Clearance Deformation, Not Object Tracking: Manifold Continuation for Safe Navigation under Limited Egocentric Sensing

Qifei Cui[1] Ruichen Deng[2] Bangan Wang[2]
[1]Department of Computer and Information Science, University of Pennsylvania
[2]Department of Mechanical Engineering and Applied Mechanics, University of Pennsylvania

Abstract

Safe and temporally consistent navigation in dynamic environments is hard for two reasons: limited egocentric sensing and limited onboard compute. Standard navigators handle dynamics by tracking individual obstacles. Tracking is fragile under occlusion, computationally heavy, and category-bound; the representation grows with the number of obstacles. We propose an alternative: treat dynamic obstacles as local deformations of a reusable clearance field. The robot maintains a single robot-frame relative field rate on persistently known support, fuses observations into a two-layer field memory (slow static, fast dynamic) to keep the corridor center line stable across turns, synthesizes a navigation manifold on an arrival-conditioned predicted field, and continues that manifold across frames via a translate–repair–optimize–validate tracker. Safety, performance, and stability decouple in center-line plus offset coordinates: safety is a hard pointwise box constraint, performance is a banded QP, and the tracking error is bounded by manifold normal velocity over the lateral gain. A multi-boundary field-CBF reads K boundary samples directly from the clearance field; a guard-zone plus material-derivative time-to-contact gate prevents intervention on distant boundaries; an adaptive but bounded buffer prevents discrete-time grazing. Across six scenarios with 10 paired seeds each, the sensing-failure rate is exactly zero, the headline empirical confirmation of the continuous-boundary assumption. A point-DS stress baseline that strips the manifold from upstream of the same safety filter isolates the manifold's safety contribution: 50% to 0% collisions on dynamic hallway and 10% to 0% on a 20 m dynamic corridor. The remaining open issue is the field-of-view information limit on blind corners, which forces collisions on any FOV-restricted controller.

1 Introduction

Object-level representations and their limits. Robots that navigate among people, doors, and carts have, almost universally, been built on a representation that enumerates obstacles. SLAM plus semantic segmentation plus a pedestrian tracker plus a planner that consumes tracked trajectories is the standard architecture [11, 15]. The representation grows linearly with the number of obstacles, and every component is brittle along a different axis: semantic segmentation fails on the long tail; trackers lose lock under occlusion; the planner depends on the class label assigned to each obstacle. Workarounds exist for each failure mode, but the representation itself — a list of typed objects — is the source of the brittleness.

The controller-sufficient signal. The relevant question for the controller is not where each object is, but whether the passage is open and remains open until the robot transits it. Maintaining a clearance field Γ (distance to the nearest boundary, regardless of class) reduces the dynamic environment to one signal: the field's relative rate of change in the robot frame, $\dot{\Gamma}_{\mathrm{rel}}$. Pedestrians, doors, carts, and the robot's own motion all collapse into this rate. The controller's decisions (proceed, yield, retreat) depend on Γ and $\dot{\Gamma}_{\mathrm{rel}}$ projected onto a one-dimensional locus inside the passage, not on per-object state.

Approach. We build the navigator on three design principles. *(i) Deformation, not tracking.* Dynamic obstacles enter the algorithm only as local deformations of the clearance field, with no object identities,

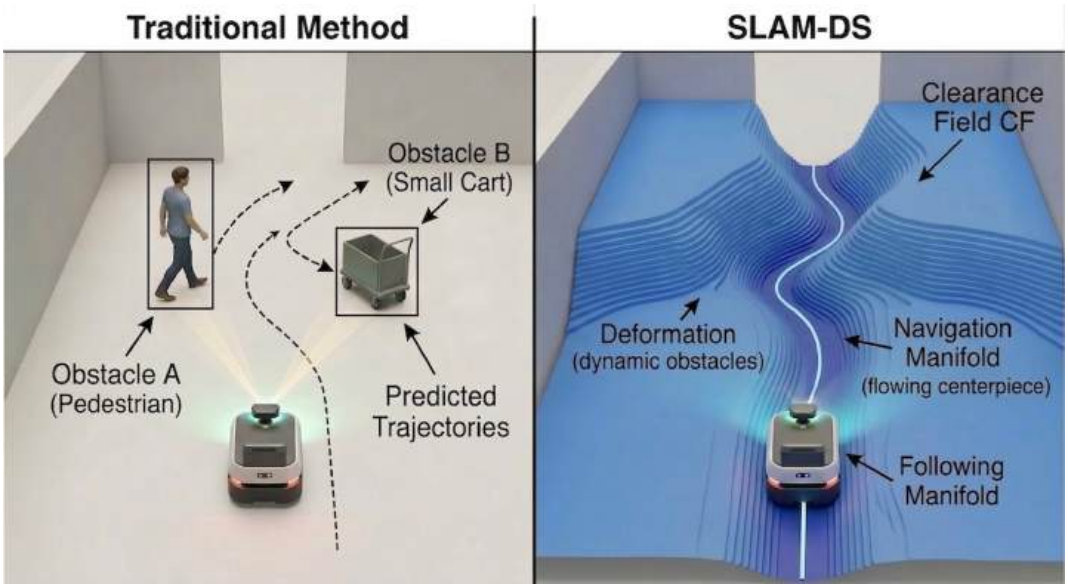


Figure 1: **Clearance-field deformation, not object tracking.** Left, traditional pipeline: enumerate obstacles, track each, predict per-object trajectories. Right, our representation: a single clearance field Γ whose contours deform as the world moves; the navigation manifold γ_t^* flows along the ridge of Γ.

no semantic classes, no per-object filters. *(ii) Manifold continuation, not replanning.* The local navigation target is a curve γ_t^* embedded in predicted free space. A stateful tracker translates the previous-frame manifold forward, repairs the prefix inside an adaptive tube, regrows the suffix, and validates the result; when the geometry remains regular the manifold persists, otherwise the tracker falls back to full extraction and reports the cause through a three-category diagnostic. *(iii) Stability-aware tracking.* A closed-form practical-stability bound relates the tracking error to the manifold's normal velocity over the lateral gain; the tracker is engineered to respect the C^1-regularity condition the bound requires.

Scaling under limited egocentric compute. The algorithm's state size does not grow with the number of obstacles; it grows with the manifold discretization ($N = 21$ stations in our default). Manifold continuation reuses the prior-frame solution, so per-frame work is dominated by a banded QP on a one-dimensional offset. Because safety, performance, and stability decouple structurally, the safety analysis does not have to be revisited when the performance objective or the predictor changes.

Contributions. **C1:** clearance-deformation navigation without object tracking, plus a two-layer field memory (slow static and fast dynamic) that keeps the center line stable through turns and re-detections (Section 4.1). **C2:** stateful manifold continuation via a translate–repair–optimize–validate tracker with a three-category fallback diagnostic (Section 4.4). **C3:** decoupled safety, performance, and stability in center-line plus offset coordinates (Section 5). **C4:** a multi-boundary field-CBF with a guard-zone plus material-derivative-TTC activation gate and a bounded adaptive buffer; object-free and free of the medial-axis gradient degeneracy of a single-scalar field-CBF (Section 4.3). **C5:** empirical validation in which a point-DS stress baseline (same field-CBF, no manifold) isolates the manifold's safety contribution (dynamic hallway 50% $\rightarrow$ 0% and long corridor 10% $\rightarrow$ 0% collision rate), plus a component ablation that switches off the tracker, the SLAM memory, and the multi-boundary CBF independently (Section 6).

2 Related Work

Classical learned dynamical systems for motion generation use fixed-point attractors and reshape the velocity field around known obstacles [4, 5]; the attractor is a point, so a changing local passage geometry does

not change the target. Our work replaces the point with a time-varying curve while keeping the DS controller structure. Manifold-aware DS (neural geometric fabrics [13], SE(3) LPV-DS [10], partially-contracting DS [7]) define the manifold from demonstrations or task structure; ours defines it from predicted free space at runtime. Predictive navigators such as SCOPE [14] and Dynamic Gap [2] also decouple prediction from policy, but feed sample-based planners; our predicted field feeds a manifold attractor, so passage selection is an emergent geometric consequence rather than a discrete branch comparison.

CBFs [1] build h as the signed distance to a finite list of obstacles. In dynamic environments these object-CBFs require either an obstacle tracker or a higher-order extension that estimates obstacle velocity; recent work addresses these via observation-conditioned barriers [3], non-convex unsafe-set analyses [6], and coupled barrier-certified DS [9]. We define the barrier directly on the clearance field via K active boundary samples; apparent boundary motion enters through $\partial_t \Gamma_{\text{curr}}$, the safety layer stays object-free, and the medial-axis gradient cancellation a single-scalar field-CBF would inherit is avoided by construction. Frenet-frame methods [12] decompose motion similarly into a longitudinal reference and a lateral offset, but the reference is a fixed lane center with no optimality property; in our formulation the zero-offset point is the clearance-maximizing ridge, so any nonzero offset is an explicit, bounded trade. The dynamic-SLAM line [11, 15] and object-level trajectory predictors [8] maintain per-object state and accept depth plus segmentation as input; we accept only depth and never instantiate an object list, so comparisons in Section 6 are restricted to baselines with comparable sensing requirements.

3 Problem Setting and Notation

Workspace and free space. Let $\mathcal{W} \subset \mathbb{R}^2$ be a compact, connected workspace with static boundary $\partial\mathcal{W}$. The time-varying obstacle set is $\mathcal{O}_t = \mathcal{O}_{\text{static}} \cup \bigcup_{i=1}^{N(t)} \mathcal{O}_t^{(i)}$, with each dynamic component having a continuously moving boundary. Free space is $\mathcal{F}_t = \mathcal{W} \setminus \mathcal{O}_t$; robot-feasible free space is $\mathcal{F}_t^r = \{x \in \mathcal{F}_t : d(x, \partial\mathcal{O}_t) > r_{\text{robot}}\}$.

Clearance field. $\Gamma(x,t) = d(x, \partial\mathcal{O}_t)$ for $x \in \mathcal{F}_t$, and 0 otherwise. For continuous-derivative arguments we use the soft-min surrogate Γ_α on smooth branches with $\Gamma_\alpha \to \min_i d_i$ as $\alpha \to \infty$ (Appendix A); the implementation uses a discrete distance transform.

Robot and sensing. Single-integrator $\dot{x} = u$, $x, u \in \mathbb{R}^2$, exact for a holonomic omniwheel base. Robot radius $r_{\text{robot}} = 0.12$ m; actuator caps $\max v_x = 0.75$ m/s, $\max v_y = 0.5$ m/s, $\max \omega_z = 1.2$ rad/s. Simulated depth sensor: FOV 120°, range 2.5 m, field resolution 0.15 m. The intended deployment target is an XLeRobot-class platform with Jetson onboard compute, an omnidirectional base, and an Intel RealSense D455 RGB-D camera; the depth-to-occupancy-grid front-end that feeds Layer 1 is documented in Appendix D. Hardware results are deferred to a follow-up report.

Continuous boundary deformation. Let $\Omega_K(t)$ denote the robot-frame subset whose geometry is currently maintained from observation plus decayed local memory, and $\Omega_\cap(t) := \Omega_K(t) \cap \Omega_K(t - \Delta t)$ the persistently known overlap.

Assumption 1 (Continuous boundary deformation)**.** With RGBD sensing, no obstacle appears instantaneously inside $\Omega_\cap(t)$: every observed boundary enters through the FOV boundary and then deforms continuously.

The assumption is essential for the field-rate channel (Section 4.1) and the practical-stability bound (Section 5.2). It excludes occluded teleporting motion and sensor dropouts, which Section 7 discusses.

Notation. Safety parameters: r_{robot}, minimum safety margin $d_{\min}$, robustness buffer ϵ, composed minimum clearance $\rho_{\min} := r_{\text{robot}} + d_{\min} + \epsilon$. Controller parameters: lateral gain $k_\perp$, longitudinal cap $v_{\max}$. The streamwise coordinate ξ runs along a corridor center line $c(\xi)$ with unit tangent $\hat{t}(\xi)$ and unit normal $n(\xi)$; for a path in the tubular neighborhood of c, $y(\xi)$ is the scalar lateral offset.

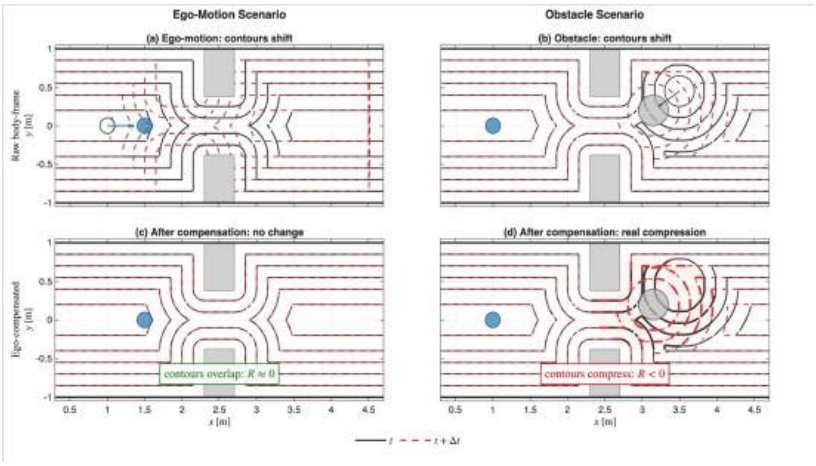


Figure 2: **Ego-motion vs. obstacle motion in $\dot{\Gamma}_{\mathbf{rel}}$.** Pure ego-motion shifts field contours; after subtracting the robot's own motion (lower panels) the contours overlap ($R \approx 0$) under ego-motion and compress ($R < 0$) under real obstacle motion. Closing is detected without instantiating the obstacle.

4 Method

The pipeline has three layers. Layer 1 builds an arrival-conditioned predicted clearance field from RGBD observations plus a single robot-frame relative field rate. Layer 2 synthesizes the navigation manifold on the predicted field via a two-stage solver and a stateful tracker. Layer 3 contracts toward the manifold with a manifold-DS controller and projects onto safety with a multi-boundary field-CBF whose boundary-motion term enters through $\partial_t \Gamma_{\text{curr}}$, so obstacle motion is handled without an obstacle list.

4.1 Layer 1: arrival-conditioned field prediction

Layer 1 maps the latest RGBD frame and pose into two field-level quantities: the robot-registered current clearance field $\Gamma_{\text{curr}}(x, t)$, fused from observation within the FOV and a two-layer field memory outside it; and the robot-frame relative field rate, computed only on the persistently known overlap $\Omega_{\cap}(t)$:

$$\dot{\Gamma}_{\text{rel}}(\xi, t) = \frac{\Gamma_r(\xi, t) - \Gamma_r(\xi, t - \Delta t)}{\Delta t}, \quad \xi \in \Omega_{\cap}(t), \tag{1}$$

and zero elsewhere. An EMA filter produces $\dot{\Gamma}_{\text{rel,sm}}$.

Two-layer field memory. A robot-centric memory that decays uniformly forgets static structure as quickly as it forgets the latest pedestrian, which destabilises the center line $c(\xi)$ under robot rotation. We separate the off-FOV memory into two world-frame layers on a shared 0.10 m grid, dynamically grown as the robot explores. The *static* layer $M_s(x, t)$ accumulates evidence with slow decay (per-step retention 0.999, $\tau_s \approx 750$ steps); the *dynamic* layer $M_d(x, t)$ accumulates fresh occupancy with fast decay (retention 0.5, $\tau_d \approx 1$ step). The off-FOV portion of Γ_{curr} is reconstructed from a distance transform on $\max(M_s, M_d)$. The static layer preserves $c(\xi)$ across robot rotations; the dynamic layer's fast decay prevents a moving obstacle from leaving a residual occupancy trace that the CBF would subsequently interpret as a closing boundary.

Arrival conditioning. The planning field is conditioned on the time the robot will arrive at each point. With travel-time horizon $h(x) = \min(d_{\text{geo}}(x_{\text{robot}}, x)/\bar{v}_{\text{acc}}(t), H_{\max})$,

$$\tilde{\Gamma}(x, t) = \Gamma_{\text{curr}}(x, t) + h(x)\, c(x, t)\, c_{\text{pred}}(x, t)\, \dot{\Gamma}_{\text{rel,sm}}(x, t), \tag{2}$$

where c is an observation-confidence decay and c_{pred} a trust decay in $h(x)$. The valid predicted clearance field is $\Gamma_{\text{pred}} = \text{bwdist}(\neg\{\tilde{\Gamma} \leq 0\}) \cdot \text{res}$. The construction answers "what will the clearance be when the robot

arrives at x," not "what is it now"; transient compression that resolves before arrival does not appear in Γ_{pred} (Appendix G).

4.2 Layer 2: manifold and solver

We define the navigation manifold $\gamma_t^* \subset \mathcal{F}_t^r$ as a smooth arc-length-parameterized curve with unit tangent $\tau_t(s)$ (Appendix G illustrates the ridge-dynamics view of γ_t^* as a ridge curve on the height function Γ). The manifold is required to satisfy six post-hoc verifiable properties: P1, robust traversability $\inf_s \Gamma(\gamma_t^*(s), t) \geq \rho_{\min}$; P2, conditional goal progress; P3, branch persistence across replans; P4, spatiotemporal C^1 regularity $\sup_s \|\gamma_t^*(s) - \gamma_{t+\Delta t}^*(s)\| \leq L\Delta t$; P5, positive reach exceeding the tracking-error bound; P6, controlled termination when no continuation satisfies P1–P5. The theoretical results depend on P1–P6, not on the solver.

Center-line plus offset. Within a corridor chart parameterized by $c(\xi)$, any candidate manifold in the tubular neighborhood reduces to a scalar offset:

$$\gamma_t^*(\xi) = c(\xi) + y^*(\xi, t)\, n(\xi), \quad y^* = y_{\text{ridge}} + \delta y^*, \tag{3}$$

where $y_{\text{ridge}}(\xi, t) = \arg\max_y \Gamma_{\text{pred}}(c(\xi) + yn(\xi), t)$ is the safety base. The offset representation inherits regularity from c and y (Appendix A).

Two-stage solver. The decoupling proposition (Section 5.1) implies safety becomes a pointwise box constraint $\delta y(\xi) \in [\delta_-(\xi), \delta_+(\xi)]$. Stage 1 extracts y_{ridge} and $[\delta_-, \delta_+]$ per station (a 1-D argmax and threshold query, no iteration). Stage 2 solves a banded QP for δy that penalizes curvature and tangent variation (proxy for P4–P5) and resists unnecessary branch switching (proxy for P3); for the common quadratic objective $J = \int w_1|\delta y - \delta y_{\text{ref}}|^2 + w_2|\delta y'|^2 + w_3|\delta y''|^2\, d\xi$ the Hessian is tridiagonal and solvable in $O(N)$ over $N \approx 15$–25 stations. The solver runs every $N_{\text{skip}} = 3$ control steps (the CBF runs every step), warm-started from the previous frame. Pseudocode is in Appendix B.

4.3 Layer 3: manifold-DS controller and field-CBF

Manifold-DS controller.

$$\dot{x} = -k_\perp\, e_\perp(x, t) + k_\parallel(t)\, \tau_t(\pi_t(x)) + u_{\text{CBF}}(x, t), \tag{4}$$

where $e_\perp = x - \gamma_t^*(\pi_t(x))$ is the transverse error and π_t is the unique projection (guaranteed by P5). The longitudinal gain is synthesized from the predicted field: $k_\parallel(t) = \text{clip}(v_{\text{cruise}}(\rho_{\text{ridge}}, \mathcal{M}_{\min}) + a_{\text{pred}}(t)\Phi_0,\, 0,\, v_{\max})$, with v_{cruise} monotone in clearance and the arrival-conditioned margin, and $a_{\text{pred}} \in [0, 1]$ the maximal feasible predictive commitment.

Anti-freeze progress floor. The controller and the CBF are both conservative; their composition can drop $\|u_{\text{nom}}\|$ to near zero in tight crossings. Before passing u_{nom} to the CBF, if the manifold is feasible and the tangent-projected magnitude $|u_{\text{nom}}^\top \tau_t|$ falls below $v_{\min}^{\text{progress}}$, we boost it to $v_{\min}^{\text{progress}}$ along τ_t. The CBF remains free to clip if any constraint would dip below its discrete-time buffer.

Multi-boundary field-CBF. The safety layer reasons about a field, not about object identities. A single-scalar barrier $h = \Gamma_{\text{curr}} - \Gamma_{\text{safe}}$ at the robot's position compresses all surrounding boundary information into one gradient; near a medial axis or a multi-agent sandwich the spatial gradient vanishes even though $\partial_t \Gamma < 0$, and the CBF perceives a closing boundary but has no preferred direction to project against. We avoid this without leaving the field formulation. Sample K near-boundary points $\{b_i\}_{i=1}^K$ from Γ_{curr} (cells with $\Gamma_{\text{curr}} \leq 2\Gamma_{\text{safe}}$ within a search radius, projected to their nearest boundary along $-\nabla\Gamma_{\text{curr}}$). For each b_i, define

$$h_i := \|x_{\text{robot}} - b_i\| - r_{\text{robot}} - d_{\min}, \quad \hat{n}_i := \frac{x_{\text{robot}} - b_i}{\|x_{\text{robot}} - b_i\|}, \tag{5}$$

and read the apparent closing rate from the field, $\dot{h}_i = -\partial_t \Gamma_{\text{curr}}(b_i, t)/\|\nabla\Gamma_{\text{curr}}(b_i)\|$. The QP is

$$u_{\text{CBF}} = \arg\min_u \|u - u_{\text{nom}}\|^2 \text{ s.t. } \hat{n}_i^\top u + \dot{h}_i + \alpha_{\text{CBF}}(h_i - d_{\text{buf},i}) \geq 0, \quad \forall i, \tag{6}$$

solved by a small active-set projection. Cancellation cannot occur because the K normals live in distinct half-planes by construction. Boundary samples come from the field; no object list, no semantic class, no tracker.

Guard zone plus material-derivative time-to-contact gate. Constraint i is committed to the QP only if $\|x_{\text{robot}} - b_i\| \le R_{\text{cbf}}$ or its time-to-contact is short, with TTC computed from the material derivative of Γ along the nominal command:

$$\text{TTC}_i := \frac{\Gamma_{\text{curr}}(b_i, t) - \Gamma_{\text{safe}}}{\max(0,\ -[\partial_t \Gamma_{\text{curr}}(b_i, t) + \nabla_x \Gamma_{\text{curr}}(b_i, t)^\top u_{\text{nom}}])}. \tag{7}$$

Using $\partial_t \Gamma$ alone misses robot-induced closing; the material-derivative form captures the case where the robot itself drives toward a static obstacle without estimating obstacle velocity. The gate eliminates the standing intervention an always-on CBF would impose in open corridors with distant boundaries.

Bounded adaptive buffer. Under discrete control with timestep Δt and worst-case relative velocity, the continuous-time CBF can graze the boundary and penetrate it on the next step. We inflate each h_i by

$$d_{\text{buf},i} := \text{clip}(\beta \cdot \max(0, -\dot{h}_i) \cdot \Delta t,\ d_{\text{buf}}^{\min},\ d_{\text{buf}}^{\max}), \tag{8}$$

hard-capped to prevent unbounded inflation that would steal performance on legitimately fast closing. With $d_{\text{buf}}^{\max} = 0.25$ m and $\Delta t = 0.2$ s this matches the closing geometry of a 1.25 m/s walker without becoming permanent.

4.4 Manifold continuation: translate–repair–optimize–validate

Running the two-stage solver from scratch every frame wastes information: under P4 the manifold barely moves between frames, so the prior-frame solution is a near-optimal warm start. The tracker performs four operations per frame. *Translate.* Advance $\gamma^*_{t-\Delta t}$ by the arc length the robot traveled. *Repair.* For each station, search inside a local adaptive tube around the predicted offset, with a branch-switch hysteresis penalty; the first unrepairable station triggers regrowth of the suffix from the last valid station. *Optimize.* An extended QP reconciles the data term, spatial smoothness, homotopy preference, and a temporal term. *Validate, or fall back.* The result is checked against five categories of desync trigger: cold start, *task reset* (goal change), *sensing failure* (pose jump beyond threshold, lateral drift beyond threshold), *regular continuation failure* (tube collapse, tangent disagreement, FOV exit, QP overshoot), or successful continuation. The per-frame diagnostic struct records the outcome with `fallback_category` in {none, task_reset, sensing_failure, regular_continuation_failure}. Section 6 reports tracking utilization, temporal deviation, and per-category fallback rate from this struct. The adaptive tube radius enforces a per-station bound on $|\Delta y_i|$ that is the discrete-time analogue of the P4 constant L in Section 5.2.

5 Analysis

We state three results. Proof sketches appear inline; full versions are in Appendix A.

5.1 Safety–performance decoupling

Proposition 1 (Decoupling)**.** *In a non-degenerate single-passage corridor chart, let $y_{ridge}(\xi) \in \arg\max_y \rho(\xi, y)$ and suppose $\partial_{yy}\rho(\xi, y_{ridge}(\xi)) < 0$. With $\delta y(\xi) := y(\xi) - y_{ridge}(\xi)$, the safety constraint $\rho(\xi, y(\xi)) \ge \rho_{\min}$ is equivalent to the pointwise box $\delta y(\xi) \in [\delta_-(\xi), \delta_+(\xi)]$.*

Sketch. The strict concavity makes the upper level set an interval $[y_-(\xi), y_+(\xi)]$; subtracting y_{ridge} gives the box. Any performance objective is now a functional of δy subject to that box; safety does not appear in the performance solve. This is strictly stronger than a weighted-sum formulation, whose safety guarantee degrades continuously as the performance weight grows.

5.2 Practical-stability bound

Definition 1. $v_\gamma^\perp(t) := \sup_s \|(I - \tau_t(s)\tau_t(s)^\top)\partial_t \gamma_t^*(s)\|$.

Proposition 2 (Tracking-error bound)**.** *Under (4) with P1–P5, the transverse tracking error obeys* $\|e_\perp(t)\| \le e^{-k_\perp t}\|e_\perp(0)\| + \sup_{\tau \le t} v_\gamma^\perp(\tau)/k_\perp$.

Sketch. The transverse component of (4) is a contracting linear system driven by the manifold's normal velocity. P4 bounds the driving term, so the steady-state error is bounded by $L/k_\perp$. Empirically, the per-frame temporal-deviation surrogate $\sup_s \|\gamma_t^*(s) - \gamma_{t-\Delta t}^*(s)\|$ is bounded by ≈ 0.094 m (mean) and ≈ 0.28 m (p95) across all six scenarios at $\Delta t = 0.2$ s (Section 6), consistent with $L \lesssim 0.5$ m/s.

5.3 Controller-sufficient statistic and CBF forward invariance

Definition 2. $\mathcal{P}[\Gamma] = (\rho_{\text{ridge}}, \dot\rho_{\text{ridge}}^{\text{rel}}, y_{\text{ridge}}, \dot y_{\text{ridge}}, \tau, \delta_-, \delta_+)$, all 1-D functions of ξ along γ_t^*.

Proposition 3 (Controller-sufficient statistic)**.** *For controller (4) with the longitudinal law of Section 4.3 and P3,* $(\mathcal{P}[\Gamma], a_{pred})$ *together with the local field neighborhood at the robot is sufficient to compute* $u(t)$ *and all predictive-feasibility decisions.*

Implication. The 2-D clearance field is never accessed after projection, so the only quantities that must be accurately predicted are 1-D functions of ξ. Prediction complexity scales with manifold length, not workspace area. This is the structural reason why field-level prediction is tractable on embedded compute. The field-CBF reads Γ_{curr} and $\partial_t \Gamma_{\text{curr}}$ at the boundary samples but does not require any prediction.

Forward invariance under the multi-boundary field-CBF. If the QP (6) is feasible and the closed-loop trajectory is absolutely continuous in $\Omega_\cap$, standard CBF arguments give forward invariance of $\bigcap_i \{x : h_i \ge d_{\text{buf},i}\}$ over the active set. A closing boundary at b_i ($\partial_t \Gamma_{\text{curr}}(b_i, \cdot) < 0$) decreases the slack of constraint i even when the robot is stationary, forcing u_{CBF} to retreat along $\hat n_i$ at the apparent rate $-\partial_t\Gamma/\|\nabla\Gamma\|$. The reaction is obstacle-motion-aware without any obstacle-velocity estimate, and the multi-boundary form supplies a non-degenerate retreat direction even when $\nabla\Gamma$ cancels at the robot's position. The argument breaks at the FOV boundary: a region that enters $\Omega_\cap$ for the first time has no history to differentiate against, and a fast obstacle traversing that boundary can cross the safe set within a single control step. Section 7 characterises the regime in which this constraint binds.

6 Experiments

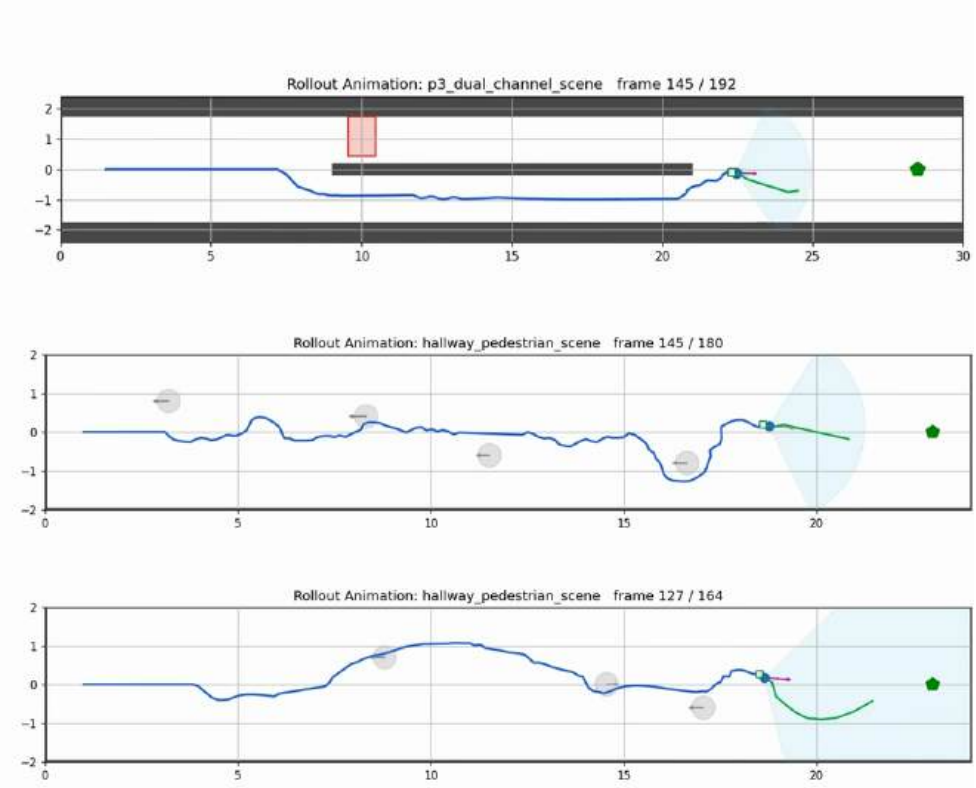


Figure 3: **Three successful 2-D rollouts.** Top: dual-channel passage, the robot selects the lower channel of a split corridor without instantiating either channel as an obstacle. Middle: hallway with four opposing pedestrians, the manifold continues through every encounter with no collisions and no sensing-failure fallbacks. Bottom: hallway with pedestrians moving in random directions, the trajectory winds around four randomly-placed walkers. Cyan cone: camera FOV; small green curve ahead of the robot: navigation manifold γ_t^*; green pentagon at the right: goal.

Figure 3 shows three representative 2-D rollouts. We test seven claims quantitatively. (Q1) The pipeline reaches a static-environment goal in a single deterministic sanity run. (Q2) Manifold continuation dominates in normal operation, and `sensing_failure` serves as the empirical diagnostic for Assumption 1. (Q3) Temporal deviation remains within the P4 bound. (Q4) A point-DS stress baseline isolates the manifold layer's safety contribution upstream of the shared multi-boundary field-CBF. (Q5) Per-layer compute meets a candidate-platform budget. (Q6) The safety-filter design path is justified by a stepwise ablation from object-CBF through the final gated multi-boundary form with bounded buffer. (Q7) Each load-bearing pipeline component (continuation tracker, two-layer field memory, multi-boundary CBF) is verified to be necessary, with a classical DWA local planner as a sensing-agnostic reference.

Setup and baselines. Numbers come from the Python reference implementation; the configuration is the one tabulated in Appendix B. *Point-DS* is (4) with γ_t^* replaced by a single point attractor and $k_\parallel$ replaced by a constant goal-attraction gain; both methods share the identical multi-boundary field-CBF, so any safety gap is attributable to the manifold layer upstream. *DWA* is a classical local planner over the same sensor input but no DS / manifold / field-CBF layer.

Scenarios and metrics. Six scenarios, 10 paired seeds each: corridor crossing (two pedestrians), dynamic hallway (one opposing walker), occluded boundary (a walker partially hidden), FOV-limited blind corner,

Table 1: **Q4: point-DS stress baseline vs. ours.** 10 paired seeds; both methods share the identical multi-boundary field-CBF, so the manifold is the only upstream difference. Bold = winner per metric.

	Collision (%) ↓		Min. clearance (m) ↑		Jerk RMS ↓	
Scenario	point-DS	ours	point-DS	ours	point-DS	ours
Corridor crossing	0.0 ± 0.0	0.0 ± 0.0	**0.450**	0.270	**0.029**	0.538
Dynamic hallway	50.0 ± 50.0	**0.0 ± 0.0**	0.150	**0.279**	4.43	**1.34**
Occluded boundary	0.0 ± 0.0	0.0 ± 0.0	**0.350**	0.200	5.31	**0.889**
FOV limited	100.0 ± 0.0	100.0 ± 0.0	0.050	0.050	**0.051**	0.948
Long corridor (dyn.)	10.0 ± 30.0	**0.0 ± 0.0**	**0.295**	0.271	4.79	**1.38**
Cluttered (5 occl.)	0.0 ± 0.0	0.0 ± 0.0	**0.295**	0.250	5.19	**2.49**

Note. Paired comparison across the six scenarios reported, 10 seeds each. Both methods share the identical multi-boundary field-CBF, so any safety gap is attributable to the manifold layer upstream. Bold = winner per metric. Full table with standard deviations on every cell is in Appendix E.

long corridor (20 m with one opposing walker), and cluttered (5 static occluders with one crossing walker). Safety metrics in the body: tracking utilization (M1), temporal deviation (M2), per-category fallback rate (M3), safety margin (M4), collision rate (M5), RMS jerk (M6). Supplementary metrics are reported in Appendix E.

Q1: static-environment baseline. The full pipeline reaches a 23 m static-corridor goal in 43.4 s (single-seed deterministic run, all contract checks pass); five tracker integration tests verify the P3 and P4 invariants; all 58 unit tests pass.

Q2 and Q3: tracker invariants. Across the six scenarios, `sensing_failure` is exactly 0% on every seed; continuous boundary deformation (Assumption 1) holds on persistently known support even on the occluded-boundary scenario built to stress it. Tracker utilisation ranges from 46.7% (FOV limited, where the relevant geometry enters the FOV mid-rollout) to 98.7% (occluded boundary, where the static memory layer holds the center line through the occluder). Temporal-deviation mean stays ≤ 0.094 m and p95 ≤ 0.28 m at $\Delta t = 0.2$ s, consistent with the P4 bound $L\,\Delta t$ at the design Lipschitz constant. Full Q2 and Q3 tables (per-category fallback rate, p95 temporal deviation, minimum clearance) are in Appendix E.

Q4: manifold ablation against point-DS. The manifold layer absorbs head-on closing geometry on dynamic hallway (collision rate 50% → 0%; mean minimum clearance 0.279 m vs. 0.150 m) and long corridor (10% → 0%), although both methods share the same safety filter. On the three static-clutter scenarios neither method collides; point-DS exhibits a slightly higher *instantaneous* minimum clearance because its constant goal-attraction term keeps it near the geometric center of the corridor, whereas our method tracks the ridge of the arrival-conditioned predicted field Γ_{pred} and commits earlier to the lateral side where the obstacle will not be at arrival time. The lower instantaneous clearance therefore reflects a measurement against the current obstacle position rather than the predicted one; under fast head-on closing, the same prediction layer steers our method to a safe lateral offset before the obstacle reaches the robot, which accounts for the lower collision rate despite the smaller instantaneous margin. On the FOV-limited blind corner both methods collide: the obstacle enters the persistently-known overlap with no $\partial_t\Gamma$ history, an information-theoretic limit shared by every FOV-restricted controller rather than a difference between the two upstream nominal commands.

Q6: safety-filter design path. The multi-boundary form, the bounded buffer, and the TTC plus guard-zone gate were obtained through a sequence of incremental ablations. Table 2 reports the M5 collision-rate trajectory on the dynamic-hallway diagnostic as the safety filter is rebuilt one design choice at a time, holding the rest of the pipeline fixed. *Object-CBF to single-scalar field-CBF*: the field formulation removes the obstacle-list bookkeeping but inherits a medial-axis gradient cancellation the legacy form does not have. *Single-scalar to multi-boundary*: drops the baseline from 60% to 10% by restoring directional authority

across K boundary samples. *Material derivative in TTC*: catches the case where the robot itself drives into a static obstacle; $\partial_t \Gamma$ alone misses robot-induced closing. *Bounded buffer*: an unbounded adaptive buffer overshoots on fast head-on closing; capping at $d_{\text{buf}}^{\max} = 0.25$ m is the smallest value that keeps the head-on case stable while letting open-corridor cruise stay unconstrained. *Two-layer SLAM-like field memory*: the final 20% → 0% drop comes from a stable center line; the tracker holds its solution through the crossing, so the nominal command arriving at the CBF is already deflected away from the closing wall.

Table 2: **Q6: safety-filter design path on dynamic hallway** (M5 collision rate, 10 seeds, $\rho_{\min} = 0.2$ m, $\Delta t = 0.2$ s). Each row replaces a single design choice while holding the rest of the pipeline fixed; the terminal row is the configuration used in Section 6.

Safety filter (this column varies)	point-DS	ours
Object-CBF over a finite obstacle list (legacy)	60%	30%
Single-scalar field-CBF $h = \Gamma_{\text{curr}} - \Gamma_{\text{safe}}$ at robot	60%	40%
Multi-boundary field-CBF, K samples, no gating	10%	20%
+ TTC+guard-zone gate, $\partial_t \Gamma$ only	80%	50%
+ material derivative in TTC	30%	30%
+ bounded buffer $d_{\text{buf}}^{\max} = 0.25$ m	50%	20%
+ **two-layer SLAM-like field memory (final)**	**50%**[†]	**0%**

[†] point-DS does not consume the SLAM memory (u_{nom} has no center-line dependency); the 50% figure is the bounded-buffer plus gated CBF over a point attractor, shown as a fixed-CBF reference.

Q7: component ablation. We also switch off each load-bearing component independently (*no continuation*: Tier-2 tracker off; *FOV only*: two-layer memory off; *single-scalar CBF*: `mb_K` = 1) and run a classical DWA local planner as a sensing-agnostic reference (Table 3).

Table 3: **Q7: component ablation across four base scenarios** (10 seeds per scenario, 60 s). M1 = tracker utilisation, M5 = collision rate.

	Corridor crossing		Dynamic hallway		Occluded boundary		FOV limited	
Variant	M1 ↑	M5 ↓	M1 ↑	M5 ↓	M1 ↑	M5 ↓	M1 ↑	M5 ↓
Ours (full)	57%	**0%**	72%	**0%**	99%	**0%**	47%	100%
– continuation (Tier-2 off)	0%	0%	0%	0%	0%	0%	0%	100%
– two-layer memory (FOV only)	73%	0%	71%	20%	58%	0%	52%	100%
– multi-boundary CBF (`mb_K` = 1)	59%	0%	69%	10%	98%	0%	74%	100%
point-DS stress baseline	100%	0%	100%	50%	100%	0%	100%	100%
DWA local planner	100%	0%	100%	90%	100%	0%	100%	100%

Note. M1 = tracker utilisation (%); M5 = collision rate (%). Lower M5 is better; M1 = 0 for variants with no continuation tracker by construction. The FOV-limited column is the information-theoretic limit and every variant collides.

The full stack is the only variant achieving 0% collision rate on the three non-FOV-limit base scenarios. Collapsing the CBF to a single boundary reintroduces the medial-axis gradient cancellation on dynamic hallway; disabling the two-layer memory removes the static layer that stabilises the center line; the DWA reference collides on 90% of dynamic-hallway seeds because it lacks a predictive-geometry layer.

Q5: compute. On the host platform the manifold solver dominates (5.4 ms mean) and everything else, including the multi-boundary CBF QP, finishes below 0.3 ms; the combined per-frame mean at 73% tracker utilisation is ≈ 5.8 ms, under the 200 ms control period. Full per-layer table in Appendix F.

7 Limitations

The continuous-boundary assumption (Assumption 1) precludes extrapolation across long occlusions; environments with semi-transparent obstacles or instantaneously appearing obstacles are out of scope. The

field-CBF inherits the FOV's information limit: a point entering the persistently-known overlap for the first time has $\partial_t \Gamma = 0$ by construction, so an obstacle that traverses the FOV boundary in less than the worst-case reaction time can force a collision on any FOV-restricted controller (the failure mode shared by both methods on the FOV-limited blind corner). The mitigation is deployment-side (lower $v_{\max}$, larger Γ_{safe}, wider sensor) rather than algorithmic. The manifold tracks the ridge of Γ_{pred} rather than the geometric center of the corridor, which yields a slightly lower minimum clearance in open corridors with slow obstacles; the value remains above $\rho_{\min}$, so the gap is a trade between conservativeness and centrality, not a safety violation. The offset representation (3) is valid inside a single corridor chart; multi-chart topology (branch merges, splits, rooms with islands) requires re-chartering, and extensions to differential-drive or bicycle kinematics require a control-input transformation that we have not validated.

References

[1] Aaron D. Ames, Xiangru Xu, Jessy W. Grizzle, and Paulo Tabuada. Control barrier function based quadratic programs for safety critical systems. *IEEE Transactions on Automatic Control*, 62(8):3861–3876, 2017. doi: 10.1109/TAC.2016.2638961.

[2] Max Asselmeier, Dhruv Ivanovic, Bhaskar Dey, and Patricio A. Vela. Dynamic gap: Safe gap-based navigation in dynamic environments. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025.

[3] Marvin Harms, Mihir Kulkarni, Nikhil Khedekar, Martin Jacquet, and Kostas Alexis. Neural control barrier functions for safe navigation. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024.

[4] Lukas Huber, Jean-Jacques Slotine, and Aude Billard. Avoiding dense and dynamic obstacles in enclosed spaces: Application to moving in crowds. *IEEE Transactions on Robotics*, 38(5):3113–3132, 2022. doi: 10.1109/TRO.2022.3160771.

[5] S. Mohammad Khansari-Zadeh and Aude Billard. Learning stable nonlinear dynamical systems with Gaussian mixture models. *IEEE Transactions on Robotics*, 27(5):943–957, 2011. doi: 10.1109/TRO.2011.2159412.

[6] Gennaro Notomista and Matteo Saveriano. Safety of dynamical systems with multiple non-convex unsafe sets using control barrier functions. *IEEE Control Systems Letters*, 6:1136–1141, 2022. doi: 10.1109/LCSYS.2021.3089097.

[7] Harish Ravichandar, Iman Salehi, and Ashwin P. Dani. Learning partially contracting dynamical systems from demonstrations. In *Proceedings of the 1st Annual Conference on Robot Learning (CoRL)*, volume 78 of *Proceedings of Machine Learning Research*, pages 369–378. PMLR, 2017.

[8] Tim Salzmann, Boris Ivanovic, Punarjay Chakravarty, and Marco Pavone. Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data. In *Computer Vision – ECCV 2020*, volume 12363 of *Lecture Notes in Computer Science*, pages 683–700. Springer, 2020. doi: 10.1007/978-3-030-58523-5_40.

[9] Martin Schonger, Hugo T. M. Kussaba, Lingyun Chen, Luis Figueredo, Abdalla Swikir, Aude Billard, and Sami Haddadin. Learning barrier-certified polynomial dynamical systems for obstacle avoidance with robots. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 17774–17780. IEEE, 2024. doi: 10.1109/ICRA57147.2024.10610828.

[10] Sunan Sun and Nadia Figueroa. SE(3) linear parameter varying dynamical systems for globally asymptotically stable end-effector control. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024.

[11] Yanan Wang, Yaobin Tian, Jiawei Chen, Kun Xu, and Xilun Ding. A survey of visual SLAM in dynamic environment: The evolution from geometric to semantic approaches. *IEEE Transactions on Instrumentation and Measurement*, 73:1–21, 2024. doi: 10.1109/TIM.2024.3420374.

[12] Moritz Werling, Julius Ziegler, Sören Kammel, and Sebastian Thrun. Optimal trajectory generation for dynamic street scenarios in a Frenet frame. In *2010 IEEE International Conference on Robotics and Automation (ICRA)*, pages 987–993, 2010. doi: 10.1109/ROBOT.2010.5509799.

[13] Mandy Xie, Karl Van Wyk, Ankur Handa, Stephen Tyree, Dieter Fox, Harish Ravichandar, and Nathan D. Ratliff. Neural geometric fabrics: Efficiently learning high-dimensional policies from demonstration. In *Proceedings of the 6th Conference on Robot Learning (CoRL)*, volume 205 of *Proceedings of Machine Learning Research*, pages 1355–1367. PMLR, 2022.

[14] Zhanteng Xie and Philip Dames. SCOPE: Stochastic cartographic occupancy prediction engine for uncertainty-aware dynamic navigation. *IEEE Transactions on Robotics*, 41:4139–4158, 2025. doi: 10.1109/TRO.2025.3578234.

[15] Chao Yu, Zuxin Liu, Xin-Jun Liu, Fugui Xie, Yi Yang, Qi Wei, and Qiao Fei. DS-SLAM: A semantic visual SLAM towards dynamic environments. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1168–1174, 2018. doi: 10.1109/IROS.2018.8593691.

A Proofs

Proof sketches below outline the arguments; the full versions are in the supplementary technical report shipped with the code release.

A.1 Proof of Proposition 1 (Safety–Performance Decoupling)

Sketch. The cross-section clearance profile $\rho(\xi, y)$ is the restriction of Γ to the cross section at ξ. By definition $y_{\text{ridge}}(\xi) \in \arg\max_y \rho(\xi, y)$. The strict concavity assumption $\partial_{yy}\rho < 0$ at the ridge implies y_{ridge} is a unique local maximizer. The set $\{y : \rho(\xi, y) \geq \rho_{\min}\}$ is the upper level set of a strictly concave function, hence an interval $[y_-(\xi), y_+(\xi)]$. Subtracting $y_{\text{ridge}}(\xi)$ gives the box $[\delta_-(\xi), \delta_+(\xi)]$. Any path with $\delta y(\xi) \in [\delta_-(\xi), \delta_+(\xi)]$ pointwise satisfies $\rho(\xi, y(\xi)) \geq \rho_{\min}$. The performance objective is now any functional of δy subject to the box; the safety geometry $\rho_{\min}$ does not appear in the performance solve. □

A.2 Proof of Proposition 2 (Tracking-Error Bound)

Sketch. The transverse component of (4) reduces, under the projection π_t, to

$$\dot{e}_\perp = -k_\perp e_\perp + (\text{transverse component of } \partial_t \gamma_t^*).$$

The driving term has magnitude at most $v_\gamma^\perp(t)$. Standard scalar comparison gives

$$\|e_\perp(t)\| \leq e^{-k_\perp t}\|e_\perp(0)\| + \int_0^t e^{-k_\perp(t-\tau)} v_\gamma^\perp(\tau) d\tau \leq e^{-k_\perp t}\|e_\perp(0)\| + \frac{\sup_{\tau \leq t} v_\gamma^\perp(\tau)}{k_\perp}.$$

P4 bounds $v_\gamma^\perp$, hence the steady-state error is finite. □

A.3 Proof of Proposition 3 (Controller-Sufficient Statistic)

Sketch. The control output is

$$u(t) = -k_\perp(x - \gamma_t^*(\pi_t(x))) + k_\parallel(t)\, \tau_t(\pi_t(x)) + u_{\text{CBF}}(x, t).$$

The first two terms read only $\gamma_t^*(\pi_t(x))$ and $\tau_t(\pi_t(x))$, both elements of $\mathcal{P}[\Gamma]$. The longitudinal gain $k_\parallel$ uses $\rho_{\text{ridge}}(\pi_t(x), t)$, $\mathcal{M}_{\min}(t)$, and $a_{\text{pred}}(t)$; the first two are in $\mathcal{P}[\Gamma]$ and a_{pred} is the maximal scalar commitment. The multi-boundary field-CBF u_{CBF} (6) reads Γ_{curr} in the boundary-sample neighborhood and $\partial_t \Gamma_{\text{curr}}$ at those samples, both maintained by Layer 1; this is local field information, not a manifold-projection element. The performance residual δy reads $\delta_\pm$, y_{ridge}, $\dot{y}_{\text{ridge}}$, all elements of $\mathcal{P}[\Gamma]$. Therefore $(\mathcal{P}[\Gamma], a_{\text{pred}})$ together with the local field neighborhood at the robot is sufficient. □

B Implementation Details

Layer 1: field and prediction. EMA smoothing factor $\alpha_{\text{EMA}} = 0.3$; observation confidence decay $\tau_{\text{decay}} = 2.5$ s; prediction trust decay $\tau_{\text{pred}} = 1.5$ s; maximum horizon $H_{\max} = 2$ s; nominal robot speed $v_{\text{robot,nom}} = 0.4$ m/s; minimum exec speed floor $v_{\text{exec,floor}} = 0.05$ m/s.

Layer 1: two-layer field memory. Memory grid resolution 0.10 m, dynamically grown world-frame bounds. Static layer per-step retention 0.999 ($\tau_s \approx 750$ steps); dynamic layer retention 0.5 ($\tau_d \approx 1$ step). Off-FOV reconstruction uses $\max(M_s, M_d)$ as occupancy then a distance transform on the same grid.

Layer 2: manifold solver. $N_{\text{stations}} = 21$ (rear 6, forward 14); cross-section sampling $M_{\text{cross}} = 31$ with half-width 1 m; $\rho_{\min} = 0.2$ m; solver-skip $N_{\text{skip}} = 3$; forward horizon $L_{\text{horizon}} = 4$ m; rear horizon $L_{\text{rear}} = 1.2$ m; smoothness weight $w_{\text{smooth}} = 0.5$; homotopy weight $w_{\text{homotopy}} = 0.2$; rear-observation minimum confidence 0.3.

Layer 2: manifold tracker. Tracker fallback thresholds: dynamic-step fallback $1.5\,L_{\text{horizon}}/N_{\text{fwd}}$; lateral fallback 0.40 m; tangent-fallback cosine 0.30; pose-jump fallback 0.50 m; goal-jump fallback 1.0 m.

Layer 3: controller and progress floor. $k_\perp = 4$; $\Phi_0 = 0.2$; $v_{\max} = 0.5$ m/s; cruise reference clearance $s_{\rho,\text{ref}} = 0.4$ m; margin saturation $[s_{\text{low}}, s_{\text{high}}] = [0, 0.15]$; deceleration $a_{\text{decel}} = 0.5$ m/s^2; control delay $dt_{\text{delay}} = 0.05$ s; heading gain $k_{\text{heading}} = 2$; command frame: body. Tangent-projected anti-freeze progress floor $v_{\min}^{\text{progress}} = 0.10$ m/s.

Layer 3: multi-boundary field-CBF. $\alpha_{\text{CBF}} = 2$; $\Gamma_{\text{safe}} = r_{\text{robot}} + d_{\min} = 0.12 + 0.10 = 0.22$ m. Boundary-sample radius $R_{\text{sample}} = 0.7$ m; guard zone $R_{\text{cbf}} = 0.30$ m; TTC threshold $T_{\text{cbf}} = 0.40$ s from material derivative $d\Gamma_{\text{nom}}/dt = \partial_t \Gamma + \nabla_x \Gamma^\top u_{\text{nom}}$ (7). Bounded buffer (8): $\beta = 1$, $d_{\text{buf}}^{\min} = 0.05$ m, $d_{\text{buf}}^{\max} = 0.25$ m, $\Delta t = 0.2$ s. Active-set QP for projection; anti-degeneracy fallback installs a manifold-tangent normal or an emergency stop if the spatial gradient cancels and a non-trivial cone of boundary normals is not available.

Robot and sensor. $r_{\text{robot}} = 0.12$ m; actuator caps $\max v_x = 0.75$ m/s, $\max v_y = 0.5$ m/s, $\max \omega_z = 1.2$ rad/s. Simulated sensor: FOV 120°; range 2.5 m; field resolution 0.15 m; occlusion modeling enabled.

Algorithm 1 Two-stage manifold solver

Input: predicted field Γ_{pred}, active center line $c(\xi)$, warm-start $\delta y_{t-\Delta t}$
Output: discrete manifold $\{\gamma_i\}_{i=1}^N$ satisfying P1–P6
for $i = 1, \ldots, N$ **do** ▷ Stage 1: ridge extraction
  $y_{\text{ridge}}(\xi_i) \leftarrow \arg\max_y \Gamma_{\text{pred}}(c(\xi_i) + y\, n(\xi_i), t)$
  $[\delta_-(\xi_i), \delta_+(\xi_i)] \leftarrow \{y - y_{\text{ridge}}(\xi_i) : \rho(\xi_i, y) \geq \rho_{\min}\}$
end for
$\delta y^* \leftarrow \arg\min_{\delta y \in [\delta_-, \delta_+]} J_{\text{smooth}}[\delta y] + J_{\text{homotopy}}[\delta y]$
$\gamma_i \leftarrow c(\xi_i) + (y_{\text{ridge}}(\xi_i) + \delta y^*(\xi_i))\, n(\xi_i)$ for $i = 1, \ldots, N$
return $\{\gamma_i\}_{i=1}^N$

Two-stage solver pseudocode.

C Reproducibility

Code release. All algorithm code (Python reference, MATLAB algorithm package, ROS 2 C++ skeleton), every scenario file, every per-seed metrics file, the experiment manifest, the table-generation scripts, and the MLflow run logs ship with the anonymized submission and will be released without anonymization upon acceptance.

Static long-corridor sanity (Q1). Single-seed deterministic MATLAB run on a 24 m static corridor: start $(0.5, 0, 0)$, target $(23.0, 0)$, tolerance 0.35 m, duration 50 s, $\Delta t = 0.2$ s. Final pose $(22.900, 0, 0)$, arrival time 43.4 s, all six contract checks pass.

Six dynamic scenarios. Entry point is the Python rollout driver, parameterized by scenario name and algorithm (`slam_ds` or `point_ds`). For each of the six scenarios, both algorithms are run on seeds 0–9, then aggregated to per-scenario JSON; a separate script emits the paper tables.

Random-seed schedule. Seeds 0–9 are used for every paired scenario; the same seed produces the same walker timing and lateral-offset draws for both algorithms, so comparisons are paired. Algorithm-internal randomness is seeded from the same seed.

Hardware platform. The reported numerical results are simulation-only. The deployment target is an XLeRobot-class omniwheel base with Jetson onboard compute, an Intel RealSense D455 RGB-D camera, and the depth-to-occupancy-grid front-end of Appendix D. A Gazebo closed-loop test stack mirrors the procedural-simulator scenarios but is not the source of any number reported here.

D Deployment Pipeline: Gazebo, Hardware, and the Occupancy-Grid Front-End

This appendix documents the simulation environment used for closed-loop integration testing (beyond the procedural rollouts that produced the numerical tables) and the deployment-side occupancy-grid front-end that feeds Γ_{curr} from a real RGBD sensor. The paper's numerical evidence (Section 6) is from the synthetic-sensor simulator; the contents of this appendix are deployment artifacts that the open-source release ships with but does not yet supply numerical results for.

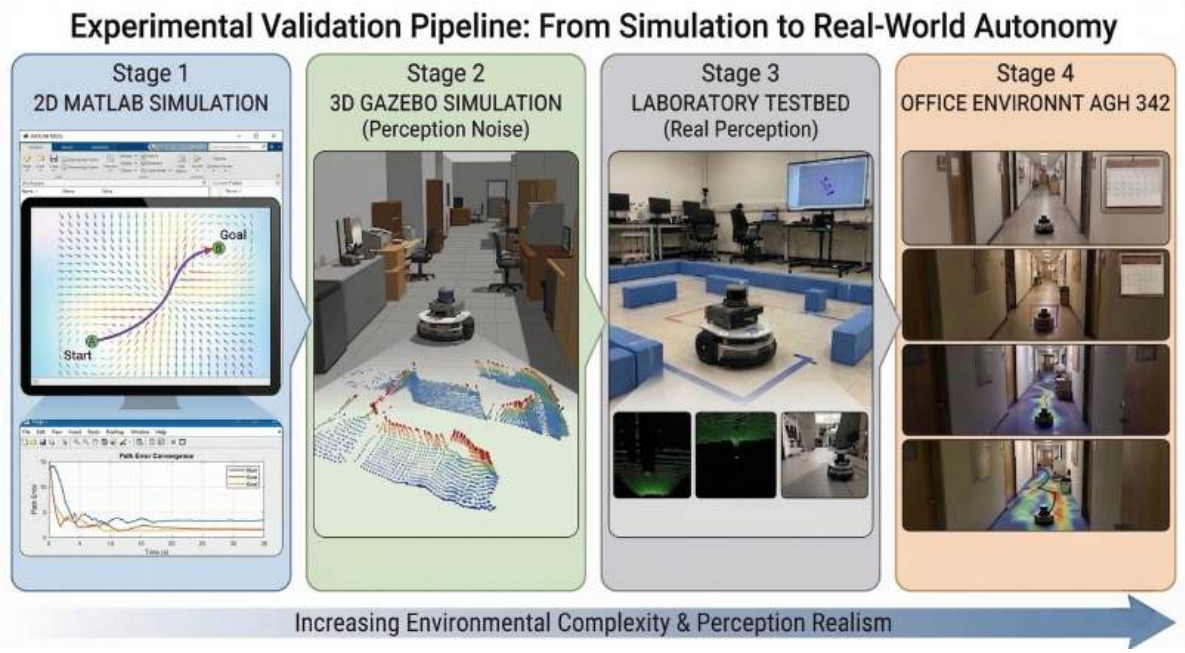
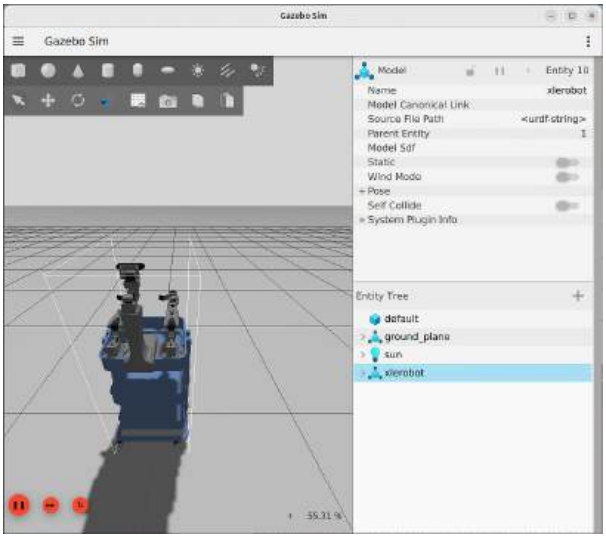


Figure 4: **Experimental validation pipeline from simulation to real-world autonomy.** The procedural simulator (Stage 1) produced the numerical evidence in Section 6; Stage 2 Gazebo and Stage 3 lab testbed are integration platforms; Stage 4 office deployment is the eventual target. Stages 2–4 are out of scope for the present paper.

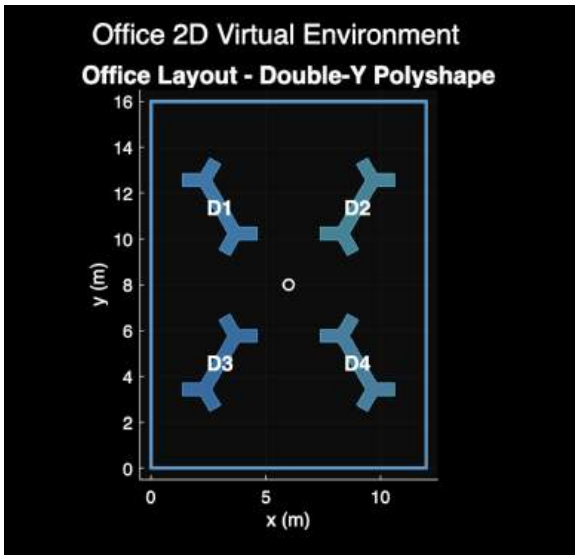
D.1 Gazebo simulation worlds

A Gazebo-based closed-loop test stack is built on the XLeRobot platform model (Figure 5): an omnidirectional mobile base, two arm assemblies, grippers, and an RGB-D camera mount. The mobile base is controlled through a unified ROS velocity-command interface; the high-level navigation pipeline publishes planar velocity commands and a lower-level controller maps them to the omni-wheel base. Two simulator backends are supported: a physics-based omni-wheel backend that exposes wheel–ground contact artifacts, and an ideal planar backend that isolates navigation behavior from contact noise.

Five corridor worlds mirror the procedural-simulator scenarios visually and physically: a long corridor, a corridor split by a center wall (passage-selection), a two-channel corridor with static passage selection, a door-like dynamic obstacle, and a long corridor with moving cylindrical pedestrian obstacles. The Gazebo stack bridges sensor data to ROS topics: RGB image, depth image, point cloud, camera calibration, robot state, odometry, and base controller interface.



Gazebo simulation with the XLeRobot platform.



2D MATLAB office layout (procedural simulator).

Figure 5: **Two of the four validation environments.** The procedural simulator (right) is the source of all numerical results in this paper; Gazebo (left) is the closed-loop integration platform.

D.2 Hardware platform (intended deployment target)

The deployment target is an XLeRobot-class platform: NVIDIA Jetson Nano onboard compute (Jetson Orin Nano Super for production), an omnidirectional mobile base with three omni wheels, two 7-DOF robotic arms, and an Intel RealSense D455 RGB-D camera. The hardware platform is built for closed-loop sim-to-real evaluation against the Gazebo scenarios. The main hardware perception component is the RealSense-based occupancy-grid pipeline of Section D.3. Full autonomous hardware navigation is not closed in the present paper; the missing component is calibrated odometry for the omnidirectional base. Without calibrated odometry, the pose estimate against the local occupancy grid is too noisy to safely deploy the DS controller and CBF in closed loop.

D.3 Depth-to-occupancy-grid front-end

The hardware and Gazebo front-end converts a rectified depth image into a 2-D occupancy grid that feeds Layer 1's Γ_{curr} via the existing camera-to-current-field module. The pipeline is summarized below.

Table 4: RealSense depth-filter parameters (current deployment).

Stage	Parameter	Value
Decimation	magnitude m	3
Spatial filter	magnitude / smooth-α / smooth-δ / holes-fill	4 / 0.25 / 30 / 2
Temporal filter	smooth-α / smooth-δ / persistency	0.40 / 20 / 3
Hole filling	mode	1
Disparity-domain filtering	enabled	true

Inputs. A 16-bit depth image $D \in \mathbb{N}^{H_I \times W_I}$ in millimeters with intrinsics $K = \mathrm{diag}(f_x, f_y, 1)$ plus principal point (c_x, c_y). The current deployment uses $f_x = f_y = 434.27$, $(c_x, c_y) = (431.77, 240.45)$, 30 FPS, depth units 1 mm.

Pipeline.

1. Decode depth and apply a RealSense-style filter chain: decimation → disparity transform → spatial filter → temporal filter → inverse disparity transform → hole filling. The composition is $D'_t = F_{\text{hole}} \circ T_{\text{disp}}^{-1} \circ F_{\text{temporal}} \circ F_{\text{spatial}} \circ T_{\text{disp}} \circ F_{\text{decim}}(D_t)$. Filter parameters (Table 4) follow the RealSense vendor defaults.
2. Project each valid depth pixel to a 3-D camera-frame point via the pinhole model.
3. Apply a vertical-slab filter $y \in [y_{\min}, y_{\max}]$ (current deployment: $[-0.1, 0.1]$ m) so that only points near the navigation plane contribute to the 2-D grid.
4. Rasterize the surviving 3-D points into an (x, z) grid with cell resolution $r = 0.05$ m, grid bounds $x \in [-6, 6]$ m, $z \in [0, 6]$ m. A cell is marked occupied when at least $n_{\min}$ returns land in it, then suppressed if its 8-connected component has fewer than N_{conn} cells (current: $N_{\text{conn}} = 8$).
5. Mark free space via Bresenham ray tracing from the camera origin to every hit endpoint and to every no-hit ray's grid boundary; preserve cells *behind* an occupied endpoint as unknown.
6. Assign the final deterministic label per cell:

$$G_{i,j} = \begin{cases} \text{occupied}, & (i,j) \in \mathcal{O} \setminus \mathcal{B}, \\ \text{unknown}, & (i,j) \in \mathcal{B}, \\ \text{free}, & (i,j) \in \mathcal{F} \setminus \mathcal{B}, \\ \text{unknown (default)}, & \text{otherwise}, \end{cases}$$

   where $\mathcal{O}$ is the retained occupied set, $\mathcal{B}$ is the occluded set behind $\mathcal{O}$, $\mathcal{F}$ is the traced free set.
7. Optionally apply post-raster temporal smoothing $P_t(c) = \lambda P_{t-1}(c) + (1-\lambda)\hat{p}_t(c)$ with $\lambda = 0.65$, then a median filter, then morphological opening/closing with kernels derived from metric radii. The current deployment runs raw (filter pipeline disabled); the filtered branch is available and tested.
8. Publish as a ROS `OccupancyGrid` with origin $(z_{\min}, -x_{\max})$ in the camera depth frame.

Information-flow alignment with Layer 1. The output occupancy labels (free, occupied, unknown) map onto the three states Layer 1 reasons about: free → contributes positive clearance to Γ^W_{obs}; occupied → contributes $\Gamma^W_{\text{obs}} = 0$; unknown → Layer 1 falls back on memory or to the conservative-minimum default. The explicit unknown label keeps the front-end consistent with Assumption 1: $\partial_t \Gamma_{\text{curr}}$ is computed only on the persistently-known overlap, and the occupancy-grid's unknown cells stay outside that overlap until two consecutive observations agree.

D.4 Sim-to-real gap, status, and follow-ups

The RealSense occupancy-grid front-end has been validated in isolation (camera calibration, intrinsics integration, real-time grid generation). The remaining gap to full autonomous hardware navigation is the omnidirectional base's odometry calibration. The natural follow-ups, in priority order, are: (i) close the base odometry calibration; (ii) repeat the Gazebo scenarios on hardware and quantify the sim-to-real gap in tracking utilization, temporal deviation, and minimum clearance; (iii) re-measure per-layer compute on the

Jetson Orin Nano Super target. None of these is the subject of the present paper, which is the algorithm and its simulation evaluation; we list them so reviewers can see the deployment runway.

E Full Tables with Standard Deviations

Table 5 reproduces the body Q2 numbers with per-cell standard deviations and the full three-category fallback breakdown. Table 6 reproduces the temporal-deviation and clearance columns with p95 statistics. Both are paired across the same seeds 0–9 per scenario.

Table 5: **Q2 (full): tracking utilization and per-category fallback rate** (mean ± σ, 10 seeds).

Scenario	Tracking util. (%)	Task reset (%)	Sensing failure (%)	Regular continuation failure (%)
Corridor crossing	56.6 ± 10.9	0.4 ± 0.2	0.0 ± 0.0	43.0 ± 11.0
Dynamic hallway	72.2 ± 15.3	0.8 ± 0.3	0.0 ± 0.0	27.0 ± 15.6
Occluded boundary	98.7 ± 0.0	0.3 ± 0.0	0.0 ± 0.0	1.0 ± 0.0
FOV limited	46.7 ± 0.0	0.3 ± 0.0	0.0 ± 0.0	53.0 ± 0.0
Long corridor (dyn.)	84.7 ± 15.1	0.4 ± 0.1	0.0 ± 0.0	14.9 ± 15.2
Cluttered (5 occl.)	73.0 ± 7.5	0.2 ± 0.0	0.0 ± 0.0	26.8 ± 7.5

Table 6: **Q3 (full): temporal deviation (mean, p95) and minimum clearance** (mean ± σ, 10 seeds).

Scenario	Temporal dev. mean	Temporal dev. p95	Min. clearance (m)
Corridor crossing	0.0455 ± 0.00879	0.209 ± 0.0347	0.270 ± 0.0600
Dynamic hallway	0.0546 ± 0.0167	0.178 ± 0.0294	0.279 ± 0.0669
Occluded boundary	0.0129 ± 0.00	0.0872 ± 0.00	0.200 ± 0.00
FOV limited	0.0936 ± 0.00	0.282 ± 0.00	0.0500 ± 0.00
Long corridor (dyn.)	0.0388 ± 0.0101	0.164 ± 0.0234	0.271 ± 0.0476
Cluttered (5 occl.)	0.0147 ± 0.00446	0.0674 ± 0.0185	0.250 ± 0.00

Q4 with per-cell standard deviations. Table 7 reproduces the body Q4 numbers with per-cell standard deviations on every metric.

Table 7: **Q4 (full): six-scenario paired comparison** (10 seeds, mean ± σ).

	Collision (%)		Min. clearance (m)		Jerk RMS	
Scenario	point-DS	ours	point-DS	ours	point-DS	ours
Corridor crossing	0.0 ± 0.0	0.0 ± 0.0	**0.450 ± 0.00**	0.270 ± 0.0600	**0.0292 ± 0.00**	0.538 ± 0.133
Dynamic hallway	50.0 ± 50.0	**0.0 ± 0.0**	0.150 ± 0.201	**0.279 ± 0.0669**	4.43 ± 0.350	**1.34 ± 0.584**
Occluded boundary	0.0 ± 0.0	0.0 ± 0.0	**0.350 ± 0.00**	0.200 ± 0.00	5.31 ± 0.00	**0.889 ± 0.00**
FOV limited	100.0 ± 0.0	100.0 ± 0.0	0.0500 ± 0.00	0.0500 ± 0.00	**0.0511 ± 0.00**	0.948 ± 0.00
Long corridor (dyn.)	10.0 ± 30.0	**0.0 ± 0.0**	**0.295 ± 0.114**	0.271 ± 0.0476	4.79 ± 0.0889	**1.38 ± 0.665**
Cluttered (5 occl.)	0.0 ± 0.0	0.0 ± 0.0	**0.295 ± 0.0150**	0.250 ± 0.00	5.19 ± 0.00687	**2.49 ± 0.542**

Note. Corridor crossing, dynamic hallway, and occluded boundary reuse the E14 (with-SLAM-memory) rollouts; FOV limited reuses E08; long corridor and cluttered are E15. Bold = winner per cell.

F Per-Layer Timing

Measured on Intel i9-14900KF, Python 3.13, single thread, 3000 steps.

Table 8: **Per-layer mean and p99 latencies (Q5).** Manifold solver and tracker dominate; everything else (including the multi-boundary CBF QP) finishes in under 0.3 ms. The full-extract-frame total is ≈ 8.8 ms.

Layer	Mean (ms)	p99 (ms)
camera to current field	0.231	0.385
prediction	0.245	0.333
manifold solver	5.39	10.5
manifold tracker	4.45	7.25
ds controller	0.0595	0.0786
safety filter	0.235	0.259

G Supplementary Method Figures

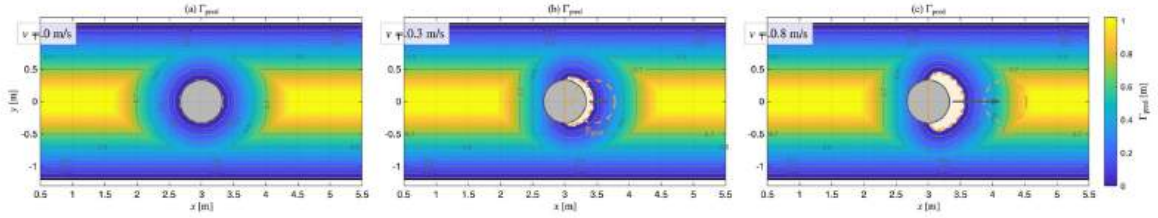


Figure 6: **Arrival-conditioned predicted clearance field Γ_{pred}.** Static obstacle ($v = 0$, left); the same obstacle drifting at 0.3 m/s (middle); at 0.8 m/s (right). The dashed orange contour shows where the obstacle will be when the robot arrives. As obstacle velocity grows, the predicted obstacle footprint shifts forward; the high-clearance corridor (yellow) opens behind it. The controller plans through Γ_{pred}, not through Γ_{curr}, so it does not retreat from a transient compression that resolves before arrival.

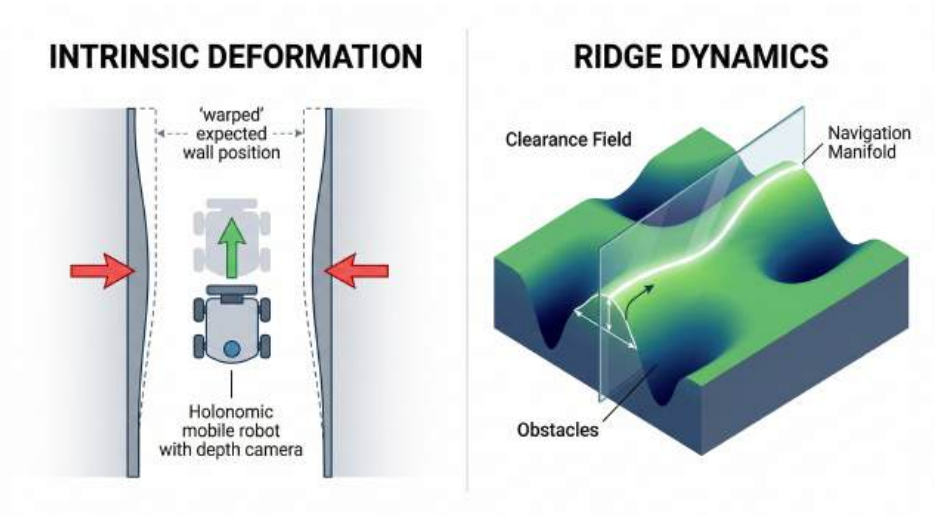


Figure 7: **Intrinsic deformation and ridge dynamics.** Left: walls compress around the robot (red arrows), pushing the warped free-space center inward; the relevant deformation is intrinsic to the corridor geometry. Right: lifting Γ to a height function makes the navigation manifold a ridge curve along the field, obstacles become valleys, and the manifold flows along the peak.