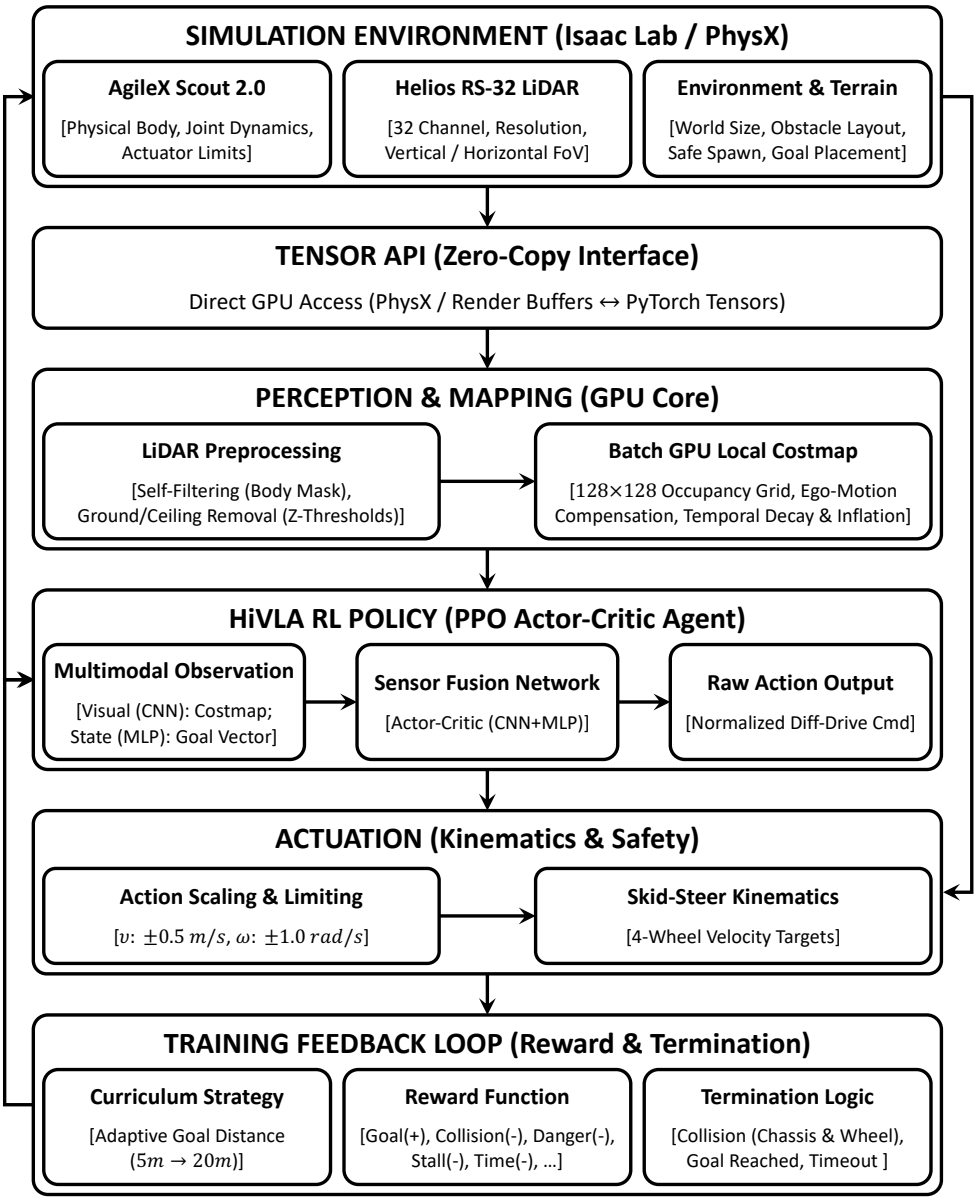



Figure 10: **RL training diagram.** Utilizing IsaacLab (Mittal et al., 2025b) and PhysX, we construct a simulation environment that precisely aligns with real-world dynamics, incorporating the USD-defined robot geometry and sensor extrinsics while tuning skid-steer friction, damping, and actuator forces alongside the 32-channel LiDAR configurations. To facilitate reward discovery and ensure stable convergence, we employ a curriculum strategy that adaptively scales the navigation challenge by increasing the goal distance from 5 m to 20 m across 20 sub-terrains mixed with cylinders and walls, as illustrated in Fig. 8b of the main paper. With rewards tailored for collision avoidance and goal-reaching, this large-scale training of 1,024 agents on a single NVIDIA RTX 6000 Ada GPU enables comprehensive exploration of the state space and robust policy optimization.

**Sensor Setup.** To guarantee seamless sim-to-real transfer of perception, we configured the 3D LiDAR module in IsaacLab to faithfully replicate the mounting position and hardware specifications of the physical Helios RS-32 LiDAR used in our real-world deployments. Specifically, the simulated sensor is mounted at a height of $0.63\,\text{m}$ above the ground. It uses 32 vertical channels to enforce a vertical field of view (FOV) from $-55^\circ$ to $+15^\circ$, and a full $360^\circ$ horizontal FOV. With a $0.2^\circ$ horizontal resolution, the sensor casts 57,632 rays per scan up to a maximum range of $10.0\,\text{m}$, providing a dense geometric representation of the surroundings identical to the real robot's sensory input.

Table 6: **Configuration of the IsaacLab (Mittal et al., 2025b) Simulation Environment**

| Category | Parameter | Value |
|---|---|---|
| Robot & Kinematics | Wheel Radius ($R$) / Wheelbase ($L$) | 0.165 m / 0.582 m |
| | Robot Dimensions (Length $\times$ Width) | 1.06 m $\times$ 0.90 m |
| | Max Linear / Angular Velocity | ±0.5 m/s / ±1.0 rad/s |
| | Policy Control Frequency | 15 Hz (≈0.067 s/step) |
| 3D LiDAR Sensor | Sensor Mounting Height | 0.63 m |
| | Channels / Max Range | 32 / 10.0 m |
| | Vertical / Horizontal FOV | -55° to +15° / 360° |
| | Horizontal Resolution | 0.2° (57,632 rays/scan) |
| Terrain & Environment | Sub-terrains (Size / Count) | 60 m $\times$ 60 m / 20 tiles |
| | Cylinder Obstacles | 80 per tile (Radius: 0.25 m, Height: 2.0 m) |
| | Wall Obstacles | 60 per tile (1.5 m $\times$ 0.75 m $\times$ 2.0 m) |
| | Min Obstacle Clearance | 1.8 m |
| | Min Spawn Clearance | 1.5 m |
| Termination Conditions | Goal Reached Threshold | 0.5 m |
| | Timeout Limit | 150.0 s |
| | Collision Threshold (Chassis / Wheels) | $>$ 200.0 N |
| | Stall Condition | $<$ 0.2 m displacement in 3.0 s |

**Environment and MDP Definitions.** Training occurs across a procedurally generated terrain composed of 20 sub-terrains. To enforce a reproducible and physically viable evaluation standard, we populated each sub-terrain tile with 80 cylinders (radius $0.25$ m, height $2.0$ m) and 60 wall-shaped boxes ($1.5$ m $\times 0.75$ m $\times 2.0$ m). We configured a rejection-sampling algorithm to position the obstacles, ensuring a minimum obstacle-to-obstacle clearance of $1.8$ m, with a $3.0$ m-wide obstacle-free platform preserved at the center of each tile. At episode initialization, we programmed the robot to spawn $0.5$ m above the ground, with a randomized yaw orientation to prevent physics clipping, and to enforce a strict $1.5$ m obstacle-free spawn clearance. We defined the Markov Decision Process (MDP) to terminate under four distinct conditions: (i) reaching within $0.5$ m of the goal, (ii) exceeding the maximum time limit of $150.0$ s, (iii) experiencing physical collisions exceeding $200.0$ N on either the chassis or the wheels (after a brief 20-step warm-up period to ignore initial spawn impacts), or (iv) stalling (accumulating less than $0.2$ m of total displacement for over $3.0$ s).

### A.2.2 Dynamic 2D Costmap Observation

Rather than relying exclusively on sparse 1D LiDAR arrays, our RL policy operates on a dynamic 2D occupancy grid costmap. This intermediate representation, as illustrated in Fig. 11, equips the agent with spatial memory without requiring Recurrent Neural Networks (RNNs) and ensures sim-to-real robustness through sensor agnosticism. While raw 1D data is easily overfitted to specific hardware specifications, the 2D costmap decouples the input space from the raw sensor stream, allowing the policy to remain fully compatible with different LiDAR models or depth cameras without retraining. The visual observation space is a single-channel $128 \times 128$ tensor that captures a $6.4$ m $\times$ $6.4$ m physical area centered on the robot, yielding a dense, uniform $0.05$ m/pixel resolution.

**Hardware-Accelerated Pipeline and Sim-to-Real Alignment.** Processing 57,632 LiDAR rays on the CPU for thousands of parallel agents in simulation induces a severe computational bottleneck. To resolve this, we constructed the entire mapping pipeline using CUDA-optimized PyTorch operations and executed it directly on the GPU in FP16 precision. To eliminate the sim-to-real gap in the perception module, this identical PyTorch codebase is deployed on the physical robot's NVIDIA Jetson AGX Orin. By setting the batch size to 1, the physical robot leverages its native Tensor Cores to process the costmap in real time with the same logic, bypassing CPU-GPU transfer overheads.

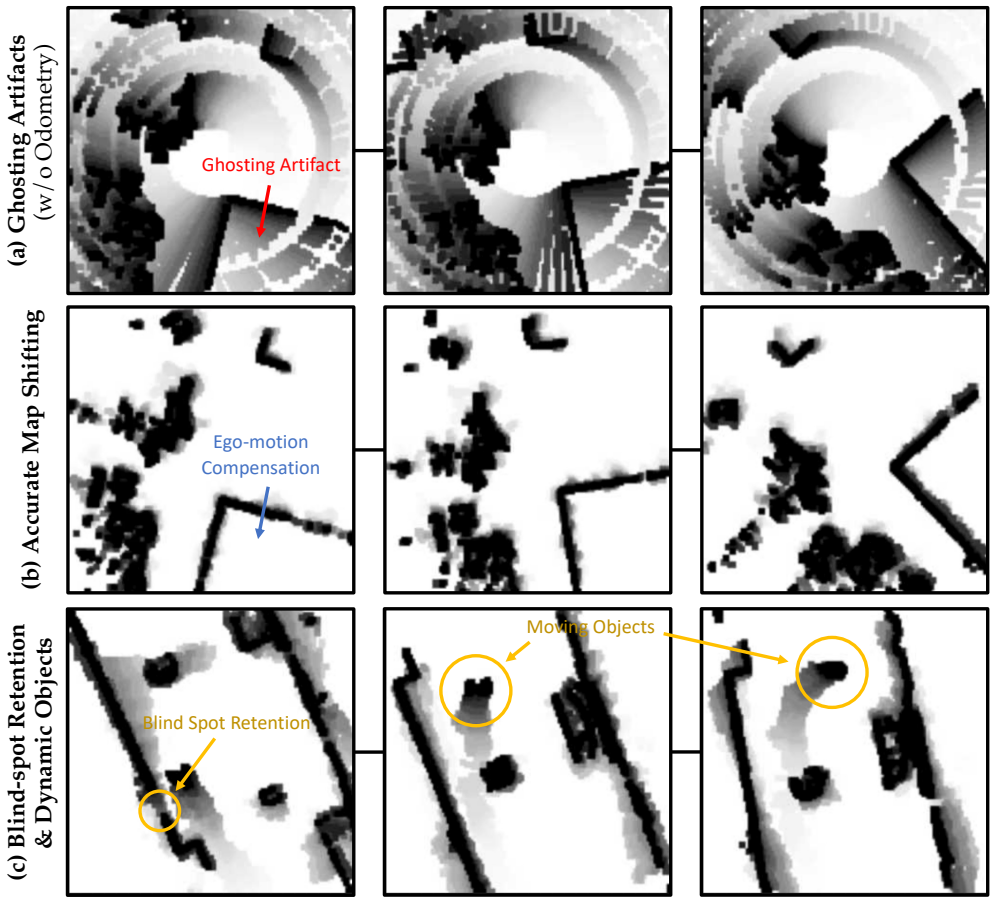



Figure 11: **Dynamic 2D Costmap Observation with Ego-Motion Compensation and Temporal Decay.** (a) Without proper odometry, uncompensated ego-motion causes severe ghosting artifacts during a clockwise rotation, permanently corrupting the spatial memory. (b) By incorporating strict ego-motion compensation during the same rotation, the local costmap accurately shifts in accordance with the robot's movement, maintaining a clean and consistent representation. (c) Applying an optimal temporal decay factor ($\gamma = 0.9$) allows the system to temporarily retain static obstacles even when they enter the sensor's blind spot, while gracefully fading the historical trails of dynamic moving objects. Consequently, this accurate spatial representation prevents the downstream RL policy from reacting to false obstacles, ensuring safe and robust collision avoidance.

**Ego-Motion Compensation via Zero-Allocation Warping.** To maintain a globally consistent map, the local costmap must accurately shift in response to the robot's movement. Our pipeline integrates 3D LiDAR point clouds with high-frequency odometry data to compute the relative translational and rotational displacements at each policy step. To eliminate real-time execution overhead, we pre-allocate an affine transformation matrix buffer, eradicating memory allocation and Garbage Collection (GC) spikes during navigation. Subsequently, we warp the previous costmap using nearest-neighbor grid sampling (`F.grid_sample`) to apply the inverse of the robot's motion. During this process, newly revealed areas entering the field of view are strictly zero-padded to ensure safety.

**Temporal Decay for Dynamic Environments.** Real-world environments inherently contain moving objects. If the costmap remains static (*i.e.*, a decay factor of 1.0), a passing pedestrian would leave a permanent obstacle trail, completely blocking the robot's path. Conversely, decaying the map too rapidly causes the agent to instantly forget static obstacles in temporary blind spots, leading to collisions. To resolve this dilemma, we apply a temporal decay factor of $\gamma = 0.9$ at each step. This specific rate strikes an optimal balance for spatial memory—safely remembering temporary blind spots while gracefully fading the historical trails of dynamic entities.

**Spatial Filtering and Safety Margins.** Finally, to ensure the policy reacts exclusively to genuine navigational hazards, we filter the LiDAR points along the Z-axis, retaining only data within $0.1\text{ m} \leq Z \leq 1.0\text{ m}$ to mask out ground noise and overhead structures. Furthermore, we apply a geometric self-masking filter to prevent the robot from perceiving its own chassis or antennas as obstacles. Rather than relying on simple CAD dimensions, the boundaries of this $1.10\text{ m} \times 0.80\text{ m}$ exclusion footprint were empirically derived by directly measuring the physical RS-LiDAR sensor's self-occlusion view. Any points falling within this measured zone are strictly ignored. To prevent the policy from shaving corners too closely, we apply a 2D Max Pooling operation ($3 \times 3$ kernel) over the grid, geometrically inflating all valid obstacles by an exact 1-pixel (0.05 m) safety margin with negligible computational overhead.

### A.2.3 Reinforcement Learning Configuration

Table 7: **Dense Reward Function Configuration.** The total reward is the sum of these 12 components. $\mathbb{I}_{\text{blk}}$ denotes the path-blocked indicator ($d_{\text{corridor}} < 0.8$ m).

| Category | Component | Formulation & Condition |
|---|---|---|
| **Task** | Goal Progress ($r_{\text{goal}}$) | $10 \cdot (d_{t-1} - d_t) + 50 \cdot (e^{-0.25 d_t} - e^{-0.25 d_{t-1}})$ |
| | Goal Success ($r_{\text{success}}$) | $+20.0$ if $d_t < 0.5$ m |
| | Time Penalty ($r_{\text{time}}$) | $-0.05$ per step |
| **Kinematics** | Projected Velocity ($r_{\text{vel}}$) | $3.0 \cdot v_x \cos(\psi_{\text{err}}) \cdot (1 - 0.85\,\tau_{\text{side}}) \cdot (1 - \mathbb{I}_{\text{blk}})$ |
| | Heading Alignment ($r_{\text{head}}$) | $1.0 \cdot (1 - \lvert\psi_{\text{err}}\rvert/\pi) \cdot \mathbb{I}(\lvert v_x\rvert > 0.1) \cdot (1 - \mathbb{I}_{\text{blk}})$ |
| | Yaw Error Reduction ($r_{\text{yaw}}$) | $2.0 \cdot (\lvert\psi_{\text{err}}\rvert_{t-1} - \lvert\psi_{\text{err}}\rvert_t)$ |
| | Micro-Displacement ($r_{\text{move}}$) | $0.2 \cdot \tanh(\Delta s / 0.03)$ |
| **Safety** | Collision Penalty ($r_{\text{col}}$) | $-50.0$ if force $> 200$ N (chassis or wheels) |
| | Proximity Danger ($r_{\text{danger}}$) | $-3.0 \cdot (1 - \text{clamp}(d_{\text{corridor}}/2.18, 0, 1))^2$ |
| **Behavior** | Angular Penalty ($r_{\text{ang}}$) | $-1.0 \cdot \lvert\omega_z\rvert \cdot (1 - \tau_{\text{side}}) \cdot (1 - \mathbb{I}_{\text{blk}})$ |
| | Centering Reward ($r_{\text{center}}$) | $2.5 \cdot (\tau_R - \tau_L) \cdot \omega_z$ |
| | Stall Penalty ($r_{\text{stall}}$) | $\max(-0.5 \cdot \max(t_{\text{stall}} - 1.0, 0)^2, -2.0)$ |

We utilize Proximal Policy Optimization (PPO) to train the navigation policy within the IsaacLab environment. The training configuration is divided into the design of a dense reward function and the specification of PPO network hyperparameters, as detailed in Tables 7 and 8, respectively.

**Reward Function.** To cultivate robust navigation in constrained environments, we engineered a dense reward function ($r_{\text{total}}$) structured into four domains (Table 7). To formalize the robot's spatial perception, we define $d_{\text{corridor}}$ as the minimum obstacle distance within a forward safety corridor, flagging the path as blocked ($\mathbb{I}_{\text{blk}} = 1$) if $d_{\text{corridor}} < 0.8$ m. Lateral tightness ($\tau_{\text{side}} = \max(\tau_L, \tau_R) \in [0, 1]$) quantifies the narrowness of the immediate surroundings. The rationale for each component is as follows: (i) **Task Progression:** Relying solely on a sparse success bonus ($r_{\text{success}}$) granted at a 0.5 m threshold is inefficient for long-horizon navigation. We employ a potential-based shaping reward ($r_{\text{goal}}$) utilizing an exponential function of the 2D goal distance ($d_t$). This provides a steepening, continuous gradient, effectively preventing the agent from being trapped in local minima. A step-wise time penalty ($r_{\text{time}}$) further encourages efficient routing. (ii) **Kinematic Control:** To ensure dynamically feasible trajectories, the projected velocity reward ($r_{\text{vel}}$) incentivizes forward motion ($v_x$) aligned with the goal ($\cos(\psi_{\text{err}})$). Crucially, scaling this reward by $\tau_{\text{side}}$ encourages the policy to accelerate aggressively in open spaces while decelerating cautiously in narrow aisles. This is complemented by continuous heading alignment ($r_{\text{head}}$), step-wise yaw error reduction ($r_{\text{yaw}}$), and micro-displacement ($r_{\text{move}}$) rewards to maintain momentum without lateral drifting. (iii) **Safety Enforcement:** For reliable sim-to-real transfer, a heavy discrete penalty ($r_{\text{col}}$) is strictly applied for physical collisions exceeding 200 N of force on the chassis or wheels. To proactively deter reckless driving, we introduce a proximity danger penalty ($r_{\text{danger}}$) formulated as a quadratic function of $d_{\text{corridor}}$. This establishes a virtual repulsive field that punishes dangerous approaches long before physical collisions occur. (iv) **Behavioral Regularization:** Skid-steer robots often suffer from oscillatory freezing in tight spaces. We penalize excessive angular velocity ($r_{\text{ang}}$) in open areas and punish prolonged stalls ($r_{\text{stall}}$) exceeding a 1.0 s grace period. Most importantly, a centering reward ($r_{\text{center}}$) leverages the spatial imbalance between the left and right clearances ($\tau_R - \tau_L$) to naturally steer the robot toward the wider side, keeping it smoothly centered through bottlenecks.

Table 8: **PPO Network and Training Hyperparameters** (SKRL (Serrano-Muñoz et al., 2023))

| Hyperparameter | Value | Hyperparameter | Value |
|---|---|---|---|
| Algorithm | PPO | Learning Rate | $3 \times 10^{-4}$ |
| Parallel Environments | 1,024 | LR Scheduler | KL-Adaptive (0.008) |
| Rollout Length | 96 steps | Discount Factor ($\gamma$) | 0.99 |
| Learning Epochs | 5 | GAE Parameter ($\lambda$) | 0.95 |
| Mini-batches | 4 | Total Timesteps | 100,000 |
| State Preprocessor | Disabled | Checkpoint Interval | 5,000 steps |

**PPO Hyperparameters.** Expanding upon the reinforcement learning methodology introduced in Sec. 3.5 of the main paper, Table 8 details the SKRL-based PPO configuration Serrano-Muñoz et al. (2023) for stabilizing 1,024 parallel IsaacLab environments. The specific rationale for each key hyperparameter is as follows: (i) **Temporal Discounting and Advantage Estimation:** We apply the standard discount factor of $\gamma = 0.99$ to preserve long-horizon reward signals across diverse procedurally generated sub-terrains. This temporal weighting is paired with a Generalized Advantage Estimation (GAE) parameter of $\lambda = 0.95$ to strike an optimal balance between bias and variance. (ii) **Observation and Batch Processing:** We completely disabled the default state preprocessor. Since our primary visual observation is a structured 2D occupancy costmap, standard continuous normalization would distort its absolute geometric scale and distinct boundary values, thereby degrading the spatial memory of the CNN encoder. For policy optimization, collecting 96 rollout steps across all 1,024 environments yields a massive batch of 98,304 transitions per iteration. To maximize sample efficiency, these transitions are rigorously optimized over 5 epochs and 4 mini-batches. (iii) **Action Space and Actor Architecture:** Driven by the non-holonomic kinematics of our target wheeled robot, the action space is explicitly restricted to two continuous control commands: the forward linear velocity ($v_x$) and the angular velocity ($\omega_z$). To ensure policy stability and prevent the generation of physically unbounded commands, the final output layer of the actor network employs a $\tanh$ activation function. This strictly bounds the raw model outputs to the normalized range $[-1, 1]$ and subsequently scales them to match the robot chassis's maximum physical velocity limits during simulation. (iv) **Adaptive Learning Rate:** To maintain policy stability across diverse procedurally generated sub-terrains, we employ a KL-adaptive learning rate scheduler starting at a base rate of $3 \times 10^{-4}$. By dynamically scaling the learning rate based on a target KL divergence threshold of 0.008, we strictly bound the magnitude of policy updates. This mechanism prevents excessively large policy steps, which can lead to catastrophic forgetting, ensuring that the robot retains its critical collision-avoidance reflexes throughout 100,000 training timesteps.

### A.2.4 RL Policy Computational Resources

Table 9: **Computational overhead of the RL Action Policy.** Evaluated on a single NVIDIA RTX 6000 Ada GPU, the lightweight CNN + MLP architecture (4.89 M parameters) achieves a near-instantaneous inference latency of 0.2 ms per step with minimal VRAM utilization.

| Configuration | Total Time | Alloc. VRAM | Peak VRAM |
|---|---|---|---|
| **Ours (RL Policy)** | 0.2 ms | 26.9 MB | 27.9 MB |

The RL Action Policy is designed as a streamlined CNN + MLP architecture with approximately 4.89 M parameters, specifically optimized for high-frequency reactive control. As detailed in Table 9, this lightweight configuration enables ultra-low-latency inference, requiring only 0.2 ms per step on an NVIDIA RTX 6000 Ada GPU. With a minimal VRAM footprint of less than 28 MB, this RL policy provides the essential agility for real-time obstacle avoidance and stability. By maintaining a negligible computational load even during continuous high-rate actuation, it ensures overall system efficiency and ultimately guarantees a stable control frequency of 10 Hz during real-world deployment, as detailed in Sec. A.3.

### A.2.5 Performance Metrics by Target Distance

Table 10: Navigation performance by distance. SR: Success Rate, CR: Collision Rate, TR: Timeout Rate.

| **Distance** | 5m | | | 10m | | | 15m | | | 20m | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Method** | SR ↑ | CR ↓ | TR ↓ | SR ↑ | CR ↓ | TR ↓ | SR ↑ | CR ↓ | TR ↓ | SR ↑ | CR ↓ | TR ↓ |
| APF Khatib (1986) | 60.6 % | **5.9 %** | 33.6 % | 48.4 % | 10.0 % | 41.6 % | 41.4 % | 10.0 % | 48.6 % | 39.3 % | 8.8 % | 51.9 % |
| DWA Fox et al. (2002) | 17.6 % | 75.8 % | 6.6 % | 16.2 % | 82.8 % | **1.0 %** | 21.1 % | 78.1 % | **0.8 %** | 16.2 % | 81.8 % | **2.0 %** |
| MPPI Williams et al. (2017) | 3.7 % | 90.0 % | **6.3 %** | 2.7 % | 91.6 % | 5.7 % | 1.6 % | 91.6 % | 6.8 % | 2.0 % | 91.4 % | 6.6 % |
| TEB Rösmann et al. (2012) | 65.6 % | 24.0 % | 10.4 % | 51.0 % | 40.6 % | 8.4 % | 41.0 % | 50.4 % | 8.6 % | 31.1 % | 59.4 % | 9.6 % |
| **Ours** | **83.8 %** | 6.4 % | 9.8 % | **88.1 %** | **4.1 %** | 7.8 % | **90.0 %** | **5.1 %** | 4.9 % | **86.7 %** | **5.5 %** | 7.8 % |

Expanding upon Sec. 4.3 of the main paper, Table 10 details the performance across target distances from 5 m to 20 m to analyze long-horizon robustness beyond the average metrics. As distance increases, baselines exhibit significant degradation: APF's termination rate (TR) rises to 51.9 % at 20 m due to cumulative entrapment in local minima, while TEB's collision rate (CR) climbs to 59.4 % in complex layouts. DWA and MPPI consistently fail across all ranges with CR exceeding 75 %. In contrast, our policy maintains a Success Rate (SR) of 86.7 % even at the maximum distance, more than doubling the performance of the strongest baseline (APF, 39.3 %). Notably, despite the fourfold increase in distance, our CR remains exceptionally stable below 6.5 %, proving that the policy effectively avoids local minima and maintains consistent safety in dense environments where classical reactive planners frequently stall or collide.

### A.2.6 Real-World Reactive Navigation of the RL Policy

To evaluate the low-level action policy for obstacle avoidance, we demonstrated the real-world navigation performance of the standalone RL policy without VLA guidance. Given simple forward commands over distances of tens of meters, the agent operates at 10 Hz to safely navigate through dynamic and unstructured environments. As shown in Fig. 12, (a) the policy successfully avoids both static objects and moving pedestrians indoors, and (b) it demonstrates robust collision avoidance against complex, irregularly shaped trees in an outdoor orchard farm.

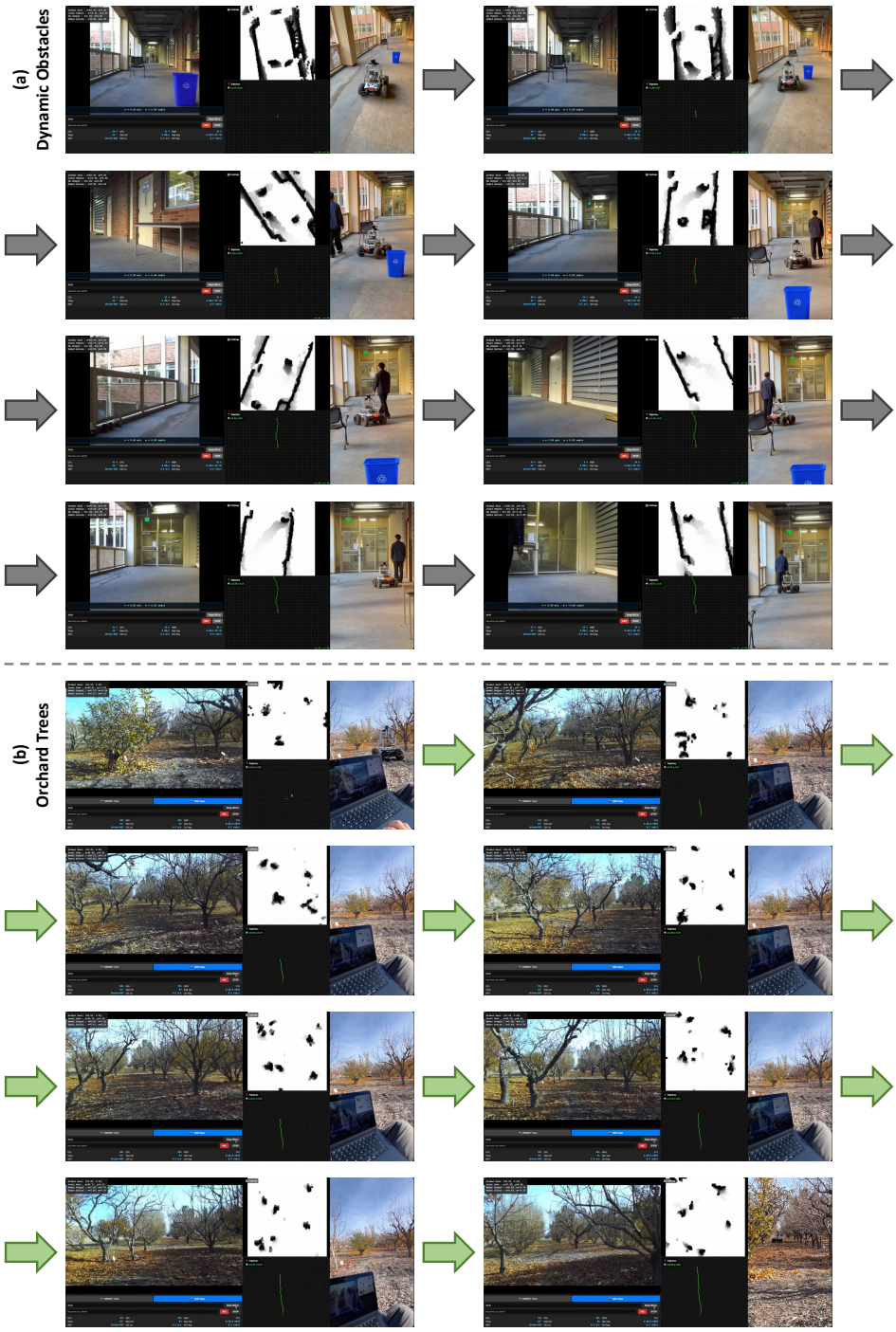



Figure 12: **Real-world navigation performance of the standalone RL policy.** Operating at 10 Hz without VLA guidance, the agent safely navigates dynamic environments over tens of meters. **(a)** Indoors, it successfully avoids static objects and moving pedestrians. **(b)** In an outdoor orchard, it demonstrates robust collision avoidance against complex, irregular trees.

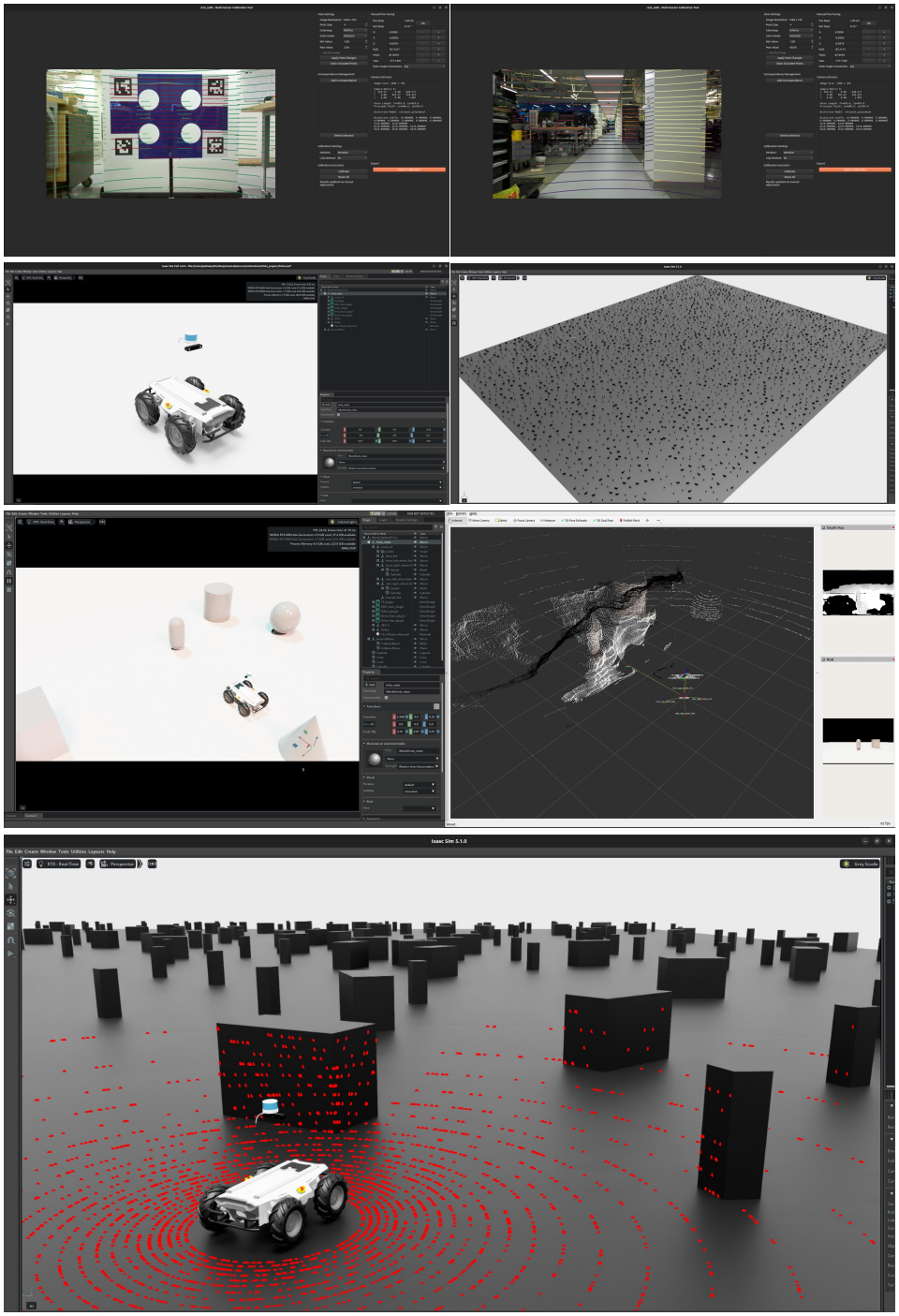


Figure 13: Overview of the robotic system and simulation environment. **First row**: sensor calibration of the ZED 2i camera and RS-32 LiDAR. **Second row**: USD robot model with sensor extrinsics and training terrains. **Third row**: Testing real-time ROS 2 integration within Isaac Sim. **Fourth row**: training visualization of a single agent among 1,024 parallel instances.

## A.3 Real-Robot Platform System Implementation

### A.3.1 System Specifications

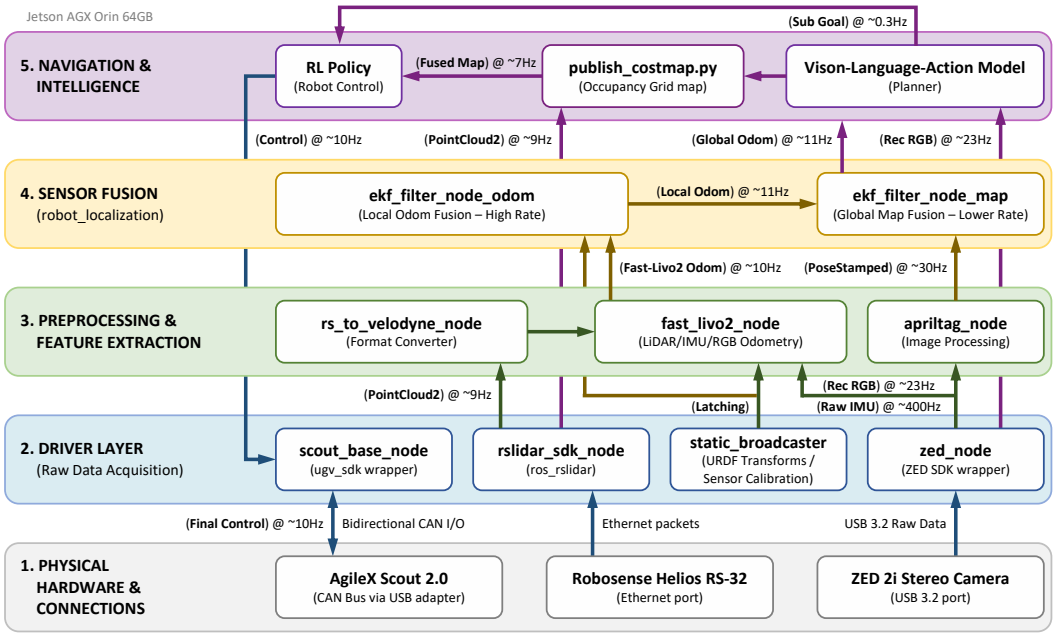



Figure 14: **ROS 2 System Architecture.** The on-device pipeline integrates multimodal data from a skid-steer robot, a 32-channel spinning LiDAR, and a stereo RGB-D camera. For precise localization and efficient resource allocation, we employ Fast-LIVO2 (Zheng et al., 2024a), which leverages LiDAR, RGB, and IMU streams. This is coupled with an Extended Kalman Filter (EKF) that fuses AprilTag (Olson, 2011) detections to establish and maintain a globally fixed coordinate system. The navigation core operates asynchronously, comprising a $\sim$0.3 Hz VLA planner for high-level sub-goal generation and a $\sim$10 Hz RL policy that executes reactive control for obstacle avoidance.

Expanding upon Sec. 4.1 of the main paper, we developed a comprehensive hardware-software bridge on our custom skid-steer robot to successfully transfer the learned policies from simulation to the physical world. As shown in Fig. 14, this architecture is explicitly designed to balance the high-frequency demands of reactive control with the intensive computational loads of visual reasoning. Furthermore, it enables effortless teleoperation and real-time monitoring of the robot through a dedicated WebUI, as illustrated in Fig. 15.

**Software Setup.** To support this architecture, we configured a robust on-device environment, with detailed specifications summarized in Table 11. The core compute engine is an NVIDIA Jetson AGX Orin 64GB running Ubuntu 22.04 and ROS 2 Humble. To maximize edge computing efficiency, the deep learning stack is fully optimized using JetPack 6.2.1 and CUDA 12.6. Within this ROS 2 ecosystem, the system effectively manages the asynchronous execution of multi-rate nodes. It seamlessly handles intensive inference for the 8-billion-parameter NaVILA model and attention-based path deviation detection, while simultaneously executing high-frequency Fast-LIVO2 sensor-fusion odometry and a lightweight RL action policy for reactive control on the AgileX Scout 2.0 mobile base. Furthermore, to facilitate real-time monitoring without degrading the main computational pipeline, we deployed a custom WebUI utilizing a Janus server and Flask. As shown in Fig. 15, this integration ensures low-latency multimedia streaming and seamless teleoperation for the user.

Table 11: **System specifications for physical deployment.** This table outlines the hardware and software stack configured on the NVIDIA Jetson AGX Orin edge computer. It details the hierarchical navigation pipeline, encompassing the 8-billion-parameter NaVILA model for high-level reasoning, the attention-based path deviation detection mechanism (parameterized by window size $W$ and thresholds), and the lightweight 4.89 M parameter RL action policy for reactive local control.

| Category | Component | Specification |
|---|---|---|
| Hardware | Compute | NVIDIA Jetson AGX Orin Dev. 64 GB |
| | Mobile Base | AgileX Scout 2.0 |
| | Camera | ZED 2i (Stereo RGB-D) |
| | LiDAR | RoboSense Helios RS-32 |
| System Software | JetPack | 6.2.1 (L4T R36.4.7) |
| | Kernel | 5.15.148-tegra |
| | OS | Ubuntu 22.04 (aarch64) |
| | CUDA | 12.6 |
| | cuDNN | 9.3.0 |
| | Python | 3.10.12 |
| ROS 2 Stack | ROS 2 Distro | Humble |
| NaVILA Model | Base Model | VILA (LLaMA-3 8B) |
| | Vision Encoder | SigLIP-So400M/patch14/384 |
| | Checkpoint | navila-llama3-8b-8f |
| | Input Resolution | $384 \times 384$ |
| | Inference Frames | 8 (7 history + 1 current) |
| Deviation Detection | Selected Heads ([L, H]) | [21, 12], [16, 1], [14, 1] |
| | Window Size | $W = 10$ steps |
| | Patience Threshold | $P = 9$ steps |
| | Natural Threshold | $\tau = 0.95$ |
| RL Action Policy | Architecture | CNN + MLP |
| | Parameters | 4.89 M |
| | Costmap Input | $1 \times 128 \times 128$ (Occupancy Grid) |
| | Goal Input | $1 \times 2$ ($\Delta x$, $\Delta y$ in local frame) |

**Hardware Setup.** As visualized in Fig. 15, the 32-channel Helios-RS 32 LiDAR is mounted centrally at a height of 0.63,m to maximize ground terrain visibility, while the ZED 2i stereo camera is positioned to provide an unobstructed egocentric view. This hardware configuration precisely aligns with the environment described in Sec. A.3. To rigidly secure these sensors, we fabricated a custom hardware enclosure. Furthermore, to ensure accurate spatial alignment between the sensors, we performed cross-sensor extrinsic calibration using an interactive, targetless calibration tool Beemelmanns (2025); Jeong et al. (2025b). As depicted in Fig. 13, this calibration process yields the following transformation matrix, $T_{\texttt{lidar}}^{\texttt{cam}}$:

$$T_{\texttt{lidar}}^{\texttt{cam}} = \begin{bmatrix} 0.0017 & -0.0349 & 0.9994 & 0.0537 \\ -1.0000 & 0.0017 & 0.0018 & 0.0593 \\ -0.0017 & -0.9994 & -0.0349 & -0.0938 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6}$$

This precise spatial mapping plays a crucial role in deriving highly accurate results across the entire navigation pipeline: RGB-based path planning for the VLA model, LiDAR-based reactive control for the RL policy, and robust LiDAR-RGB-IMU fusion for the Fast-LIVO2 odometry.

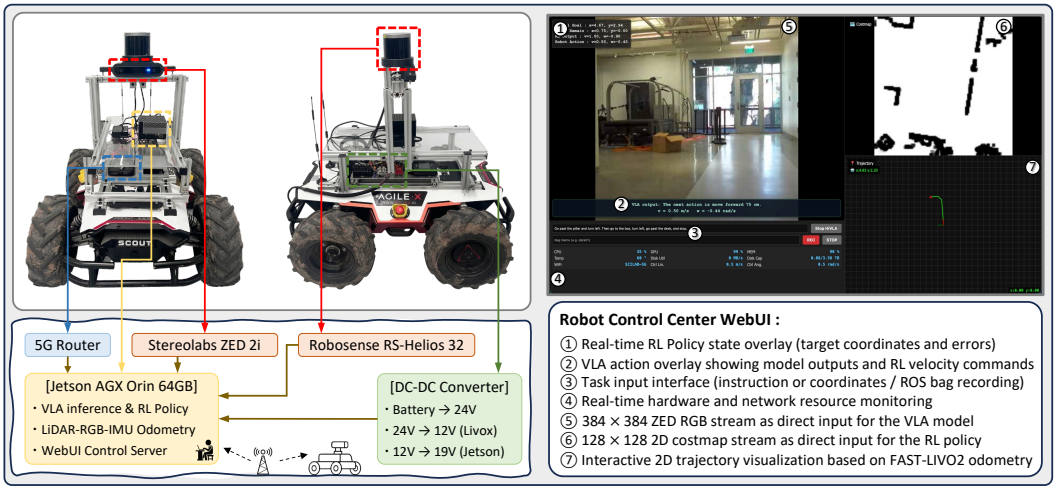



Figure 15: **Physical Robot Setup and Control Center WebUI.** (**Left**) Hardware configuration on the AgileX Scout 2.0 mobile base. A custom enclosure rigidly co-locates the Helios-RS 32 LiDAR and ZED 2i stereo camera. The onboard NVIDIA Jetson AGX Orin processes the VLA inference, RL policy, and Fast-LIVO2 odometry. Power distribution is managed via a dedicated DC-DC converter, while a 5G router ensures low-latency remote connectivity. (**Right**) The custom Robot Control Center WebUI was deployed for teleoperation and monitoring.

### A.3.2 Kinematic Action Smoother

Table 12: **Kinematic Action Smoother Configuration.** Hyperparameters defining the physical constraints and symmetric acceleration limits for the skid-steer robot.

| Parameter | Symbol | Value |
|---|---|---|
| Max Linear Velocity | $v_{\max}$ | 0.5 m/s |
| Max Angular Velocity | $\omega_{\max}$ | 0.5 rad/s |
| Linear Acceleration Limit | $a_{\max}$ | 1.0 m/s$^2$ |
| Angular Acceleration Limit | $a^{\omega}_{\max}$ | 5.0 rad/s$^2$ |
| Deadzone Threshold | $\epsilon$ | 0.02 |

At each control step $t$ operating at 10 Hz ($\Delta t = 0.1$ s), the RL policy outputs target velocity commands $(v_t^{\text{tgt}}, \omega_t^{\text{tgt}})$. Directly applying these discontinuous commands to the physical robot can induce severe mechanical oscillations and physically unsafe accelerations. To ensure stable and safe locomotion, we implemented a three-stage Kinematic Action Smoother pipeline, utilizing the physical constraints and hyperparameters detailed in Table 12.

(i) **Velocity Saturation:** The raw target velocities are first strictly bounded to the physical limits of the robot chassis to prevent hardware overload:

$$\hat{v}_t = \text{clip}(v_t^{\text{tgt}}, -v_{\max}, v_{\max}), \quad \hat{\omega}_t = \text{clip}(\omega_t^{\text{tgt}}, -\omega_{\max}, \omega_{\max}) \tag{7}$$

(ii) **Symmetric Acceleration Clamping:** We compute the instantaneous acceleration over the elapsed time $\Delta t$. To prevent jerky starts and stops, we enforce symmetric linear ($a_{\max}$) and angular ($a^{\omega}_{\max}$) acceleration limits. At the 10 Hz control frequency, this configuration allows the robot to safely decelerate from the maximum linear velocity ($v_{\max}$) to a complete stop in exactly 5 steps (0.5 s) and execute a full angular direction reversal in 2 steps (0.2 s):

$$a_t = \frac{\hat{v}_t - v_{t-1}}{\Delta t}, \qquad \bar{v}_t = v_{t-1} + \text{clip}(a_t, -a_{\max}, a_{\max}) \cdot \Delta t \tag{8}$$

$$a_t^{\omega} = \frac{\hat{\omega}_t - \omega_{t-1}}{\Delta t}, \qquad \bar{\omega}_t = \omega_{t-1} + \text{clip}(a_t^{\omega}, -a^{\omega}_{\max}, a^{\omega}_{\max}) \cdot \Delta t \tag{9}$$

(iii) **Deadzone Suppression:** Finally, sub-threshold commands are explicitly zeroed to suppress motor chatter and prevent mechanical creep when the robot is expected to remain stationary:

$$v_t = \begin{cases} 0 & \text{if } |\bar{v}_t| < \epsilon \\ \bar{v}_t & \text{otherwise} \end{cases}, \quad \omega_t = \begin{cases} 0 & \text{if } |\bar{\omega}_t| < \epsilon \\ \bar{\omega}_t & \text{otherwise} \end{cases} \tag{10}$$

The finalized, mechanically safe velocities $(v_t, \omega_t)$ are then published to the motor drivers and cached as the previous state for the subsequent cycle.

### A.3.3 State Estimation Comparison

Table 13: **State Estimation Resource Overhead.** Additional system resources consumed by the localization nodes evaluated on the NVIDIA Jetson AGX Orin 64 GB.

| Odometry | CPU Usage (%) | GPU Usage (%) | Memory (MB) |
|---|---|---|---|
| ZED VIO | +19.4 | +42.6 | +577.3 |
| Fast-LIVO2 | +26.0 | +13.3 | +367.3 |

As discussed in Sec. 3.4 and 3.5 of the main paper, our vehicle navigation framework requires precise and continuous odometry for two critical operations: spatial trajectory recording for failure recovery and ego-motion compensation for the reactive costmap. While the StereoLabs ZED 2i provides a built-in VIO solution, its reliance on the GPU forces it to share resources with the compute-intensive VLA model, inevitably degrading inference speed. To resolve this architectural bottleneck, we adopted Fast-LIVO2. As detailed in Table 13, Fast-LIVO2 offloads the odometry workload to the CPU, consuming only +13.3% GPU usage compared to ZED VIO's +42.6%. This hardware-aware distribution directly protects real-time performance. As demonstrated in Table 14, integrating ZED VIO increases the VLA inference time to 4.584 s, whereas Fast-LIVO2 maintains a much faster latency of 3.971 s. Ultimately, our fully integrated system—comprising NaVILA, the attention-based path deviation detection, Fast-LIVO2, and the lightweight RL policy—incurs a marginal latency overhead of only $\sim$0.55 s compared to running standalone NaVILA (4.022 s vs. 3.488 s).

Beyond computational efficiency, Fast-LIVO2 provides the superior state estimation accuracy paramount for robust autonomous driving. By fusing multimodal data (LiDAR, RGB, IMU) from the onboard sensors, it overcomes the inherent limitations of pure vision-based odometry. The quantitative evaluations in Table 15 reveal that Fast-LIVO2 significantly outperforms ZED VIO across both indoor and outdoor environments. Notably, in challenging outdoor scenarios, Fast-LIVO2 achieves a median ground truth pose error of just 0.375 m and a yaw error of 6.690°, substantially lower than ZED VIO's 0.839 m and 8.560°. This performance gap is visually corroborated by the trajectory analysis in Fig. 16. While ZED VIO suffers from severe scale and rotational drift—particularly in feature-sparse outdoor settings (Fig. 16c, d)—Fast-LIVO2 successfully maintains a globally

Table 14: **VLA Inference Time Comparison.** Evaluated over 100 runs each on the NVIDIA Jetson AGX Orin. As verified in Sec. A.2.4, the lightweight RL policy does not affect the VLA's inference speed. While integrating state estimation inevitably adds overhead, Fast-LIVO2 maintains a lower latency than ZED VIO by effectively leveraging the CPU, as demonstrated in Table 2. Consequently, **Ours** (NaVILA + path deviation detection + Fast-LIVO2 + RL) requires only lightweight scalar operations from the attention heads, introducing a negligible additional overhead of $\sim$0.05 s. Ultimately, with only $\sim$0.55 s of total additional computation compared to standalone NaVILA, our system successfully executes primary VLA reasoning alongside trajectory recording, path deviation monitoring, and obstacle avoidance.

| Configuration | Mean (s) | Median (s) | Min / Max (s) | Stdev (s) |
|---|---|---|---|---|
| NaVILA | 3.488 | 3.489 | 3.461 / 3.509 | 0.008 |
| NaVILA + RL | 3.488 | 3.489 | 3.461 / 3.509 | 0.008 |
| NaVILA + ZED VIO + RL | 4.584 | 4.588 | 4.525 / 4.645 | 0.025 |
| NaVILA + Fast-LIVO2 + RL | 3.971 | 3.968 | 3.918 / 4.046 | 0.024 |
| **Ours** | 4.022 | 4.020 | 3.910 / 4.120 | 0.041 |

Table 15: **State Estimation Performance Comparison.** Fast-LIVO2 vs. ZED VIO Odometry evaluated over 10 indoor and 10 outdoor runs. The position and yaw errors were measured by computing the accumulated drift relative to a fixed AprilTag before and after each run. The best (lowest) error in each metric is highlighted in bold.

| Environment | Method | GT Pose Error (m) | | GT Yaw Error (°) | | Static Drift (cm) | |
|---|---|---|---|---|---|---|---|
| | | Median | Mean | Median | Mean | Median | Mean |
| **Indoor** | Fast-LIVO2 | **0.082** | **0.331** | **0.640** | **4.463** | 0.080 | 0.080 |
| | ZED VIO | 0.276 | 0.413 | 1.345 | 5.850 | **0.025** | **0.053** |
| **Outdoor** | Fast-LIVO2 | **0.375** | **0.731** | **6.690** | **13.469** | **0.065** | 0.166 |
| | ZED VIO | 0.839 | 4.589 | 8.560 | 22.314 | 0.070 | **0.124** |

consistent trajectory and properly closes the loop. Consequently, replacing the default VIO with Fast-LIVO2 not only preserves the VLA model's computational bandwidth but also guarantees the LiDAR-enhanced precision necessary for reliable spatial memory and safe reactive control.

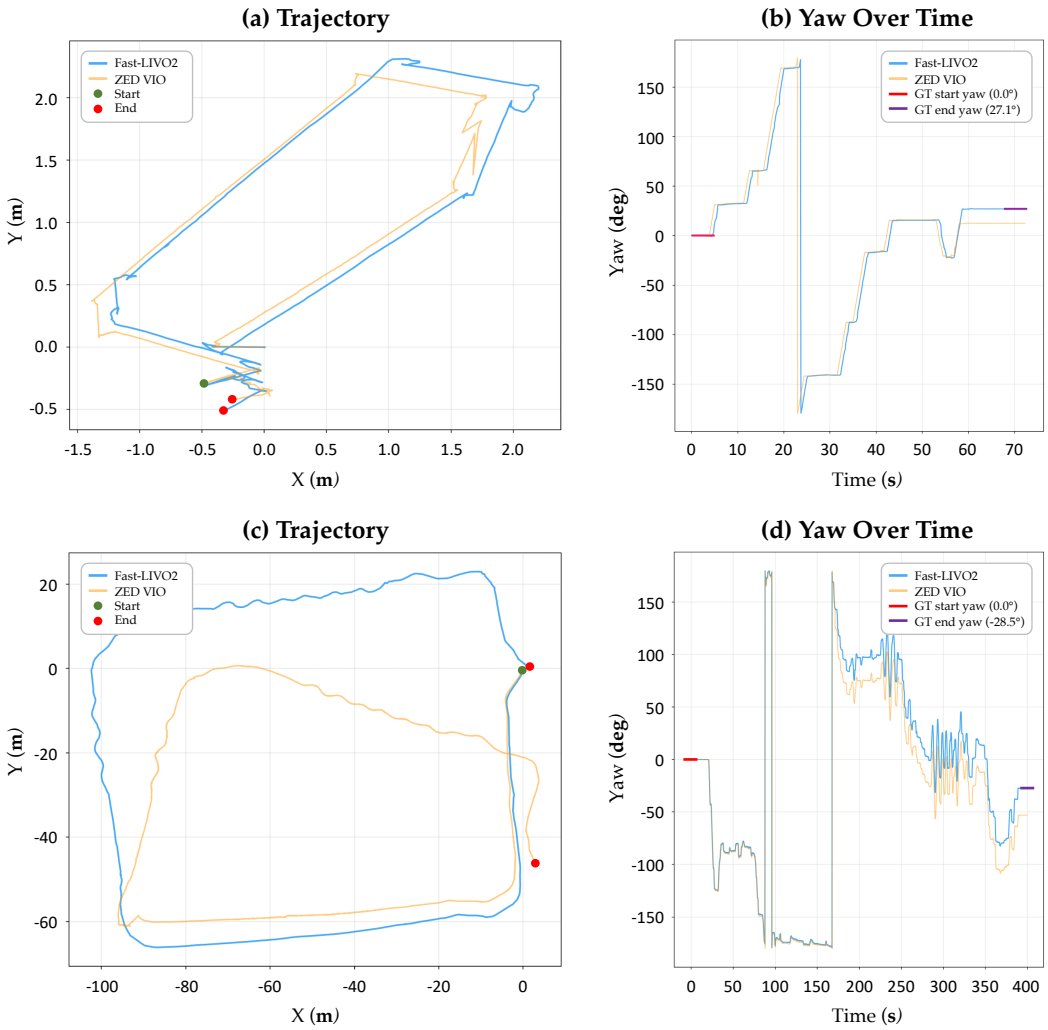



Figure 16: **Trajectory and Yaw Estimation Analysis.** Representative runs comparing **Fast-LIVO2** and **ZED VIO** in (a, b) indoor and (c, d) outdoor environments. The robot starts at the green dot and ends at the red dot. **Fast-LIVO2** successfully closes the loop in both scenarios, whereas **ZED VIO** suffers from severe scale and rotational drift, particularly in the outdoor environment.

### A.4 Real-World Deployment Results

This section presents real-world testing of our fully integrated framework, combining the VLA backbone with path deviation detection, a rollback mechanism to the saved normal trajectory, and a low-level RL avoidance policy. We highlight two key capabilities. First, Fig. 17 demonstrates robust, real-time obstacle avoidance despite naive VLA commands. Second, Fig. 18 illustrates autonomous trajectory recovery to the last normal state following a path deviation.

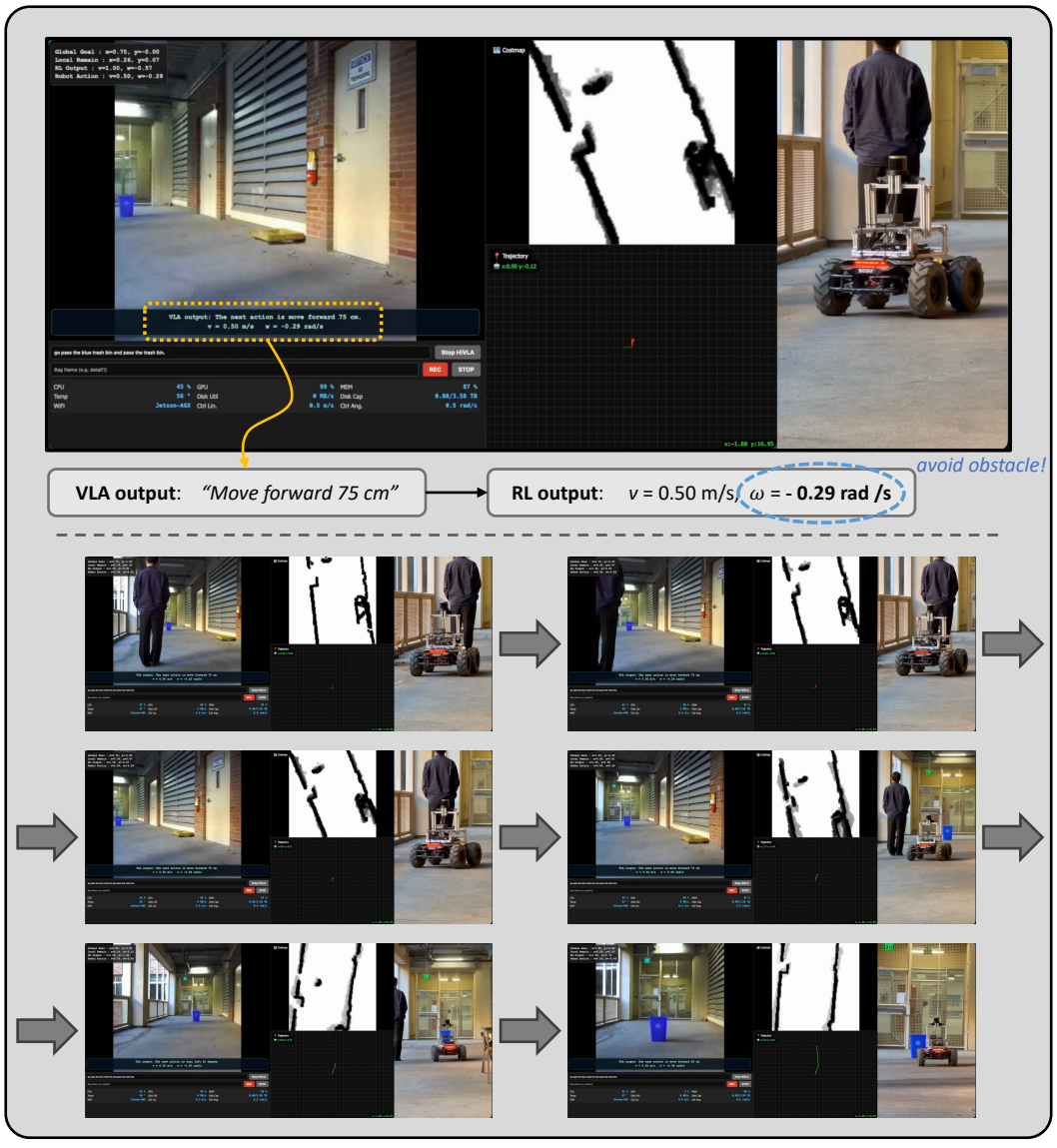



Figure 17: **Robust Obstacle Avoidance via Low-Level RL Policy.** This sequence demonstrates the system's ability to correct unsafe high-level commands in real-time. Although the VLA model naively outputs a forward movement command ("Move forward 75 cm") despite a pedestrian blocking the path, our RL policy independently detects the obstacle. It immediately overrides the naive command by executing a safe avoidance maneuver (v=0.50 m/s, ω=0.29 rad/s), allowing the robot to steer around the pedestrian without collision and continue its navigation.

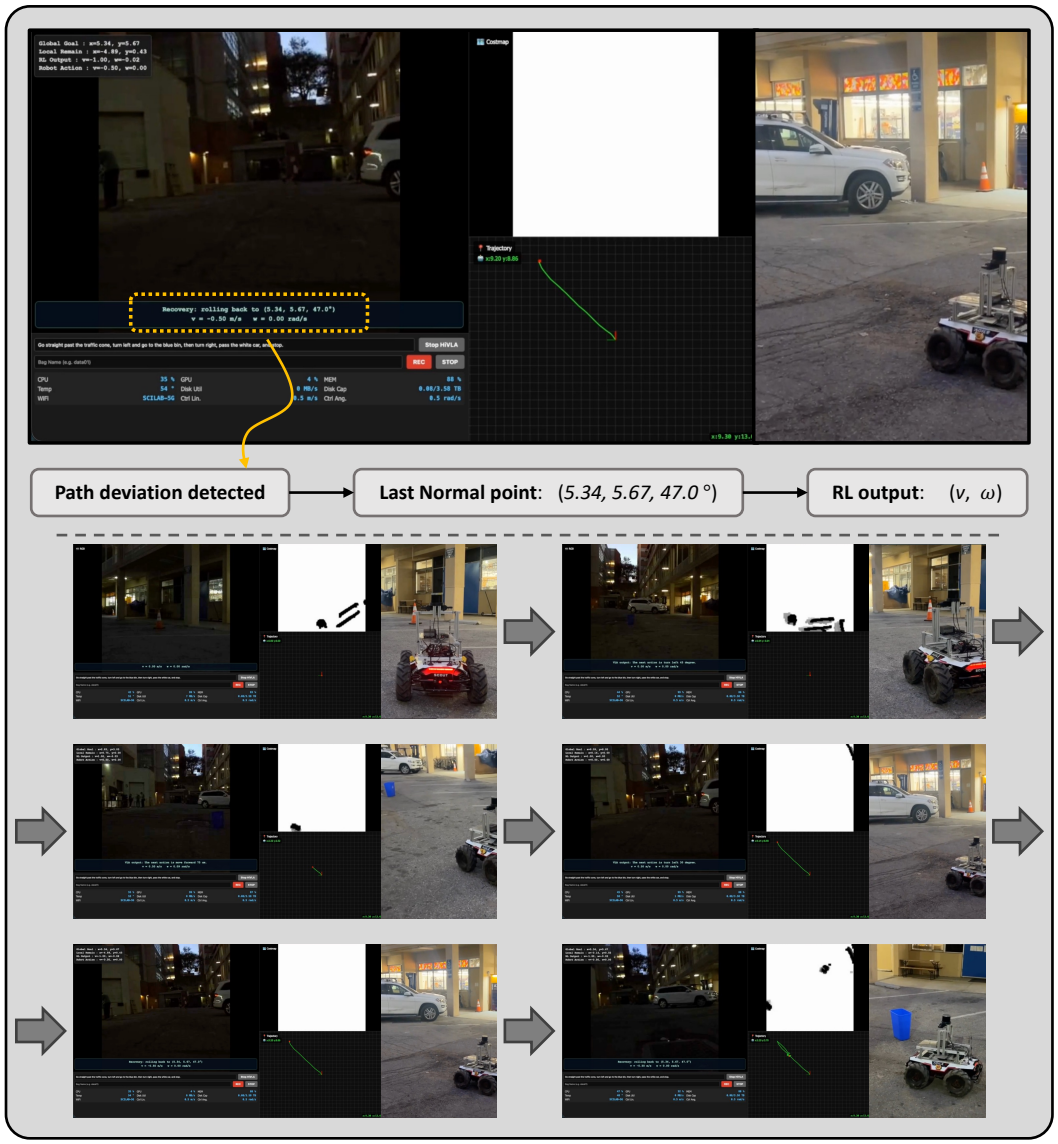



Figure 18: **Path Deviation Detection and Autonomous Trajectory Recovery.** This sequence demonstrates the rollback mechanism. When a path deviation caused by VLA hallucination is detected during navigation, the system aborts the erroneous trajectory. It then initiates a recovery phase toward the last normal state (e.g., $x = 5.34, y = 5.67, z = 47.0°$) from the real-time recorded trajectory. During this process, the robot automatically executes a direct safe recovery to that point, guided by the obstacle avoidance actions of the RL policy, ensuring the mission resumes smoothly.