

Shared Configuration Abstractions for Heterogeneous Fleet Navigation

Hana Sato • Tomas Keller • Mei-Lin Chao

Advanced Robotics Group, Toyota Research Institute, Palo Alto, CA, USA

PUBLISHED

September 10, 2026

REGISTERED

September 17, 2026

RECORD TYPE

Preprint

ABSTRACT

Heterogeneous robot fleets that must follow, avoid, or rendezvous with one another need more than per-robot planning: each robot must reason about how its peers are shaped, how quickly they can stop, and how these properties change during operation. Configuration-driven local planners such as AgniNav describe a single robot through an abstract configuration of body geometry, sensing, and velocity limits, but they treat other robots as generic obstacles. This paper extends the configuration descriptor into a fleet-shared abstraction. Each robot publishes a versioned configuration handle on the fleet bus, and the unchanged local planner evaluates interaction-critical derived quantities, namely a predicted clearance envelope and a stop distance, as functions of the peer's live configuration, with conservative inflation when a handle is stale. The approach is evaluated in simulation on a mixed fleet of a Go2-class quadruped, a B2-class quadruped, and a wheeled robot across 12 corridor and intersection scenarios with 5 seeds each. Relative to a configuration-blind planner that models peers as footprint blobs, interaction success increases from 77.7 to 91.6 percent, envelope prediction error decreases from 12.2 cm to 10.8 cm, and stop-distance error decreases from 15.3 cm to 14.3 cm, while the 95th-percentile planner overhead is 3.1 ms compared with 3.5 ms. Ablations indicate that the stop-distance derivation, the clearance envelope, and live configuration updates each account for 9.0 to 10.1 percentage points of interaction success. Recorded failure cases illustrate behavior under stale handles and mid-episode payload changes.

INDEX TERMS heterogeneous multi-robot navigation, cross-embodiment local planning, configuration descriptors, legged robots, inter-robot collision avoidance, fleet coordination

KAGG NOTE Inter-robot safety in a mixed fleet depends on quantities derived from the peer's body, not only from one's own; publishing the configuration, rather than a footprint, lets every robot compute them.

01 Introduction

Mobile robot fleets in warehouses, campuses, and inspection sites are increasingly heterogeneous. A single site may operate compact quadrupeds for stairs and clutter, larger load-carrying quadrupeds, and wheeled bases for long flat runs. Such robots do not merely share space; they interact. One robot follows another through a corridor, two robots negotiate an intersection, or a robot converges on a peer for a hand-off. In each case, the decision of one robot depends on physical properties of the other: how wide it is, how far it extends ahead of its tracked center, and how far it travels before it can stop.

Cross-embodiment navigation research has made substantial progress on the single-robot side of this problem. Configuration-driven local planning, exemplified by AgniNav [1], describes a robot through an abstract configuration of body geometry, sensors, and velocity limits, so that one planning stack can run across embodiments without retraining. Related geometric pipelines such as CeRLP [2] and learned generalist policies such as GNM, ViNT, and NoMaD [3, 4, 5] share the same aspiration: decouple the navigation stack from any single body. However, these systems reason about the robot's own embodiment. Other robots enter the planner as generic dynamic obstacles, typically inflated discs or occupancy blobs, whose dimensions and dynamics are either assumed or inferred from perception.

This gap matters most in exactly the interactions a fleet is built for. A disc that is large enough to cover a load-carrying quadruped wastes corridor width when that robot turns sideways, and a disc that is small enough to admit passing underestimates the forward reach of a long body. More importantly, a geometric footprint does not encode stopping behavior. Whether a robot may enter an intersection ahead of a peer depends on whether that peer can stop before the conflict zone, which depends on the peer's speed, braking capability, actuation latency, and current payload. These quantities are known exactly by the peer itself and are expensive and unreliable to infer from outside.

This paper proposes to close the gap by sharing the configuration itself. Each robot publishes a versioned *configuration handle* that extends the AgniNav descriptor with the dynamic parameters required for interaction. Each robot's local planner, which is otherwise unchanged, evaluates two interaction-critical derived quantities as explicit functions of the peer's handle: a predicted clearance envelope and a stop distance. These quantities enter the planner as constraints and costs, govern a simple yielding rule at conflict zones, and are inflated conservatively when a handle becomes stale.

The contributions of this paper are as follows.

1. A fleet-shared configuration abstraction that extends a single-robot configuration descriptor [1] with braking, la-

tency, and payload parameters, together with versioning and staleness semantics for publication on a fleet bus.

2. Closed-form derivations of two interaction-critical quantities, the peer clearance envelope and the peer stop distance, as functions of the peer’s live configuration, and their integration into an unchanged sampling-based local planner as hard constraints, soft costs, and a stop-distance-based yielding rule.
3. A staleness-aware inflation scheme that degrades gracefully to conservative behavior when a peer’s handle is delayed, and a version-keyed cache that keeps per-cycle planner overhead low.
4. A simulated evaluation on a mixed fleet of a Go2-class quadruped, a B2-class quadruped, and a wheeled robot across 12 corridor and intersection scenarios, with comparisons against configuration-blind and fixed-peer-model baselines, an ablation of each derived quantity, and an analysis of recorded failure cases.

02 Related Work

2.1 Configuration-driven and cross-embodiment navigation

AgniNav [1] is the closest prior work and the direct basis of this paper. It factorizes cross-embodiment local planning around an abstract robot configuration. The configuration contains a four-parameter collision envelope (collision-relevant height, front length, rear length, and half width), sensor descriptors, and velocity limits. The height parameter conditions monocular image-to-scan prediction so that perception reports obstacles relevant to the declared body, while the planar envelope parameters configure collision checking in a sampling-based local planner. Because embodiment enters only through this configuration, a single planning stack can be deployed on robots of different size and locomotion type by changing parameters rather than retraining. The present work retains this factorization and the planner unchanged, but it addresses a question AgniNav does not pose: how a robot should use the configuration of *another* robot. In AgniNav, peers are indistinguishable from any other obstacle that perception reports. We extend the descriptor with dynamic parameters (braking deceleration, latency, payload), make it a published and versioned object, and derive peer-dependent quantities from it. In this sense, our method is a fleet-level generalization of AgniNav’s configuration interface rather than a replacement for its planner.

CeRLP [2] pursues a related decomposition in which heterogeneous visual observations are converted into a unified geometric representation, including scale-corrected depth and a height-adaptive scan abstraction, before a shared local planner acts on it. Like AgniNav, it makes the robot’s own body explicit but does not represent peer embodiments. Generalist navigation policies take a data-driven route: GNM [3] trains a single policy on trajectories from multiple robots using a normalized action space, ViNT [4] scales this approach with a Transformer backbone and goal-conditioned objective, and NoMaD [5] unifies goal-reaching and exploration with a diffusion decoder. These models learn priors that transfer across platforms, but embodiment is implicit in data and the policies are not designed to consume a peer’s physical parameters. RING [6] randomizes body size and camera configuration during training, NavFoM [7] introduces embodiment and camera identifiers into a unified navigation foundation model, and EA-Nav [8] conditions

imitation learning on embodiment geometry tokens and adds risk-aware correction. These methods demonstrate that explicit embodiment inputs reduce observation-action ambiguity for the ego robot; our work applies the same insight to the robot’s model of its peers. Hu et al. [9] learn context-dependent safety margins for visual navigation instead of fixed inflation. Their observation that static margins trade collisions against detours motivates our use of envelopes that adapt to the peer’s configuration and speed rather than a single blob radius.

2.2 Local planning for mobile robots

The dynamic window approach [10] samples admissible velocity commands within acceleration limits and scores the resulting arcs, a structure retained by many modern planners, including the sampling-based planner used by AgniNav. Timed elastic bands [11] optimize a trajectory in multiple homotopy classes under kinematic constraints, and model predictive path integral control [12] evaluates large batches of sampled control sequences under a stochastic optimal control objective. The Nav2 stack [13] packages such planners as interchangeable controller plugins over ROS 2 [14], whose publish-subscribe middleware and quality-of-service settings also provide the natural transport for configuration handles. Our method does not modify the planner’s search; it adds constraints and cost terms that depend on peer handles, so it is compatible with any of these planner families.

2.3 Multi-robot collision avoidance

Velocity obstacles [15] characterize the set of ego velocities that lead to collision with a moving obstacle within a time horizon. Reciprocal velocity obstacles [16] and optimal reciprocal collision avoidance (ORCA) [17] share avoidance responsibility between agents and yield collision-free velocities under a reciprocity assumption, and NH-ORCA [18] extends the method to non-holonomic robots by bounding tracking error. These methods are elegant and fast, but they typically model each agent as a disc with known radius and assume that all agents follow the same avoidance rule. Our envelope generalizes the disc to an oriented, speed-dependent region whose dimensions come from the peer’s published configuration, and our stop-distance terms make the braking asymmetry between agents explicit rather than assuming symmetric reciprocity.

Control barrier functions [19] provide forward-invariance guarantees for safe sets, and safety barrier certificates [20] apply them to multi-robot collision avoidance by minimally modifying nominal controls in a quadratic program. The barrier constraints in these works depend on the braking capabilities of both agents, so they presuppose exactly the peer information that our handles supply. The responsibility-sensitive safety model [21] formalizes safe longitudinal distances for vehicles in terms of response time and the minimum and maximum braking of each party; our following constraint adopts this structure and instantiates the parameters from configuration handles rather than from conservative population-wide bounds.

Learning-based decentralized avoidance, including CADRL [22], the multi-robot policy of Long et al. [23], and GA3C-CADRL [24], learns reactive policies from simulated interactions and does not require communication, but it encodes agents through observed position, velocity, and radius only. Trajectory-level multi-agent planners such as EGO-Swarm [25], MADER [26], and distributed model predictive control [27] share planned trajectories among agents and check them

for conflicts. Our approach is complementary: we share configurations rather than trajectories, which are smaller, change rarely, and remain valid when a peer replans. Conflict-based search [28] solves multi-agent path finding optimally on graphs and could supply the global routes on which our local interactions occur.

2.4 Heterogeneous fleets and interaction-aware navigation

Early architectures for heterogeneous teams, such as ALLIANCE [29], and formal treatments of multi-robot task allocation [30] established that robots should advertise capabilities so that peers can reason about them. The survey of Rizk et al. [31] documents how heterogeneity in sensing, mobility, and size complicates coordination and identifies capability modeling as a recurring requirement. Those works focus on task-level capabilities, whereas we advertise motion-level physical parameters for local planning. Research on crowd and social navigation shows that ignoring the interacting partner's behavior leads to freezing or unsafe motion [32], and recent surveys and evaluation guidelines [33, 34] call for metrics that capture interaction quality rather than only goal success. We follow that guidance by reporting interaction success together with prediction errors for the quantities that govern interaction.

2.5 Legged locomotion and time-varying dynamics

Quadrupeds such as ANYmal [35] and the MIT Cheetah 3 [36] achieve agile motion through model-based control, in which convex model-predictive control computes ground reaction forces under friction-cone constraints; these constraints bound the achievable deceleration and depend on body mass. Learned controllers [37, 38, 39] trained in massively parallel simulation [40] produce robust locomotion over rough terrain, and rapid motor adaptation [41] estimates environment and payload factors online from proprioceptive history. These results imply that a legged robot's stopping behavior is not a fixed constant: it changes with payload and terrain, and the robot itself is best placed to estimate it. Our live configuration handle is the mechanism through which such onboard estimates are made available to peers. Finally, Kadian et al. [42] show that simulation rankings do not always predict physical performance, which bounds the conclusions that can be drawn from the simulated study reported here.

03 Problem Formulation

Consider a fleet $\mathcal{N} = \{1, \dots, N\}$ of mobile robots operating in a shared planar workspace $\mathcal{W} \subset \mathbb{R}^2$. Robot i has state $x_i = (p_i, \theta_i, v_i, \omega_i)$, where $p_i \in \mathbb{R}^2$ is the position of its body-frame origin, θ_i its heading, and v_i, ω_i its forward and yaw rates. The local planner runs with control period Δt and plans over a horizon $T = K\Delta t$.

3.1 Configuration descriptor

Following AgniNav [1], each robot is described by a configuration. We write the extended configuration of robot i at time t as

$$c_i(t) = (g_i, k_i, \tau_i, M_i, m_i(t)), \quad (1)$$

$$g_i = (h_i, \ell_i^f, \ell_i^r, w_i), \quad k_i = (\bar{v}_i, \bar{\omega}_i, \bar{a}_i, \bar{b}_i),$$

where the geometry g_i consists of the collision-relevant height h_i , the front and rear extents ℓ_i^f and ℓ_i^r measured from the body-frame origin, and the half width w_i . The limits k_i comprise

$\bar{v}_i, \bar{\omega}_i, \bar{a}_i$, which bound speed, yaw rate, and forward acceleration, and $\bar{b}_i$ is the maximum braking deceleration of the unloaded robot. The term τ_i is the response latency from a stop command to the onset of deceleration, M_i is the nominal mass, and $m_i(t) \geq 0$ is the current payload mass. The geometric part g_i and $\bar{v}_i, \bar{\omega}_i$ are present in the original single-robot descriptor; $\bar{b}_i, \tau_i, M_i, m_i$ are the fleet extension. Only m_i is expected to change during an episode in this study, but the formulation allows any field to change.

3.2 Configuration handle

A configuration handle is the published record

$$H_i = (i, \kappa_i, t_i, c_i), \quad (2)$$

where $\kappa_i \in \mathbb{N}$ is a version number that increases whenever c_i changes and t_i is the publication timestamp. Robot j holds the most recent handle H_i it has received; its age at time t is $\alpha_i(t) = t - t_i$.

3.3 Interaction tasks and derived quantities

An interaction task couples two or more robots through a predicate on their joint trajectory: *follow* requires robot i to track behind robot j without closing below a safe gap, *avoid* requires two robots to pass in a corridor or cross at an intersection without contact, and *rendezvous* requires two robots to converge to a meeting configuration without overshoot into each other. An episode is an *interaction success* if every robot reaches its goal within the time limit, no collision occurs, and the interaction predicate holds throughout.

Robot i must reason about two quantities for every peer j . The first is the clearance envelope $\mathcal{E}_j(t + s) \subset \mathcal{W}$, the region that peer j may occupy at a future offset $s \in [0, T]$ including the distance it would cover while stopping. The second is the stop distance $d_j^{\text{stop}}(v)$, the distance peer j travels from a stop command issued at speed v until it comes to rest. The problem addressed in this paper is to compute $\mathcal{E}_j$ and d_j^{stop} from the peer's handle H_j , to use them in robot i 's local planner without modifying its search procedure, and to remain safe when H_j is delayed.

We evaluate the predictions against their realized values. The envelope prediction error is the mean distance between the predicted envelope boundary and the boundary of the region actually swept by the peer over the same horizon, and the stop-distance error is $|d_j^{\text{stop}}(v) - d_j^{\text{obs}}|$, where d_j^{obs} is the stopping distance measured in simulation when a stop is commanded.

04 Method

The method has three parts: a publication protocol for configuration handles, the derivation of peer quantities from handles, and the integration of these quantities into the local planner. The complete per-cycle procedure is summarized at the end of this section.

4.1 Handle publication on the fleet bus

Each robot publishes H_i on a fleet-wide topic in two situations: immediately when c_i changes (event-driven, with κ_i incremented), and periodically as a heartbeat with period T_{hb} carrying the unchanged version. Event-driven publication bounds the time between a physical change, such as a payload being loaded, and its availability to peers by the onboard detection latency plus transport delay. Payload mass m_i is estimated onboard; any estimator can be used, including load cells, joint-

torque residuals, or proprioceptive adaptation modules of the kind used in rapid motor adaptation [41]. The handle is compact (a few tens of bytes), so it can be sent with reliable delivery even on congested links. Receivers discard any handle whose version is lower than the one they hold, which makes the protocol robust to reordering.

4.2 Peer stop distance

Payload reduces achievable deceleration. For a force-limited drive, the deceleration scales inversely with total mass, and for a legged or wheeled robot it is also bounded by the friction coefficient μ of the contact. We therefore use the effective braking deceleration

$$b_j^{\text{eff}} = \min\left(\mu g, \bar{b}_j \frac{M_j}{M_j + m_j}\right), \quad (3)$$

where g is gravitational acceleration. The peer stop distance from speed v is then the latency travel plus the constant-deceleration braking distance,

$$d_j^{\text{stop}}(v) = v \tau_j + \frac{v^2}{2 b_j^{\text{eff}}}. \quad (4)$$

Equation (4) is deliberately simple. It is exact for a first-order latency followed by constant deceleration, it is monotone in v , τ_j , and m_j , and it is differentiable, which allows it to be used inside gradient-based planners as well as sampling-based ones. For illustration only, a robot with $\tau_j = 0.1$ s and $b_j^{\text{eff}} = 3$ m/s² moving at 1.5 m/s has $d_j^{\text{stop}} = 0.15 + 0.375 = 0.525$ m; a payload equal to 30% of M_j lowers b_j^{eff} to about 2.31 m/s² and raises d_j^{stop} to about 0.64 m. A difference of this order is comparable to the half width of a mid-sized quadruped, which is why a fixed nominal model can misjudge interactions after the payload changes.

4.3 Peer clearance envelope

Let $\hat{x}_j(t+s) = (\hat{p}_j, \hat{\theta}_j, \hat{v}_j)$ denote robot i 's prediction of peer j 's state at offset s , obtained from onboard tracking or from pose broadcasts by a constant-velocity, constant-yaw-rate roll-out. In the peer's predicted body frame, a point $q \in \mathbb{R}^2$ corresponds to the workspace point $p = \hat{p}_j + R(\hat{\theta}_j) q$, where $R(\cdot)$ is the planar rotation. The clearance envelope is the oriented box obtained by mapping the body-frame box $\mathcal{B}_j(s)$ into the workspace,

$$\begin{aligned} \mathcal{E}_j(t+s) &= \{\hat{p}_j + R(\hat{\theta}_j) q : q \in \mathcal{B}_j(s)\}, \\ \mathcal{B}_j(s) &= [-\ell_j^r - \delta_j, \ell_j^f + d_j^{\text{stop}}(\hat{v}_j) + \delta_j] \\ &\quad \times [-w_j - \delta_j, w_j + \delta_j], \end{aligned} \quad (5)$$

which covers the peer's body at the predicted pose, extended forward by the distance it would need to stop if it began braking at that instant. The extension is applied only in the direction of travel, so a long robot moving slowly is not penalized laterally. The margin δ_j accounts for prediction uncertainty and handle staleness,

$$\delta_j(s) = \delta_0 + \sigma_j(s) + \bar{v}_j [\alpha_j - T_{\text{hb}}]_+ + \frac{1}{2} \bar{a}_j [\alpha_j - T_{\text{hb}}]_+^2, \quad (6)$$

where δ_0 is a fixed base margin, $\sigma_j(s)$ is the standard deviation of the predicted position at offset s , and $[\cdot]_+ = \max(\cdot, 0)$. When the handle is fresh ($\alpha_j \leq T_{\text{hb}}$), the last two terms vanish. When it is stale, the margin grows with the distance the

peer could have covered at its maximum speed and acceleration since the last expected update. In addition, while $\alpha_j > T_{\text{hb}}$ the stop distance is evaluated with the most conservative payload the peer class admits, $m_j = m_j^{\text{max}}$, so that a missed payload update cannot shorten the envelope. This rule reproduces the intended behavior in the recorded failure case of Section 5.5: a stale handle widens the envelope and causes a delay rather than an unsafe maneuver.

The height h_j is not used in (5). It is retained in the handle because it determines whether peer j occludes robot i 's sensors and because the original descriptor uses it to condition perception [1].

4.4 Integration into the local planner

The local planner is the unchanged per-robot sampling-based planner: at each control cycle it samples candidate commands $u = (v, \omega)$ within the dynamic window [10] defined by robot i 's own limits, rolls out each candidate over K steps, and selects the minimum-cost admissible candidate. Let $\mathcal{F}_i(u, s)$ be robot i 's own footprint at offset s under candidate u , computed from its own configuration as in AgniNav, and let $\text{sd}(A, B)$ be the signed distance between two sets (positive when disjoint). Peer handles enter the planner in three ways.

Hard envelope constraint. A candidate is admissible only if

$$\text{sd}(\mathcal{F}_i(u, s_k), \mathcal{E}_j(t+s_k)) \geq 0 \quad \forall j \neq i, k = 1, \dots, K, \quad (7)$$

in addition to the existing static-obstacle constraint. Because $\mathcal{E}_j$ already contains the peer's stopping region, satisfying (7) means that robot i does not plan into space that peer j might still occupy if it began to stop.

Soft interaction cost. Admissible candidates are scored by

$$\begin{aligned} J(u) &= w_g J_{\text{goal}}(u) + w_s J_{\text{smooth}}(u) + w_o J_{\text{obs}}(u) \\ &\quad + w_x \sum_{j \neq i} \sum_{k=1}^K [\rho - \text{sd}(\mathcal{F}_i(u, s_k), \mathcal{E}_j(t+s_k))]_+^2, \end{aligned} \quad (8)$$

where the first three terms are the planner's existing goal, smoothness, and static-obstacle costs and the last term penalizes candidates that pass within a comfort distance ρ of any peer envelope. The quadratic hinge keeps the cost zero far from peers, so the interaction term does not distort behavior in free space.

Following and yielding. For a follow task with leader j ahead of robot i along a shared path, the gap Γ_{ij} between the front of i and the rear of j must satisfy the responsibility-sensitive condition [21]

$$\Gamma_{ij} \geq d_i^{\text{stop}}(v_i) - \frac{v_j^2}{2 \bar{b}_j} + \epsilon, \quad (9)$$

where the follower uses its own effective braking via d_i^{stop} and the leader is assumed to brake as hard as its unloaded limit $\bar{b}_j$ allows, which is the worst case for the follower; $\epsilon > 0$ is a standstill margin. Constraint (9) is enforced as an upper bound on the sampled v . At an intersection or corridor bottleneck with conflict zone Z , let λ_j be the path distance from peer j 's front to the entry of Z . Robot i proceeds into Z only if the peer can stop before entering it,

$$\lambda_j \geq d_j^{\text{stop}}(\hat{v}_j) + \delta_j, \quad (10)$$

and otherwise yields by restricting its candidates to those that stop before Z . If both robots satisfy (10) with respect to each

other, the tie is broken by the lower handle identifier i , which makes the rule deterministic and prevents symmetric deadlock. The rule needs no negotiation messages: each robot evaluates the same inequality from the same handles.

4.5 Caching and computational cost

The derived parameters that depend only on the configuration, namely b_j^{eff} , the configuration-dependent extents of $\mathcal{B}_j$ in (5), and the conservative fallback values, are computed once per received version κ_j and cached. Per cycle, the planner evaluates (4) at the predicted speeds, which is a closed-form expression, and performs box-to-footprint signed-distance queries. The added work therefore scales as $O(|\mathcal{C}| K (N - 1))$ for $|\mathcal{C}|$ candidates, and it replaces rather than adds to the disc-based dynamic-obstacle checks of the baseline planner.

4.6 Per-cycle procedure

1. Receive handles; for each peer j with a new version κ_j , recompute and cache the configuration-dependent quantities.
2. For each peer, compute the handle age α_j , the margin $\delta_j(s)$ from (6), and, if stale, the conservative payload fallback.
3. Predict peer states $\hat{x}_j(t + s_k)$ and construct envelopes $\mathcal{E}_j(t + s_k)$ from (4) and (5).
4. Apply task constraints: follow gap (9) and conflict-zone yielding (10).
5. Sample candidates in the dynamic window, reject those violating (7), score the remainder with (8), and execute the minimum-cost command.
6. If the robot's own configuration has changed, increment κ_i and publish H_i immediately.

05 Experiments

5.1 Setup and platform

The study uses a simulated mixed fleet of three robots: two quadrupeds, one Go2-class and one B2-class, and one wheeled robot. The three platforms differ in footprint, mass, and braking behavior, and the B2-class robot carries a payload that can change during an episode. The evaluation comprises 12 corridor and intersection scenarios that exercise the follow, avoid, and rendezvous interactions. Each scenario is run with 5 random seeds and 30 episodes per scenario-seed pair, for a total of 1,800 episodes per method. Configuration handles are published on the fleet bus, and the interaction-critical quantities are computed from the peer's configuration as described in Section 4. The local planner itself is unchanged from the per-robot baseline; all methods share the same candidate sampling, rollout, and non-interaction cost terms.

5.2 Baselines and metrics

Three baselines isolate the contribution of peer configurations. The *configuration-blind planner* represents every peer as a footprint blob, a disc of fixed size independent of the peer's actual geometry or dynamics. The *per-robot planner without interaction terms* uses the robot's own configuration for collision checking, as in AgniNav [1], but treats peers as generic dynamic obstacles without the envelope, following, or yielding terms. The *interaction-terms, fixed peer model* baseline uses the same interaction terms as our method but evaluates them on a fixed model of each peer rather than on the live published configuration, so payload changes and other updates are not

reflected. We report interaction success, envelope prediction error, stop-distance error, and the 95th percentile of per-cycle planner overhead, as defined in Section 3.

5.3 Main results

Table 1 reports the main comparison. The fleet-shared configuration method achieves 91.6% interaction success, compared with 77.7% for the configuration-blind planner, 77.9% for the per-robot planner without interaction terms, and 88.6% for interaction terms on a fixed peer model. The 13.9-point gain over the configuration-blind planner corresponds to a reduction of the interaction failure rate from 22.3% to 8.4%, or about 62% in relative terms. The per-robot planner, which models its own body accurately but not its peers, performs nearly identically to the configuration-blind planner (77.9% versus 77.7%), which indicates that accurate self-modeling alone does not improve interaction outcomes. Adding interaction terms, even with a fixed peer model, accounts for most of the improvement (10.7 points over the per-robot planner), and live configurations contribute a further 3.0 points.

Envelope prediction error decreases monotonically across the four methods, from 12.2 cm to 11.9 cm, 11.2 cm, and 10.8 cm. The total reduction of 1.4 cm, about 11% relative to the configuration-blind planner, is modest in absolute terms, which is consistent with the fact that all methods predict the peer's position with the same tracker and differ only in the shape and extent assigned around it. Stop-distance error does not follow the same ordering. The fixed peer model has the largest stop-distance error (16.4 cm), larger than the configuration-blind planner (15.3 cm) and the per-robot planner (15.8 cm). A plausible explanation, consistent with the payload failure case below, is that a fixed model is systematically biased once the B2-class robot's payload departs from its nominal value, whereas the baselines without explicit stop-distance models are not tied to a particular nominal value. Our method has the lowest stop-distance error (14.3 cm), 1.0 cm below the configuration-blind planner and 2.1 cm below the fixed peer model.

The 95th-percentile planner overhead lies between 3.1 ms and 3.5 ms for all methods, and our method has the lowest value (3.1 ms). We do not interpret the ordering of these values as significant, because the differences are at most 0.4 ms and dispersion statistics were not recorded. The relevant conclusion is that evaluating peer-dependent envelopes and stop distances does not increase the tail latency of the planner relative to disc-based peer handling, which is consistent with the version-keyed caching and closed-form evaluation described in Section 4.

5.4 Ablation

Table 2 removes one component at a time. Removing the stop-distance derivation reduces interaction success from 91.6% to 81.5% (10.1 points), removing the clearance envelope reduces it to 82.6% (9.0 points), and replacing live configurations with a fixed peer model reduces it to 81.7% (9.9 points). Each component is therefore individually necessary, and the stop-distance derivation has the largest single effect. This ordering agrees with the intuition that intersection and following interactions are governed by whether a peer can stop in time, which a purely geometric envelope cannot capture.

The fixed-peer-model ablation (81.7%) is lower than the fixed-peer-model baseline in Table 1 (88.6%). The two entries were logged as separate runs, and the recorded data do not

Table 1: Main comparison on the mixed fleet (12 scenarios, 5 seeds, 30 episodes each). Interaction success is higher-is-better; prediction errors and planner overhead are lower-is-better.

Method	Interaction success (%)	Envelope prediction error (cm)	Stop-distance error (cm)	Planner overhead p95 (ms)
Configuration-blind planner (footprint blobs)	77.7	12.2	15.3	3.5
Per-robot planner, no interaction terms	77.9	11.9	15.8	3.2
Interaction terms, fixed peer model	88.6	11.2	16.4	3.3
Ours (fleet-shared configurations)	91.6	10.8	14.3	3.1

Table 2: Ablation of the fleet-shared configuration components.

Variant	Interaction success (%)
Ours (fleet-shared configurations)	91.6
w/o stop-distance derivation	81.5
w/o clearance envelope	82.6
fixed peer model (no live config)	81.7

isolate the source of this 6.9-point difference. We therefore treat the comparison in Table 1 as the conservative estimate of the benefit of live configurations (3.0 points) and the ablation as an upper indication (9.9 points); resolving the discrepancy requires a controlled rerun with matched settings.

5.5 Failure cases

Stale handle at an intersection. In intersection scenario 9, the wheeled robot’s published configuration was 1.2 s stale due to bus backpressure. The quadruped’s planner detected the age through α_j , inflated the margin according to (6), and evaluated the stop distance with the conservative payload fallback. The resulting envelope widened conservatively; no incident occurred, and the cost was a 0.4 s delay. This case illustrates the intended degradation mode: staleness reduces efficiency rather than safety. It also exposes a dependency on the bus quality of service, since longer outages would cause correspondingly longer delays.

Blob model in a corridor. The configuration-blind baseline treats the B2-class robot as a 0.6 m blob. In corridor 5, this produced 4 near-collisions per episode. A disc of fixed diameter does not represent the elongated body of a large quadruped or the forward extension contributed by its stopping distance, so passing maneuvers that appear clear under the blob model come too close to the actual body. The oriented, speed-dependent envelope in (5) avoids this mismatch by construction.

Payload change under a fixed peer model. The fixed peer model degrades when the B2-class robot’s payload changes mid-episode: its stop-distance error reaches 26 cm, compared with its average of 16.4 cm in Table 1. With live handles, the payload change triggers an event-driven publication, and the updated stop distance reaches peer planners in 41 ms. This is the mechanism through which live configurations improve the stop-distance error and interaction success relative to the fixed peer model.

06 Limitations

The study has several limitations. First, all results are obtained in simulation. Braking behavior, contact friction, and bus latency in simulation may differ from physical robots, and simulated rankings do not always transfer [42]; physical validation on the same fleet is required. Second, the fleet contains three robots of three types. Larger fleets would stress the bus and the pairwise checks, and more diverse embodiments, such as mo-

bile manipulators whose envelope changes with arm motion, are not represented by the rigid box model in (5). Third, only point estimates were recorded for each metric; without confidence intervals or per-seed dispersion, small differences, such as those in envelope error and planner overhead, should not be over-interpreted. Fourth, the discrepancy between the fixed-peer-model rows of Tables 1 and 2 is unresolved. Fifth, the method trusts published handles. A faulty payload estimator or a misconfigured robot would publish incorrect parameters, and the current design has no mechanism to cross-check a handle against observed motion. Finally, the stop-distance model (4) assumes constant deceleration on level ground and does not model slopes, slippery surfaces, or gait-phase effects on legged braking.

07 Conclusion

This paper extended a configuration-driven, cross-embodiment local planner into a fleet-shared abstraction. Each robot publishes a versioned configuration handle, and peers derive a clearance envelope and a stop distance from it, which enter an unchanged local planner as constraints, costs, and a deterministic yielding rule, with conservative inflation when handles are stale. In simulation on a mixed fleet of two quadrupeds and one wheeled robot, the approach raised interaction success from 77.7% for a configuration-blind planner to 91.6%, reduced envelope and stop-distance prediction errors to 10.8 cm and 14.3 cm, and kept 95th-percentile planner overhead at 3.1 ms. Ablations show that the stop distance, the clearance envelope, and live updates are each necessary. Future work will validate the approach on physical robots, cross-check published handles against observed peer motion, extend the envelope to articulated and time-varying bodies, and characterize behavior under prolonged bus degradation.

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] 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.
- [3] D. Shah, A. Sridhar, A. Bhorkar, N. Hirose, and S. Levine. GNM: A General Navigation Model to Drive Any Robot. arXiv:2210.03370, 2022.
- [4] 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.
- [5] A. Sridhar, D. Shah, C. Glossop, and S. Levine. NoMaD: Goal Masked Diffusion Policies for Navigation and Exploration. arXiv:2310.07896, 2023.
- [6] A. Eftekhari et al.. The One RING: a Robotic Indoor Navigation Generalist. arXiv:2412.14401, 2024.

- [7] J. Zhang et al.. Embodied Navigation Foundation Model. arXiv:2509.12129, 2025.
- [8] 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.
- [9] J. Hu, S. Yuan, G. C. R. Bethala, A. Tzes, and Y. Fang. Learning Adaptive Safety Margins for Visual Navigation. arXiv:2607.18200, 2026.
- [10] D. Fox, W. Burgard, and S. Thrun. The Dynamic Window Approach to Collision Avoidance. *IEEE Robotics and Automation Magazine*, 4(1), 1997.
- [11] C. Rösmann, F. Hoffmann, and T. Bertram. Integrated Online Trajectory Planning and Optimization in Distinctive Topologies. *Robotics and Autonomous Systems*, 88, 2017.
- [12] G. Williams, N. Wagener, B. Goldfain, P. Drews, J. M. Rehg, B. Boots, and E. A. Theodorou. Information Theoretic MPC for Model-Based Reinforcement Learning. *IEEE International Conference on Robotics and Automation (ICRA)*, 2017.
- [13] S. Macenski, F. Martín, R. White, and J. Ginés Clavero. The Marathon 2: A Navigation System. arXiv:2003.00368, 2020.
- [14] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall. Robot Operating System 2: Design, Architecture, and Uses in the Wild. arXiv:2211.07752, 2022.
- [15] P. Fiorini and Z. Shiller. Motion Planning in Dynamic Environments Using Velocity Obstacles. *The International Journal of Robotics Research*, 17(7), 1998.
- [16] J. van den Berg, M. Lin, and D. Manocha. Reciprocal Velocity Obstacles for Real-Time Multi-Agent Navigation. *IEEE International Conference on Robotics and Automation (ICRA)*, 2008.
- [17] J. van den Berg, S. J. Guy, M. Lin, and D. Manocha. Reciprocal n-Body Collision Avoidance. *Robotics Research (ISRR)*, Springer Tracts in Advanced Robotics, vol. 70, 2011.
- [18] J. Alonso-Mora, A. Breitenmoser, M. Rufli, P. Beardsley, and R. Siegwart. Optimal Reciprocal Collision Avoidance for Multiple Non-Holonomic Robots. *Distributed Autonomous Robotic Systems*, Springer Tracts in Advanced Robotics, vol. 83, 2013.
- [19] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada. Control Barrier Functions: Theory and Applications. arXiv:1903.11199, 2019.
- [20] L. Wang, A. D. Ames, and M. Egerstedt. Safety Barrier Certificates for Collisions-Free Multirobot Systems. *IEEE Transactions on Robotics*, 33(3), 2017.
- [21] S. Shalev-Shwartz, S. Shammah, and A. Shashua. On a Formal Model of Safe and Scalable Self-driving Cars. arXiv:1708.06374, 2017.
- [22] Y. F. Chen, M. Liu, M. Everett, and J. P. How. Decentralized Non-communicating Multiagent Collision Avoidance with Deep Reinforcement Learning. arXiv:1609.07845, 2016.
- [23] P. Long, T. Fan, X. Liao, W. Liu, H. Zhang, and J. Pan. Towards Optimally Decentralized Multi-Robot Collision Avoidance via Deep Reinforcement Learning. arXiv:1709.10082, 2017.
- [24] M. Everett, Y. F. Chen, and J. P. How. Motion Planning Among Dynamic, Decision-Making Agents with Deep Reinforcement Learning. arXiv:1805.01956, 2018.
- [25] X. Zhou, J. Zhu, H. Zhou, C. Xu, and F. Gao. EGO-Swarm: A Fully Autonomous and Decentralized Quadrotor Swarm System in Cluttered Environments. arXiv:2011.04183, 2020.
- [26] J. Tordesillas and J. P. How. MADER: Trajectory Planner in Multi-Agent and Dynamic Environments. arXiv:2010.04466, 2020.
- [27] C. E. Luis and A. P. Schoellig. Trajectory Generation for Multiagent Point-To-Point Transitions via Distributed Model Predictive Control. arXiv:1809.04230, 2018.
- [28] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant. Conflict-Based Search for Optimal Multi-Agent Pathfinding. *Artificial Intelligence*, 219, 2015.
- [29] L. E. Parker. ALLIANCE: An Architecture for Fault Tolerant Multirobot Cooperation. *IEEE Transactions on Robotics and Automation*, 14(2), 1998.
- [30] B. P. Gerkey and M. J. Matarić. A Formal Analysis and Taxonomy of Task Allocation in Multi-Robot Systems. *The International Journal of Robotics Research*, 23(9), 2004.
- [31] Y. Rizk, M. Awad, and E. W. Tunstel. Cooperative Heterogeneous Multi-Robot Systems: A Survey. *ACM Computing Surveys*, 52(2), 2019.
- [32] P. Trautman and A. Krause. Unfreezing the Robot: Navigation in Dense, Interacting Crowds. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010.
- [33] C. Mavrogiannis, F. Baldini, A. Wang, D. Zhao, P. Trautman, A. Steinfeld, and J. Oh. Core Challenges of Social Robot Navigation: A Survey. arXiv:2103.05668, 2021.
- [34] A. Francis et al.. Principles and Guidelines for Evaluating Social Robot Navigation Algorithms. arXiv:2306.16740, 2023.
- [35] M. Hutter et al.. ANYmal - a Highly Mobile and Dynamic Quadrupedal Robot. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016.
- [36] J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim. Dynamic Locomotion in the MIT Cheetah 3 Through Convex Model-Predictive Control. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.
- [37] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning Quadrupedal Locomotion over Challenging Terrain. arXiv:2010.11251, 2020.
- [38] 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.
- [39] N. Rudin, D. Hoeller, P. Reist, and M. Hutter. Learning to Walk in Minutes Using Massively Parallel Deep Reinforcement Learning. arXiv:2109.11978, 2021.
- [40] V. Makoviychuk et al.. Isaac Gym: High Performance GPU-Based Physics Simulation For Robot Learning. arXiv:2108.10470, 2021.
- [41] A. Kumar, Z. Fu, D. Pathak, and J. Malik. RMA: Rapid Motor Adaptation for Legged Robots. arXiv:2107.04034, 2021.
- [42] A. Kadian et al.. Sim2Real Predictivity: Does Evaluation in Simulation Predict Real-World Performance? arXiv:1912.06321, 2019.