

A Cross-Embodiment Navigation Stack for Heterogeneous Robot Teams: Planner, Interface Calibration, and Future-Range Perception

Zhaoyu Li • Jingyao Sun • Yichen Wang • Bowen Huang • Tianyu Zhou

Legged Locomotion Group, Tsinghua University, Beijing, China

PUBLISHED

October 5, 2026

REGISTERED

October 6, 2026

RECORD TYPE

Preprint

ABSTRACT

Running one navigation stack on a heterogeneous robot team requires more than a planner that accepts different footprints: each robot's velocity interface drifts with payload and battery state, and range perception must be queried along the motions that the robot can actually execute. This paper presents a cross-embodiment navigation stack with three coupled layers that share a single data contract keyed on a versioned configuration descriptor. A configuration-driven local planner built on AgniNav plans over abstract robot configurations; an active interface calibration layer built on GAUGE keeps each robot's command-to-motion model current through planner-conditioned probes triggered by residual drift; and a control-aligned future-range layer built on FutureRay supplies free-range predictions along the directions each embodiment can execute under its calibrated interface. The stack was deployed on a mixed team of a Go2-class quadruped, a B2-class quadruped, and a wheeled robot on a three-floor indoor-adjacent course with doorways, a 12 cm step, and a light-degraded stairwell, during a 14 h campaign and short ablation runs. The full stack reached 88.2 percent end-to-end task success with a 28 ms p95 replanning latency. Removing any single layer lowered success by 3.0 to 10.6 percentage points, although the two recorded ablation summaries disagree on which layer matters most. A simulated humanoid embodiment was onboarded in 2.1 h and succeeded in 94 percent of 10 runs. The paper also characterizes the failure envelope of the stack.

INDEX TERMS cross-embodiment navigation, legged robots, heterogeneous robot teams, local planning, velocity interface calibration, future-range perception

KAGG NOTE Cross-embodiment navigation is a data-contract problem: the planner, the interface model, and the range perception must agree on one versioned configuration descriptor, and each must be kept current while the robots run.

AUTHOR AFFILIATIONS

Zhaoyu Li, Jingyao Sun, Yichen Wang, Bowen Huang: Legged Locomotion Group, Tsinghua University, Beijing, China.
Tianyu Zhou: Eastern Institute of Technology, Ningbo, Zhejiang, China.

01 Introduction

Robot teams deployed in buildings, campuses, and industrial sites are rarely homogeneous. A typical team pairs agile quadrupeds, which handle steps and stairs, with wheeled platforms, which carry payload efficiently on flat floors. Operating such a team with one navigation stack is attractive because it concentrates engineering effort, keeps behavior consistent across robots, and makes the addition of a new platform a configuration task rather than a software project. Cross-embodiment navigation research has pursued this goal along two lines: generalist policies trained on data from many robots [4, 5, 6, 8], and explicit abstractions that expose the embodiment to a shared planner [9, 1].

Configuration-driven planning, exemplified by AgniNav [1], addresses the geometric part of the problem: a single local plan-

ner is parameterized by an abstract robot configuration (body envelope, sensors, velocity limits), so the same code serves bodies of different size and shape. Field deployment, however, exposes two further gaps that a configuration of static parameters does not close. First, legged robots expose an *opaque velocity interface*: the commanded twist is tracked by a learned or proprietary locomotion controller whose gain, bias, and lag change with payload, battery state, gait, and floor surface [2, 26, 25]. A planner that rolls out candidate commands through a stale interface model predicts motions the robot does not execute. Second, range perception is usually queried along the body heading or over a full scan of the *present* scene, whereas collision risk depends on the range along the directions the robot will actually move under the commands it can execute in the next second [3]. Each of these gaps has been addressed in isolation: GAUGE calibrates opaque quadruped interfaces with planner-conditioned probes [2], and FutureRay predicts control-aligned future range [3]. What has not been reported is how the three pieces behave when they are joined into one stack that runs a mixed team for a full operating day.

The difficulty in joining them is not algorithmic novelty in any single layer but consistency between layers. The planner, the calibration layer, and the perception layer each hold a model of the same robot; if these models disagree, for example be-

cause the interface was recalibrated after a payload change but the perception layer still aligns its queries with the old model, the stack plans with inconsistent assumptions. This paper therefore organizes the stack around a single data contract: a versioned *configuration descriptor* that is the join key between all three layers.

The contributions of this paper are as follows.

1. A cross-embodiment navigation stack that composes a configuration-driven local planner (AgniNav backbone), an active interface calibration layer (GAUGE layer), and a control-aligned future-range layer (FutureRay layer) through one versioned configuration descriptor, including the version-consistent join rule that prevents layers from mixing stale and current robot models.
2. A concrete formulation of the coupling: a per-axis interface model estimated online with recursive least squares and a CUSUM drift trigger, probe selection restricted to commands the planner already deems safe, and a speed governor that converts the conservative future-range envelope into a braking-distance constraint on each candidate command.
3. A deployment on a mixed team of two quadrupeds (Go2-class and B2-class) and one wheeled robot on a three-floor indoor-adjacent course over a 14 h campaign, with end-to-end task success, interface drift, onboarding time, and re-planning latency, together with per-layer ablations.
4. An analysis of the failure envelope of the stack, covering a light-degraded stairwell, a payload- and battery-induced interface drift, and the onboarding of a simulated humanoid embodiment, and an explicit account of where the recorded evidence is inconclusive.

02 Related Work

2.1 Cross-embodiment navigation policies and abstractions

Generalist navigation models treat embodiment diversity as a data-sharing problem. GNM trains a goal-conditioned policy on trajectories from several wheeled and legged robots using a normalized action space and temporal context [4]; VINT scales this recipe with a Transformer backbone and adapts it to new task specifications [5]; NoMaD unifies goal reaching and exploration with a diffusion action decoder [6]. RING randomizes body size, rotation pivot, and camera placement in simulation to obtain an indoor navigation generalist [7], and NavFoM trains one model across quadrupeds, drones, wheeled robots, and vehicles with embodiment and camera identifiers [8]. EA-Nav conditions an imitation policy on embodiment geometry and adds risk-aware correction on augmented high-risk data [10]. Beyond navigation, Open X-Embodiment [11], Octo [12], and CrossFormer [13] show that a single policy can be trained across manipulators, mobile bases, and legged platforms. These approaches share knowledge through learned representations; the stack in this paper instead shares knowledge through an explicit descriptor, which makes the quantities that determine safety (envelope, limits, interface response) inspectable and updatable at run time. The two views are compatible: a generalist policy could serve as a global or semantic layer above the stack described here.

CeRLP converts heterogeneous visual observations into a unified geometric representation, including scale-corrected depth and a height-adaptive scan abstraction, before local plan-

ning [9]. AgniNav [1] pushes this geometric route further and is the planning backbone of the present stack. AgniNav plans over an abstract robot configuration that collects body geometry (a collision envelope described by front length, rear length, and half width, plus a collision-relevant height), sensor placement, and velocity limits. The collision-relevant height conditions a monocular image-to-scan predictor, while the planar envelope parameters configure collision checking in a sampling-based local planner, so that one planning stack runs unchanged on robots of different size. The present work keeps AgniNav’s configuration abstraction but makes two changes. First, the configuration is no longer a static file: it is a versioned record whose interface fields are rewritten online by the calibration layer. Second, AgniNav assumes that a commanded velocity is realized within the declared limits; here, rollouts pass through an explicitly calibrated command-to-motion model, which is necessary when the same nominal command produces different motion on a loaded and an unloaded quadruped.

2.2 Local planning and legged navigation systems

The dynamic window approach selects velocity commands from the set reachable within one control interval and scores them by heading, clearance, and speed [14]; it remains the template for most velocity-space local planners, including the controllers distributed with the ROS 2 navigation system [15]. The planner layer of this stack is a dynamic-window planner in this sense, but its window and rollouts are derived from the descriptor rather than tuned per robot. For legged robots, ArtPlanner combines sampling-based planning with learned reachability for robust field navigation [16], and Kim et al. learn a forward dynamics model of a quadruped’s velocity tracking and use it to bias trajectory sampling toward safe commands [17]. The latter is the closest antecedent of the interface-aware rollouts used here; the difference is that the model in this stack is a small, explicitly parameterized model maintained online by active probing, rather than a learned network trained offline. End-to-end alternatives learn locomotion and local navigation jointly [18, 19]; they reach high agility but tie the navigation behavior to one body, which is the coupling a cross-embodiment stack is designed to avoid. Team-scale field systems from the DARPA Subterranean Challenge, notably CERBERUS [40] and NeBula [41], demonstrated heterogeneous legged, wheeled, and aerial teams under degraded perception; they also illustrate that much of the integration cost of such teams lies in per-platform tuning, which motivates measuring the time required to add an embodiment.

2.3 Velocity interfaces and their calibration

Modern quadrupeds and humanoids are commanded through a velocity interface realized by a learned locomotion policy [20, 21, 22, 23, 27]. Such policies adapt internally to terrain and payload [24], estimate body velocity concurrently with control [25], and may expose behavior parameters that change tracking characteristics [26]. From the planner’s point of view the interface is therefore opaque and nonstationary: the mapping from commanded to realized twist is neither documented nor constant. GAUGE [2] addresses exactly this setting. It treats the quadruped velocity controller as a black box, issues short probe commands that are conditioned on the planner’s current needs (probes are drawn from commands the planner is likely to issue and that are safe in the current scene), and fits a calibrated model of the command-to-motion response online. The

calibration layer of the present stack adopts GAUGE’s planner-conditioned probing, adds an explicit drift trigger so that probes are issued only when tracking residuals indicate a change, and writes the result back into the shared descriptor so that the planner and the perception layer consume the same model. It also applies the same procedure to a wheeled robot, for which the interface is less opaque but still drifts with load. Observability-aware trajectory design for self-calibration [28] and sequential change detection [29] provide the classical tools on which these choices rest.

2.4 Perception for agile and degraded-condition navigation

Perceptive locomotion work has shown that egocentric depth or elevation input lets legged robots negotiate steps, stairs, and gaps [23, 31, 32, 33]. For navigation, self-supervised traversability estimation adapts quickly to new environments [34], BADGR learns navigational affordances from experience [35], and high-speed flight work demonstrates that perception must anticipate where the vehicle will be rather than where it is [36]. **FutureRay** [3] formulates this anticipation for quadrupeds: it predicts free range along the robot’s own predicted, control-aligned motion directions over a short horizon, so that obstacle queries correspond to what the locomotion controller can actually execute, rather than to the present body heading or a uniform scan. The present stack uses FutureRay’s query structure with two changes: the directions are obtained by rolling candidate commands through the *calibrated* interface model supplied by the calibration layer, and the prediction carries an uncertainty estimate [37] that is converted into a conservative range envelope consumed by a braking-distance speed governor. This connects FutureRay to the safety-filter view of collision avoidance [38] and to recent work on adaptive safety margins for visual navigation [39].

2.5 Evaluation of embodied navigation

Anderson et al. recommend reporting success together with path efficiency and explicit success criteria [43], and Kadian et al. show that simulator rankings can fail to predict real-world rankings [42]. Physics simulators such as Isaac Gym [30] make it cheap to instantiate new bodies, which is useful for exercising an onboarding procedure but does not substitute for physical validation. Accordingly, the evaluation below separates physical-team results from the simulated humanoid onboarding experiment.

03 Problem Formulation

3.1 Team, descriptor, and interface

A team consists of robots $i \in \mathcal{R} = \{1, \dots, N\}$. Each robot is described by a versioned configuration descriptor

$$\kappa_i^{(v)} = (\mathcal{G}_i, \mathcal{S}_i, \mathcal{L}_i, \Theta_i^{(v)}), \quad (1)$$

where $v \in \mathbb{N}$ is the descriptor version. The geometry block $\mathcal{G}_i = (l_f, l_r, w, h, h_s)$ contains front length l_f , rear length l_r , and half width w of the planar collision envelope, the collision-relevant height h , and the maximum traversable step height h_s . The sensor block $\mathcal{S}_i$ lists each range-producing sensor with its modality, extrinsic transform, and rate. The limit block $\mathcal{L}_i = (\bar{v}_x, \bar{v}_y, \bar{\omega}, \bar{a}, t_d)$ contains the longitudinal, lateral, and yaw-rate command limits, a braking deceleration bound $\bar{a}$, and the end-to-end actuation latency t_d . The interface block $\Theta_i^{(v)}$ holds the parameters of the command-to-motion model defined be-

low; it is the only block that changes during operation, and every change produces a new version v .

At control step k (period Δt), robot i receives a command $u_k = (u_x, u_y, u_\omega)_k \in \mathcal{U}_i$, where $\mathcal{U}_i$ is the box defined by $\mathcal{L}_i$ (with $\bar{v}_y = 0$ for the wheeled robot). The realized body twist $\xi_k = (\xi_x, \xi_y, \xi_\omega)_k$ is produced by an opaque controller,

$$\xi_{k+1} = F_i(\xi_k, u_{k-d_i}, z_k) + \epsilon_k, \quad (2)$$

where d_i is the command delay in control steps, z_k collects unobserved conditions (payload, battery state, gait, floor), and ϵ_k is tracking noise. The stack does not have access to F_i ; it maintains a parametric approximation $\hat{F}(\cdot; \Theta_i^{(v)})$.

3.2 Navigation task and metrics

The environment is represented by a multi-floor route graph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ whose edges carry traversal requirements (minimum clear width, step height, stairs), and by the onboard range observations o_k . A task assigns robot i a goal node $g \in \mathcal{V}$. Robot i must reach g while keeping its envelope $\mathcal{B}_i(x)$ at pose x clear of obstacles,

$$\text{dist}(\mathcal{B}_i(x_k), \mathcal{O}_k) \geq m_i \quad \forall k, \quad (3)$$

with $\mathcal{O}_k$ the obstacle set and m_i a clearance margin. A task is a success if the goal is reached without collision, operator intervention, or timeout. Four metrics are reported. End-to-end task success is the fraction of tasks that succeed. Interface drift for robot i is the mean relative change of the identified steady-state tracking gains,

$$\delta_i(t) = \frac{1}{|J_i|} \sum_{j \in J_i} \frac{|k_{i,j}(t) - k_{i,j}(t_0)|}{k_{i,j}(t_0)}, \quad (4)$$

where J_i is the set of commanded axes, $k_{i,j}$ is the gain of axis j (defined in (5)), and t_0 is the start of the campaign; the campaign figure is reported per operating day and averaged over the team. Onboarding time is the wall-clock time required to produce a working descriptor and calibration probe plan for a new embodiment. Repanning latency is the time from receipt of the latest perception and descriptor snapshot to publication of the selected command sequence; its 95th percentile (p95) is reported.

The design problem is to choose planner, calibration, and perception layers that maximize task success across $\mathcal{R}$ subject to (3), using one implementation for all robots and adding a new robot by writing a new descriptor rather than new code.

04 Method

The stack consists of three layers that run at different rates and exchange data only through descriptor-stamped messages. The interface is commanded at a 20 ms control period (Δt), the planner runs at 10 Hz, and calibration runs asynchronously, issuing probes only when triggered. Table 1 summarizes what each layer reads and writes.

4.1 Configuration descriptor as the join key

Each layer subscribes to the descriptor store and caches the latest version per robot. The join rule is that a planning cycle for robot i may only combine an interface model and a range prediction that carry the same version v . When the calibration layer publishes $\Theta_i^{(v+1)}$, the perception layer recomputes its queries under the new model in the next cycle, and the planner discards

Table 1: Data contract between the three layers. Every message carries the robot index i and descriptor version v .

Layer	Reads	Writes	Rate
Planner (AgniNav backbone)	$\mathcal{G}_i, \mathcal{L}_i, \Theta_i^{(v)}$; conservative range envelope	Command sequence stamped (i, v) ; candidate set for probing	10 Hz
Calibration (GAUGE layer)	Commands and realized twists; planner candidate set; range envelope	New version $\Theta_i^{(v+1)}$	Asynchronous, drift-triggered
Future range (FutureRay layer)	$\mathcal{S}_i, \mathcal{G}_i, \Theta_i^{(v)}$; range observations; candidate commands	Range mean and uncertainty per query, stamped (i, v)	10 Hz, with the planner

any range prediction still stamped v . This rule costs at most one planning cycle of delay after each recalibration, and it removes a class of silent errors in which perception is aligned with one motion model and the planner rolls out another. Configuration sharing means that all robots are served by the same planner and perception code instantiated from their descriptors; there are no per-robot planner parameters outside the descriptor. The route graph $\mathcal{H}$ is shared as well: an edge is admissible for robot i only if its requirements are compatible with $\mathcal{G}_i$ (for example, its step height does not exceed h_s), so each robot receives a route compatible with its body.

4.2 Interface model and online estimation

Each commanded axis j of robot i is modeled as a first-order lag with gain, bias, and delay,

$$\begin{aligned} \xi_{j,k+1} &= a_j \xi_{j,k} + (1 - a_j) (k_j u_{j,k-d} + b_j), \\ a_j &= e^{-\Delta t / \tau_j}, \end{aligned} \quad (5)$$

with steady-state gain k_j , bias b_j , time constant τ_j , and delay d steps, so that $\Theta_i = \{(k_j, b_j, \tau_j)\}_{j \in J_i} \cup \{d\}$. The model is deliberately small: a diagonal first-order model captures the dominant effects of payload (reduced gain and slower response) and battery sag (reduced gain at high command) on the axes the planner uses, and it is identifiable from a few seconds of excitation. Writing $\alpha_j = a_j$, $\beta_j = (1 - a_j)k_j$, and $\gamma_j = (1 - a_j)b_j$, (5) is linear in $\eta_j = (\alpha_j, \beta_j, \gamma_j)$ with regressor $\varphi_{j,k} = (\xi_{j,k}, u_{j,k-d}, 1)$. The parameters are tracked with recursive least squares with forgetting factor $\lambda \in (0, 1]$,

$$\begin{aligned} K_k &= \frac{P_{k-1} \varphi_k}{\lambda + \varphi_k^\top P_{k-1} \varphi_k}, \\ \hat{\eta}_k &= \hat{\eta}_{k-1} + K_k (\xi_{k+1} - \varphi_k^\top \hat{\eta}_{k-1}), \\ P_k &= \frac{1}{\lambda} (P_{k-1} - K_k \varphi_k^\top P_{k-1}), \end{aligned} \quad (6)$$

from which $k_j = \beta_j / (1 - \alpha_j)$, $b_j = \gamma_j / (1 - \alpha_j)$, and $\tau_j = -\Delta t / \log \alpha_j$ are recovered. The delay d is selected from a small candidate set by minimum prediction residual. Passive estimation from ordinary driving is poorly conditioned, because during long straight segments the commands are nearly constant and P_k grows along unexcited directions; this is the reason for the active probing below.

4.3 Drift trigger and planner-conditioned probes

The calibration layer monitors the normalized one-step prediction residual $e_k = |\xi_{k+1} - \varphi_k^\top \hat{\eta}_{k-1}| / \hat{\sigma}_e$, where $\hat{\sigma}_e$ is the residual scale estimated at the last convergence, and accumulates a one-sided CUSUM statistic [29],

$$\begin{aligned} S_k &= \max(0, S_{k-1} + e_k - \nu), \\ \text{trigger if } S_k &> h_c, \end{aligned} \quad (7)$$

with reference value ν and threshold h_c . A trigger opens a probe episode. Following GAUGE [2], probes are planner-conditioned: they are drawn from the planner's current candidate set $\hat{\mathcal{U}}_i$, restricted to commands whose conservative clearance $c^-(u) = \min_\tau D_\tau(u)$, defined in Section 4.5, is at least a probe margin $m_p > 0$ and whose loss of planned progress $\Delta_{\text{prog}}(u)$ is at most ε . Among these, a short probe sequence Π is chosen greedily to maximize the information gain on the parameters,

$$\begin{aligned} \Pi^* &= \arg \max_{\Pi \subseteq \hat{\mathcal{U}}_i} \log \det \left(P^{-1} + \sum_{u \in \Pi} \varphi(u) \varphi(u)^\top \right) \\ \text{s.t. } c^-(u) &\geq m_p, \Delta_{\text{prog}}(u) \leq \varepsilon, \end{aligned} \quad (8)$$

a D-optimal design in the sense of observability-aware calibration [28]. The episode ends when the parameter covariance of the gains falls below a threshold and the residual statistic has reset, at which point the layer publishes $\Theta_i^{(v+1)}$. Restricting probes to the planner's candidate set has two consequences that motivate the design: probes excite the part of the command space the planner actually uses, so the model is accurate where it matters, and probes never require a separate safety check, because they inherit the clearance test of the planner.

4.4 Control-aligned future range

For a candidate command sequence $u_{0:H}$ the planner rolls the calibrated model (5) forward to obtain predicted poses p_τ and realized twists ξ_τ at query times $\tau \in \{\tau_1, \dots, \tau_M\}$ within the horizon. The query direction at time τ is the direction of predicted travel,

$$\psi_\tau = \psi_\tau^{\text{body}} + \text{atan2}(\xi_{y,\tau}, \xi_{x,\tau}), \quad (9)$$

which differs from the body heading whenever the robot moves laterally or turns, as quadrupeds routinely do. Following FutureRay [3], a prediction head estimates, from the recent range observations of the sensors in $\mathcal{S}_i$, the free range from p_τ along ψ_τ . The head outputs a mean μ and a standard deviation σ [37], and the stack uses the conservative envelope

$$r^-(u, \tau) = \mu(u, \tau) - \beta \sigma(u, \tau), \quad (10)$$

with confidence multiplier β ; the envelope width is $2\beta\sigma$. When perception degrades, for example under low illumination, σ grows, the envelope widens, and the downstream constraint becomes more conservative without any mode switch.

4.5 Speed governor and planner

The range that the robot body occupies along direction ψ relative to its center is $\ell_i(\psi)$, computed from the envelope (l_f, l_r, w) in $\mathcal{G}_i$. A command is admissible if, at every query time, the robot can stop within the conservative free range,

$$r^-(u, \tau) - \ell_i(\psi_\tau) - m_i \geq \|\xi_\tau\| t_d + \frac{\|\xi_\tau\|^2}{2\bar{a}}, \quad (11)$$

which bounds the admissible speed at query τ by

$$v_\tau^{\max} = \bar{a} \left(-t_d + \sqrt{t_d^2 + \frac{2D_\tau}{\bar{a}}} \right), \quad (12)$$

$$D_\tau = r^-(u, \tau) - \ell_i(\psi_\tau) - m_i.$$

Equation (12) is the positive root of (11) written as an equality. It makes the speed reduction in degraded perception explicit: a wider envelope lowers D_τ and therefore the admissible speed. The planner follows the dynamic-window structure [14] with AgniNav’s configuration-driven collision model [1]. Candidates are sampled from the window

$$\mathcal{W}_i = \{u \in \mathcal{U}_i : \|u - u_{\text{prev}}\|_\infty \leq \bar{a} T_p\}, \quad (13)$$

where T_p is the planning period, and each admissible candidate is scored by

$$J(u) = w_g \|p_{\tau_M} - g_{\text{loc}}\| + w_c \sum_{m=1}^M \exp\left(-\frac{D_{\tau_m}}{\rho}\right) + w_s \|u - u_{\text{prev}}\|^2 - w_v \bar{\xi}_x, \quad (14)$$

where g_{loc} is the local goal on the current route edge, ρ a clearance length scale, $\bar{\xi}_x$ the mean predicted forward speed, and w_g, w_c, w_s, w_v nonnegative weights. The first command of $u^* = \arg \min J$ is executed. All terms are computed from the descriptor and the calibrated model, so no weight or limit is tuned per robot outside κ_i . If no candidate is admissible at the lowest speed in the window, the robot stops, marks the current route edge as degraded, and requests a route on $\mathcal{H}$ that avoids it; this is the conservative path-around behavior observed in the stairwell (Section 5.5).

4.6 Adding an embodiment

Adding a robot requires three artifacts: the descriptor blocks $\mathcal{G}$, $\mathcal{S}$, and $\mathcal{L}$, taken from the body model, sensor extrinsics, and the manufacturer’s command limits; an initial probe plan, generated automatically from (8) with the planner’s candidate set in an open area; and an initial interface estimate $\Theta^{(0)}$ obtained by executing that plan. No planner or perception code is changed. The onboarding-time metric measures the wall-clock time to produce these artifacts.

IMPLEMENTATION NOTE

The stack treats every velocity controller as opaque, including the wheeled robot’s. All layers exchange descriptor-stamped messages, and every logged plan records the version tuple (i, v) under which it was computed, which makes it possible to attribute a failure to a specific interface model.

05 Experiments

5.1 Platform and course

The team consisted of two quadrupeds, one Go2-class and one B2-class, and one wheeled robot. The course spanned three

floors of an indoor-adjacent site, organized as 1.9 km loops that included doorways, a 12 cm step, and a light-degraded stairwell. The evaluation comprised a one-day campaign with 14 h of operation and, in addition, 10 short runs per ablation variant. The full stack joined the three layers on the configuration descriptor as described in Section 4. Compared variants removed one layer at a time: the calibration layer was replaced by static interface models; the future-range layer was replaced by present range from the current observations, not aligned with predicted motion; and configuration sharing was replaced by separately configured per-robot planners.

5.2 Main comparison

Table 2 reports the main comparison. The full stack attained 88.2% end-to-end task success, compared with 77.7% without the calibration layer, 82.9% without the future-range layer, and 85.2% without configuration sharing, i.e., differences of 10.5, 5.3, and 3.0 percentage points. It also had the lowest replanning p95 latency (28 ms, against 30 to 31 ms) and the shortest onboarding time (2.1 h, against 2.2 to 2.4 h). All latencies are well within the 100 ms planning period. The latency and onboarding differences are small, and the data do not include a spread across repetitions, so these two columns are best read as showing that the coupled design does not add cost, rather than as evidence of a speed advantage.

Interface drift was 0.8 to 0.9%/day in every variant. This is expected under the definition in (4): drift is a property of the robots’ interfaces under payload and battery changes, not of the navigation stack, and the calibration layer does not reduce drift but tracks it. The relevant effect of calibration is therefore visible in task success rather than in the drift column.

5.3 Ablation

Table 3 reports the success-only ablation summary. It agrees with Table 2 on the full-stack figure (88.2%) and on the qualitative finding that removing any layer lowers success: by 3.2 points without the calibration layer, 10.6 points without the future-range layer, and 7.9 points without configuration sharing.

The two summaries disagree on the size and order of the per-layer effects. In Table 2, removing calibration is the most damaging change (10.5 points) and removing configuration sharing the least (3.0 points); in Table 3, removing future range is the most damaging (10.6 points) and removing calibration the least (3.2 points). The recorded data do not resolve which summary reflects the relative importance of the layers, and the study does not report per-run variance that would allow a significance test. The conclusions drawn here are therefore limited to what both summaries support: the full stack is the best variant in both, and each layer, when removed, costs between 3.0 and 10.6 points of task success. A ranking of the layers is not claimed.

5.4 Interface drift during the campaign

The team-level drift of 0.8%/day masks larger excursions on individual robots. The B2-class quadruped’s interface drifted by 1.9% by hour 10 of the campaign, attributed to the combination of payload and battery depletion. The CUSUM trigger of (7) opened a probe episode, and the calibration layer re-converged the model in 8 min using 3 probes, after which a new descriptor version was published and both the planner and the perception layer switched to it under the join rule. Two observations follow. First, a per-robot excursion of 1.9% within one day is more

Table 2: Main comparison on the mixed team. Drift is the interface drift over the campaign; onboarding is the time to add a new embodiment.

Method	End-to-end task success (%)	Interface drift over campaign (%/day)	Time to add a new embodiment (h)	Replan p95 latency (ms)
Stack without calibration layer (static interface models)	77.7	0.8	2.4	30
Stack without future-range layer (present range)	82.9	0.9	2.2	31
Stack without configuration sharing (per-robot planners)	85.2	0.9	2.4	30
Ours	88.2	0.8	2.1	28

Table 3: Ablation summary: end-to-end task success.

Variant	End-to-end task success (%)
Full stack	88.2
w/o calibration layer	85.0
w/o future-range layer	77.6
w/o configuration sharing	80.3

than twice the team-level daily figure, so an aggregate drift metric understates the risk to the most heavily loaded robot. Second, the episode was short and required few probes, which is the intended effect of drawing probes from the planner’s candidate set: probes are useful motions, not a separate calibration routine.

5.5 Failure cases

Table 4 collects the recorded failure and stress cases; they define the failure envelope discussed in Section 6.

Light-degraded stairwell. At 40 lx the range predictions became markedly less certain, and the conservative envelope of (10) widened by a factor of 3.8. Through (12), the widened envelope reduced the admissible speed, and the team traversed the stairwell at 1.1 m/s with a mean clearance margin of 16 cm. In 2 of 10 runs the team used the conservative path around the stairwell, which in this stack is the route-level fallback taken when no candidate remains admissible. This is the intended degradation mode, in which uncertainty is converted into lower speed and, at the limit, a detour, rather than into collisions. It also shows the cost of this design: a detour lengthens the route, and the stack has no mechanism to recover range certainty in poor light other than slowing down.

Interface drift on the B2-class quadruped. This case is discussed above. Its significance for the failure envelope is that without the calibration layer the planner would have continued to roll out commands through a model whose gains were off by 1.9%. That error is small per step, but it accumulates over the planning horizon and biases the predicted query directions in (9).

Adding a simulated embodiment. A fourth embodiment, an H1-class humanoid asset in simulation, was added by writing its descriptor and generating its calibration probe plan, which took 2.1 h. This is consistent with the onboarding time of the full stack in Table 2. The first successful loop ran at 0.9 m/s, and the humanoid succeeded in 94% of 10 runs. Because this result was obtained in simulation, it shows that the onboarding procedure works without code changes; it does not show that a physical humanoid would perform similarly [42].

06 Discussion: The Failure Envelope

The recorded cases outline three boundaries of the stack’s operating envelope. The first is *perceptual*: the future-range layer converts perception uncertainty into speed through (10) and

(12). This yields graceful degradation as long as the envelope widens monotonically with true uncertainty, but the stack offers no remedy when the envelope becomes so wide that no command is admissible; it can only detour. The stairwell, in which 2 of 10 runs detoured, lies near this boundary. The second boundary is *interface nonstationarity*. Drift that develops over hours, such as the 1.9% excursion at hour 10, is within the envelope: it is detected and corrected in minutes. Changes faster than a probe episode, such as a sudden gait switch inside the locomotion controller or a slip-inducing floor, would violate the assumption that Θ_i is piecewise constant on the time scale of minutes. The third boundary is *descriptor expressiveness*. The descriptor captures a planar envelope, a step height, command limits, and a diagonal first-order interface. An embodiment whose safety depends on quantities outside this set, for example a humanoid whose stopping distance depends on gait phase or an articulated payload that changes the envelope, can be onboarded quickly but may be modeled optimistically.

What adding a new embodiment takes is, in this stack, a descriptor and a probe plan. The 2.1 h figure for the simulated humanoid matches the full-stack onboarding time in Table 2, and the variants without calibration or without configuration sharing required 2.4 h. The variant without the calibration layer needs a manually identified static interface model, and per-robot planners need their own parameter sets, which is consistent with the slightly longer times, although the differences are small.

07 Limitations

- **Inconsistent ablation summaries.** The two recorded summaries (Tables 2 and 3) disagree on the relative contribution of the layers. Only the conclusions shared by both are drawn, and a repeated, pre-registered ablation is needed to rank the layers.
- **No variance estimates.** The results are point estimates from one campaign and 10 short runs per ablation, without confidence intervals; differences of a few points or a few milliseconds should not be over-interpreted.
- **One site, one team.** All physical results come from a single three-floor course and a three-robot team. Outdoor terrain, crowds, and larger teams were not tested.
- **Simulated humanoid only.** The fourth embodiment was evaluated in simulation, so its 94% success does not transfer to hardware without validation.
- **Model class.** The interface model is diagonal and first order; cross-axis coupling and gait-dependent dynamics are not represented. Team coordination is limited to treating other robots as obstacles; no joint multi-robot planning was used.

08 Conclusion

Table 4: Recorded failure and stress cases.

Case	Observation	Stack response and outcome
Light-degraded stairwell (40 lx)	Future-range envelope widened 3.8x	Team speed reduced to 1.1 m/s; cleared with 16 cm mean margin; 2 of 10 runs used the conservative path around
B2-class interface drift (payload + battery)	Interface drifted 1.9% by hour 10	Triggered probes re-converged the model in 8 min (3 probes)
Simulated 4th embodiment (H1-class humanoid asset)	Descriptor and calibration probe plan generated in 2.1 h	First successful loop at 0.9 m/s; 94% success over 10 runs

This paper presented a cross-embodiment navigation stack in which a configuration-driven planner, an active interface calibration layer, and a control-aligned future-range layer share one versioned configuration descriptor. On a mixed team of two quadrupeds and a wheeled robot across a three-floor course, the full stack reached 88.2% end-to-end task success at a 28 ms p95 replanning latency. Each layer, when removed, cost between 3.0 and 10.6 points of success, although the recorded summaries disagree on which layer matters most. The failure cases show the intended degradation modes: perception uncertainty was converted into reduced speed (1.1 m/s in a 40 lx stairwell) or a detour, and a 1.9% interface drift was corrected within 8 min. A new embodiment was onboarded in simulation in 2.1 h by writing a descriptor and a probe plan, without code changes. The main open questions are a properly replicated ablation that ranks the layers, richer interface models for fast nonstationarity, and physical validation of embodiments beyond quadrupeds and wheeled bases.

References

- [1] T. Zang, S. Cheng, H. Huang, S. Wang, and W. Zhang. Agni-Nav: Configuration-Driven Cross-Embodiment Local Planning for Robot Navigation. arXiv:2606.10903, 2026.
- [2] T. Zang, H. Huang, S. Cheng, and W. Zhang. GAUGE: Planner-Conditioned Active Calibration of Opaque Quadruped Velocity Interfaces. arXiv:2607.02841, 2026.
- [3] T. Zang, S. Wang, H. Huang, and S. Cheng. FutureRay: Control-Aligned Future Range for Agile Quadruped Navigation. arXiv:2608.01197, 2026.
- [4] D. Shah, A. Sridhar, A. Bhorkar, N. Hirose, and S. Levine. GNM: A General Navigation Model to Drive Any Robot. arXiv:2210.03370, 2022.
- [5] D. Shah, A. Sridhar, N. Dashora, K. Stachowicz, K. Black, N. Hirose, and S. Levine. ViNT: A Foundation Model for Visual Navigation. arXiv:2306.14846, 2023.
- [6] A. Sridhar, D. Shah, C. Glossop, and S. Levine. NoMaD: Goal Masked Diffusion Policies for Navigation and Exploration. arXiv:2310.07896, 2023.
- [7] A. Eftekhari et al.. The One RING: a Robotic Indoor Navigation Generalist. arXiv:2412.14401, 2024.
- [8] J. Zhang et al.. Embodied Navigation Foundation Model. arXiv:2509.12129, 2025.
- [9] H. Xi, M. Tan, X. Zhang, S. Cheng, S. Wang, Y. Gu, X. Shen, and W. Zhang. CeRLP: A Cross-embodiment Robot Local Planning Framework for Visual Navigation. arXiv:2603.19602, 2026.
- [10] J. Zhang, Y. Du, X. Guo, S. Sun, X. Liu, Y. Sun, G. Lu, W. Sui, and J. Li. EA-Nav: Learning Safe Visual Navigation Policies with Embodiment Awareness. arXiv:2607.19880, 2026.
- [11] Open X-Embodiment Collaboration et al.. Open X-Embodiment: Robotic Learning Datasets and RT-X Models. arXiv:2310.08864, 2023.
- [12] Octo Model Team et al.. Octo: An Open-Source Generalist Robot Policy. arXiv:2405.12213, 2024.
- [13] R. Doshi, H. Walke, O. Mees, S. Dasari, and S. Levine. Scaling Cross-Embodied Learning: One Policy for Manipulation, Navigation, Locomotion and Aviation. arXiv:2408.11812, 2024.
- [14] D. Fox, W. Burgard, and S. Thrun. The Dynamic Window Approach to Collision Avoidance. IEEE Robotics and Automation Magazine, 4(1):23–33, 1997.
- [15] S. Macenski, F. Martín, R. White, and J. Ginés Clavero. The Marathon 2: A Navigation System. arXiv:2003.00368, 2020.
- [16] L. Wellhausen and M. Hutter. ArtPlanner: Robust Legged Robot Navigation in the Field. arXiv:2303.01420, 2023.
- [17] Y. Kim, C. Kim, and J. Hwangbo. Learning Forward Dynamics Model and Informed Trajectory Sampler for Safe Quadruped Navigation. arXiv:2204.08647, 2022.
- [18] N. Rudin, D. Hoeller, M. Bjelonic, and M. Hutter. Advanced Skills by Learning Locomotion and Local Navigation End-to-End. arXiv:2209.12827, 2022.
- [19] D. Hoeller, N. Rudin, D. Sako, and M. Hutter. ANYmal Parkour: Learning Agile Navigation for Quadrupedal Robots. arXiv:2306.14874, 2023.
- [20] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter. Learning Agile and Dynamic Motor Skills for Legged Robots. arXiv:1901.08652, 2019.
- [21] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning Quadrupedal Locomotion over Challenging Terrain. arXiv:2010.11251, 2020.
- [22] N. Rudin, D. Hoeller, P. Reist, and M. Hutter. Learning to Walk in Minutes Using Massively Parallel Deep Reinforcement Learning. arXiv:2109.11978, 2021.
- [23] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning Robust Perceptive Locomotion for Quadrupedal Robots in the Wild. arXiv:2201.08117, 2022.
- [24] A. Kumar, Z. Fu, D. Pathak, and J. Malik. RMA: Rapid Motor Adaptation for Legged Robots. arXiv:2107.04034, 2021.
- [25] G. Ji, J. Mun, H. Kim, and J. Hwangbo. Concurrent Training of a Control Policy and a State Estimator for Dynamic and Robust Legged Locomotion. arXiv:2202.05481, 2022.
- [26] G. B. Margolis and P. Agrawal. Walk These Ways: Tuning Robot Control for Generalization with Multiplicity of Behavior. arXiv:2212.03238, 2022.
- [27] I. Radosavovic, T. Xiao, B. Zhang, T. Darrell, J. Malik, and K. Sreenath. Real-World Humanoid Locomotion with Reinforcement Learning. arXiv:2303.03381, 2023.
- [28] K. Hausman, J. Preiss, G. S. Sukhatme, and S. Weiss. Observability-Aware Trajectory Optimization for Self-Calibration with Application to UAVs. arXiv:1604.07905, 2016.
- [29] E. S. Page. Continuous Inspection Schemes. Biometrika, 41(1/2):100–115, 1954.

- [30] V. Makoviychuk et al.. Isaac Gym: High Performance GPU-Based Physics Simulation For Robot Learning. arXiv:2108.10470, 2021.
- [31] A. Agarwal, A. Kumar, J. Malik, and D. Pathak. Legged Locomotion in Challenging Terrains using Egocentric Vision. arXiv:2211.07638, 2022.
- [32] X. Cheng, K. Shi, A. Agarwal, and D. Pathak. Extreme Parkour with Legged Robots. arXiv:2309.14341, 2023.
- [33] Z. Zhuang, Z. Fu, J. Wang, C. Atkeson, S. Schwertfeger, C. Finn, and H. Zhao. Robot Parkour Learning. arXiv:2309.05665, 2023.
- [34] J. Frey, M. Mattamala, N. Chebrolu, C. Cadena, M. Fallon, and M. Hutter. Fast Traversability Estimation for Wild Visual Navigation. arXiv:2305.08510, 2023.
- [35] G. Kahn, P. Abbeel, and S. Levine. BADGR: An Autonomous Self-Supervised Learning-Based Navigation System. arXiv:2002.05700, 2020.
- [36] A. Loquercio, E. Kaufmann, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza. Learning High-Speed Flight in the Wild. arXiv:2110.05113, 2021.
- [37] A. Kendall and Y. Gal. What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision? arXiv:1703.04977, 2017.
- [38] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada. Control Barrier Functions: Theory and Applications. arXiv:1903.11199, 2019.
- [39] J. Hu, S. Yuan, G. C. R. Bethala, A. Tzes, and Y. Fang. Learning Adaptive Safety Margins for Visual Navigation. arXiv:2607.18200, 2026.
- [40] M. Tranzatto et al.. CERBERUS: Autonomous Legged and Aerial Robotic Exploration in the Tunnel and Urban Circuits of the DARPA Subterranean Challenge. arXiv:2201.07067, 2022.
- [41] A. Agha et al.. NeBula: Quest for Robotic Autonomy in Challenging Environments; TEAM CoSTAR at the DARPA Subterranean Challenge. arXiv:2103.11470, 2021.
- [42] A. Kadian et al.. Sim2Real Predictivity: Does Evaluation in Simulation Predict Real-World Performance? arXiv:1912.06321, 2019.
- [43] P. Anderson et al.. On Evaluation of Embodied Navigation Agents. arXiv:1807.06757, 2018.