

Dharma: A Robot-Agnostic, Safety-Layered ROS 2 Autonomy Stack for Service Robots in Crowded Public Spaces

I Wayan Agus Darmawan^{1,2}

¹Department of Computer Engineering, Institut Teknologi Sepuluh Nopember (ITS), Surabaya, Indonesia

²PT Ezra Robotics Teknologi, Indonesia
darmawandevloper@gmail.com

Abstract—Service robots that guide, inspect or deliver in malls, stations and offices drive among people on legs or wheels, and the software that moves them is usually written for one robot. We present Dharma, an open ROS 2 autonomy stack (Foxy, Humble and Jazzy) that separates what is generic from what is robot-specific: one robot profile (footprint, base height, kinematics, limits, frames) is the single source of every geometric and dynamic assumption, typed and validated configuration replaces untyped parameters, and algorithm families are plugins. The stack combines LiDAR-inertial odometry behind a gravity-aligned bridge, a fiducial-anchored SE(2) extended Kalman filter over the map-to-odometry correction with chi-square gating, log-odds mapping with jump and correction gating, footprint-aware A* planning, path tracking with a Frenet path shift around people, and a protective field that limits translation, rotation and their combination with a single braking model shared by every speed limit in the stack. Sensing is fail-safe by construction: a blind or stale sensor is an obstacle, never free space. We describe the models, the software engineering that keeps them consistent (pure cores, module-size and purity checks, a Foxy, Humble and Jazzy CI matrix), and a seeded simulation protocol for stopping distance, blind-sensor behaviour, localization error and avoidance performance. In seeded simulations the field kept its margin, a LiDAR without returns zeroed the command within 0.13 s, the filter cut the localization error from 0.41 to 0.08 m, and path shift, after fixes of defects the runs exposed, passed people in all 40 runs.

Index Terms—service robots, legged robots, ROS 2, localization, speed and separation monitoring, software architecture

I. INTRODUCTION

Robots that guide visitors, patrol or carry goods in public buildings share their space with crowds that do not watch out for them [1]–[3]. Quadruped platforms are attractive there: they climb thresholds and ramps and are perceived as approachable, but their bodies pitch and roll with the gait, their footprint is long compared with its width, and their drivers expose only a velocity command. Products built on such platforms tend to grow their autonomy around one robot: the footprint is repeated in five parameter files, the base height is assumed in two different values, frames are hard-coded, and safety checks that were right for one speed are silently wrong for the next.

Dharma grew out of such a product (a tour-guide robot) and was refactored into a general, robot-agnostic stack. An independent review of the original code modelled every sub-

system and found three structural weaknesses: pose estimates carried no uncertainty and were never corrected continuously against the map; geometric assumptions (robot radius, base height, level frames) were inconsistent across packages; and the robot description was scattered, so the stack was not robot-agnostic in practice. It also found safety defects: a planner radius smaller than the robot’s half diagonal, a protective field that ignored rotation, and blind sensors read as free space. This paper describes the redesigned stack (v0.2.0) and the models behind it.

Contributions.

- A layered, robot-agnostic architecture in which one robot profile is the single source of geometry, limits and frames, with typed configuration validated at start and on every change, typed status interfaces and plugin registries (Sec. III).
- A gravity-aligned bridge for LiDAR-inertial odometry and a fiducial-anchored SE(2) extended Kalman filter over the map-to-odometry correction: drift grows with motion, not time, the innovation is taken where the robot is, and a chi-square gate opens with the uncertainty (Sec. IV-B).
- One braking model shared by the speed profile, the path shift and a protective field that limits translation, rotation and combined motion, with fail-safe sensing semantics and a latched e-stop that must be re-armed (Secs. IV-E–IV-F).
- The engineering that keeps the models consistent: pure algorithm cores, enforced module size and purity, a ROS 2 Foxy, Humble and Jazzy CI matrix, and traceability of every review finding to a fix and a regression test (Sec. V).
- A seeded simulation protocol with odometry drift, latency and sensor dropout to evaluate stopping, blind-sensor behaviour, localization and avoidance (Sec. VI).

II. RELATED WORK

Navigation frameworks. ROS 2 [4] and its navigation system Nav2 [5], [6] provide behaviour-tree navigation, costmap layers, planners and controllers such as regulated pure pursuit [7]. Dharma is smaller in scope and complementary: it targets a robot class (legged and omnidirectional service robots

driven through a velocity interface) and puts the emphasis on a single robot description, uncertainty-aware localization and a separate safety layer whose limits are derived from the same braking model as the planner and the follower. The transform library `tf` [8] carries every frame; we add per-edge age diagnosis on top.

LiDAR-inertial odometry. FAST-LIO2 [9] is an iterated error-state Kalman filter with direct point-to-plane updates; LIO-SAM [10] uses smoothing and mapping; Leg-KILO [11] adds leg kinematics for dynamic legged robots. These estimators take their world frame from the first IMU samples, so a robot that starts tilted produces a tilted world; Dharma levels it explicitly in a static bridge.

Fiducial localization and scan matching. AprilTag [12], [13] gives the pose of a known marker; the EKF formulation and chi-square gating follow [14], [15]. Correlative scan matching on a likelihood field [16] supplies continuous corrections against the map. Occupancy grids [17] with log-odds updates are the map representation; graph-based systems such as Cartographer [18] and SLAM Toolbox [19] close loops, which Dharma leaves for future work.

Planning, tracking and reactive avoidance. A* [20] on an inflated costmap, pure pursuit [21], the Stanley controller [22] and MPPI [23] are the path-level building blocks; vector field histograms [24], potential fields [25], follow-the-gap [26] and the dynamic window approach [27] are classical reactive methods. Lateral offsets in a Frenet frame [28] let a robot pass an obstacle while keeping its route.

Safety. Speed and separation monitoring limits speed so the robot can stop before a person reaches it [29], [30]. Certified functions follow ISO 13849-1 [31] or IEC 61508 [32], and personal care robots ISO 13482 [33]. Dharma’s protective field is a software channel modelled on speed and separation monitoring; it is explicitly the second channel next to a hardware e-stop and a safety-rated scanner, not a certified function.

III. ARCHITECTURE

Layers. Fig. 1 shows the stack. The robot adapter belongs to the product (driver, URDF, sensor mounts and the robot profile). Estimation and local perception turn raw LiDAR, IMU and camera data into odometry and obstacle sectors around the robot; the world model holds the map, the sites and the correction from odometry to the map; navigation plans and follows paths. Safety is a separate package beside all layers: one control multiplexer is the only publisher of the robot’s velocity command, and a health monitor turns degraded sensing into a requested speed scale. Each repository depends only on the two base packages, and the product bringup composes the package launch files.

One robot profile. Every node reads the same `robot.*` parameters, parsed into an immutable `RobotProfile`: kinematics (omnidirectional or differential), base height above the floor, a rectangular footprint (front, rear, half width), motion limits (speeds, acceleration, braking deceleration, yaw deceleration, reaction time) and frame names. Derived quantities are

computed, not copied: the inscribed radius $r_{in} = \min(f, b, w)$, the circumscribed radius $r_c = \sqrt{\max(f, b)^2 + w^2}$, the self filter of perception and mapping, the height band above the floor, the lateral speed limit ($v_y = 0$ for differential robots). For the default profile ($f = b = 0.40$ m, $w = 0.28$ m) $r_{in} = 0.28$ m and $r_c = 0.49$ m; the original planner used 0.35 m, so a path it accepted could swing a corner of the body about 14 cm into a wall while turning. The profile validates cross-field consistency (no lateral speed for a differential robot, distinct frames).

Typed configuration. Each node declares its parameters as frozen dataclasses whose fields carry a default, unit, range, choices, description and a *live* flag. The node helper turns them into ROS parameter descriptors, validates every value at start and on every `ros2 param set` (rejecting it with a reason), reports unknown parameters (typos) and makes safety values read-only at runtime. A node that receives an invalid safety parameter refuses to start instead of running with a default.

Plugins and typed interfaces. Controllers and avoidance policies implement abstract base classes and register in a named registry that is also open to Python entry points, so a product adds an implementation without forking. All status topics are typed messages with enum constants (odometry, localization, mapping, site, planner, navigation, control, health) instead of strings; safety-relevant actions are separate trigger services so that a default request can never release or stop something.

Frames. The tree follows REP-105 (Fig. 2): `map`→`odom`→`camera_init`→`body`→`base_link`, where `camera_init` is the IMU frame at estimator start and `body` the IMU now. The pose of the robot in the map is

$${}^{map}\mathbf{T}_{base}(t) = {}^{map}\mathbf{T}_{odom} L {}^{base}\mathbf{T}_{imu} {}^{ci}\mathbf{T}_{body}(t) {}^{imu}\mathbf{T}_{base}, \quad (1)$$

with the static mounting ${}^{base}\mathbf{T}_{imu}$ from the URDF, ${}^{imu}\mathbf{T}_{base}$ its inverse and the level rotation L of Sec. IV-A. Frame names come from the profile; nothing is hard-coded.

IV. METHODS

A. Gravity-aligned odometry bridge

FAST-LIO2 [9] (or Leg-KILO [11]) estimates ${}^{ci}\mathbf{T}_{body}(t) \in \text{SE}(3)$ with the world frame `camera_init` equal to the IMU frame at start. If the robot starts tilted by θ , that world is tilted: a horizontal distance s appears as a height change and a shorter planar distance,

$$\Delta z \approx s \sin \theta, \quad \frac{\Delta s_{xy}}{s} = 1 - \cos \theta, \quad (2)$$

i.e. 3.5, 8.7 and 13.9 cm of false height per metre for $\theta = 2^\circ, 5^\circ, 8^\circ$. Mapping and perception filter obstacles in a height band above the floor and assume a level frame, so a start at the previous 8° limit puts the floor 5 m ahead 0.7 m above the band. Two measures remove this. A launch prefix holds the estimator (and every respawn) until the IMU is still and level: largest angular rate below 0.03 rad/s, relative standard deviation of $\|\mathbf{f}\|$ below 1 %, tilt at most 3° . Then localization measures the mean specific force $\bar{\mathbf{f}}$ over a still

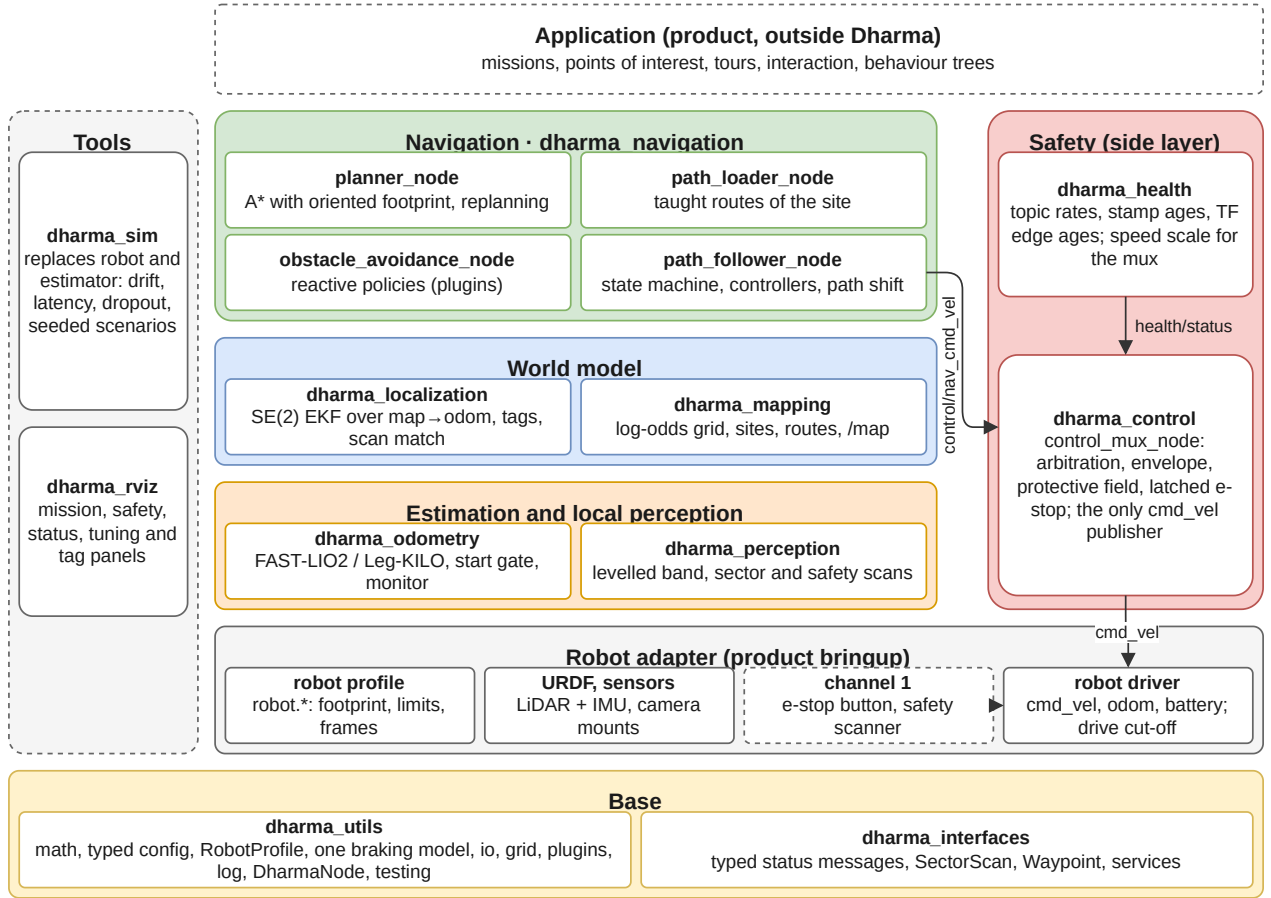


Fig. 1. Layers of Dharma. Every layer depends only on the layers below it and on the base (`dharma_utils`, `dharma_interfaces`); the safety layer sits beside them and is the last gate before the robot driver. Colours: estimation orange, world model blue, navigation green, safety red, base yellow, hardware grey.

window and computes the up direction of the base at start, $\mathbf{u} = \mathbf{R}_{imu}^{base} \mathbf{R}_{body}^{ci}(t) \bar{\mathbf{f}} / \|\cdot\|$. The level rotation L is the rotation that aligns $\mathbf{u}$ with $\mathbf{z}$, with its yaw removed so that `odom`'s x axis keeps the start heading. The bridge publishes

$$odom \mathbf{T}_{ci} = L^{base} \mathbf{T}_{imu}, \quad body \mathbf{T}_{base} = (base \mathbf{T}_{imu})^{-1}, \quad (3)$$

so $odom \mathbf{T}_{base}(t_0) = L$: `odom` starts at `base_link`, with z against gravity. After an estimator restart it is levelled again. Only the direction of $\bar{\mathbf{f}}$ is used, so accelerometer units do not matter.

The odometry monitor compares the estimator speed with the robot's own odometry as a residual test [34]:

$$r_k = v_k^{lio} - v_k^{leg}, \quad \chi_k^2 = \frac{r_k^2}{\sigma_{lio}^2 + \sigma_{leg}^2}, \quad (4)$$

$$S_k = \max(0, S_{k-1} + \chi_k^2 - \kappa),$$

with an alarm while $S_k > h$ ($\kappa = 3.84$, the 95 % quantile of χ_1^2 , $h = 20$). The thresholds follow from the sensor variances, not from robot speeds. A restart is detected only when the pose returns to the origin after a message gap or a physically impossible translation jump; a yaw change alone (turning in place near the start) never counts.

B. Fiducial-anchored SE(2) EKF

The original stack replaced `map`→`odom` by every tag detection, without uncertainty and with ad-hoc rejection thresholds (0.5 m, 15°) that compared the correction itself. A heading error $\delta\psi$ of that correction displaces the robot by $e_\perp \approx L \delta\psi$ after a distance L from the tag (17.5 cm after 10 m at $\delta\psi = 1^\circ$), and a small rotation of `map`→`odom` far from the odometry origin moves the robot metres. Dharma estimates the correction itself (Fig. 3).

State and prediction. The state is $\mathbf{x} = (x, y, \psi)$ of $map \mathbf{T}_{odom}$ with a 3×3 covariance $\mathbf{P}$. With the odometry pose $\mathbf{o} = odom \mathbf{T}_{base}$, the robot pose is $h(\mathbf{x}) = \mathbf{x} \oplus \mathbf{o}$ and its Jacobian with respect to the state

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & -(y_b - y) \\ 0 & 1 & x_b - x \\ 0 & 0 & 1 \end{bmatrix} \quad (5)$$

contains the lever arm from the `odom` origin to the robot (x_b, y_b) . Drift is proportional to motion: the odometry is counted in steps of distance Δs and rotation $\Delta\theta$, and

$$\mathbf{P} \leftarrow \mathbf{P} + \mathbf{H}^{-1} \mathbf{Q} \mathbf{H}^{-\top}, \quad \mathbf{Q} = \text{diag}(\sigma_s^2 \Delta s, \sigma_s^2 \Delta s, \sigma_{\psi s}^2 \Delta s + \sigma_{\psi \theta}^2 \Delta \theta), \quad (6)$$

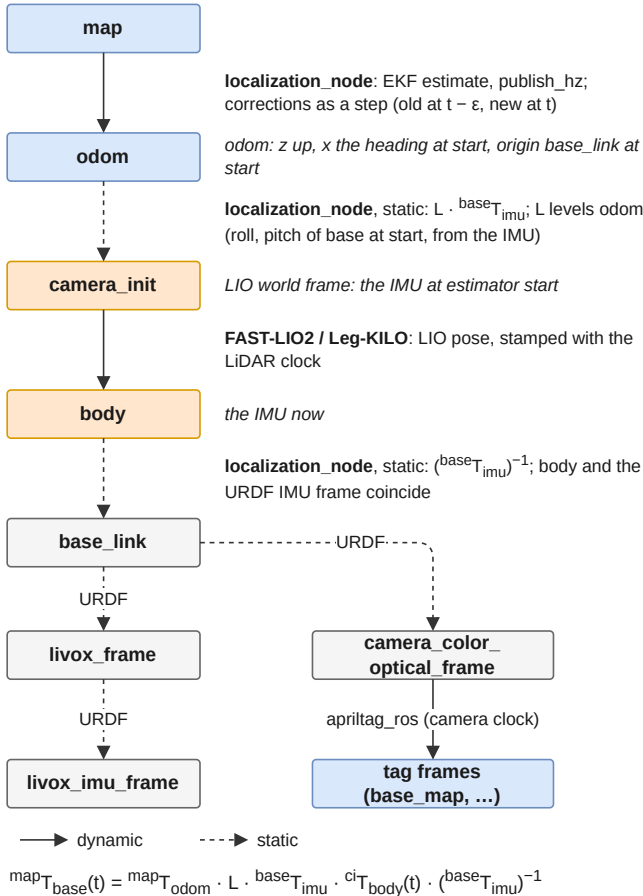


Fig. 2. TF tree. The LiDAR-inertial odometry publishes $\text{camera_init} \rightarrow \text{body}$; localization adds the static bridge (with the level rotation L) and the map correction.

so a robot standing still never becomes uncertain (defaults $\sigma_s = 0.03 \text{ m}/\sqrt{\text{m}}$, $\sigma_{\psi s} = 0.1^\circ/\sqrt{\text{m}}$, $\sigma_{\psi\theta} = 0.5^\circ/\sqrt{\text{rad}}$).

Update and gate. A source measures the robot pose z of $\text{map} \rightarrow \text{base_link}$ with covariance R . The innovation is taken on the robot pose,

$$\nu = z - h(x), \quad S = HPH^T + R, \quad (7)$$

$$d^2 = \nu^T S^{-1} \nu \leq \chi_{3,0.99}^2 = 11.34,$$

(yaw wrapped), so a fix is judged by how far it moves the robot. The gate grows with P after a long drive and is tight right after a correction. Accepted measurements update the state with $K = PH^T S^{-1}$ and the Joseph-form covariance $P \leftarrow (I - KH)P(I - KH)^T + KRK^T$, with a floor on the robot pose covariance against overconfidence. The state is NONE before the first fix and UNCERTAIN when σ_{xy} or σ_{ψ} exceed their limits.

Sources. A tag with known map pose ${}^{\text{map}}T_{\text{tag}}$ seen at ${}^{\text{cam}}T_{\text{tag}}$ gives

$${}^{\text{map}}T_{\text{base}} = {}^{\text{map}}T_{\text{tag}} ({}^{\text{base}}T_{\text{cam}} {}^{\text{cam}}T_{\text{tag}})^{-1}, \quad (8)$$

computed in SE(3) (the optical frame is never planar) and flattened to SE(2). Its noise grows with distance d and view angle

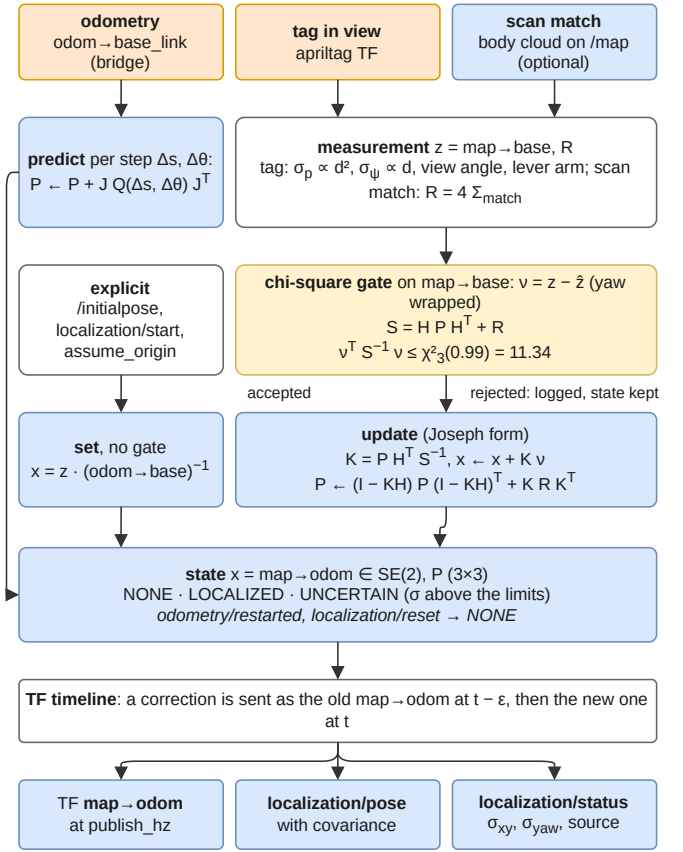


Fig. 3. Localization: an EKF over $\text{map} \rightarrow \text{odom}$, predicted by odometry motion and corrected by fiducials, scan matching and explicit operator input.

$\alpha, \sigma_p = \sigma_1 d^2 / \cos \alpha, \sigma_{\psi} = \sigma_{\psi 1} d / \cos \alpha$, and a heading error moves the robot around the tag: with l the robot-to-tag vector rotated by 90° , $R_{xy} = \sigma_p^2 I + \sigma_{\psi}^2 l l^T$ and $R_{xy, \psi} = \sigma_{\psi}^2 l$, so far tags count little. Optional scan matching projects the registered cloud into the level frame, keeps a height band and matches it on the map with a correlative matcher [16], using the scaled covariance of the score surface. Operator pose estimates, an explicit localization/start and assume_origin set the state without gate, so a far but consistent tag after a kidnapping still relocates on request.

Publication. tf2 interpolates between samples, so a correction between two periodic samples would map scans to poses that never existed. A correction is therefore published as the previous $\text{map} \rightarrow \text{odom}$ at $t - \epsilon$ ($\epsilon = 1 \text{ ms}$) and the new value at t . The robot pose is also published with its covariance HPH^T .

C. Occupancy mapping with jump and correction gating

Each cell holds log-odds $l = \log \frac{p}{1-p}$ updated by

$$l_t = \text{clip}(l_{t-1} + \{l_{\text{hit}} = 0.85, l_{\text{miss}} = -0.4\}, [-2, 3.5]), \quad (9)$$

occupied above $\text{logit}(0.65) = 0.619$; free space is carved only up to the nearest hit per angular bin, so glass and open doors are not cleared. A saturated cell needs $\lceil (3.5 + 0.619)/0.4 \rceil = 11$ crossing rays to become free. Points are kept between a minimum and maximum height above the floor (the base

height below `base_link`), measured in the levelled map frame. Since scans are integrated at their TF pose, the map is only as good as the pose; two gates keep bad poses out. A scan whose sensor moved more than $d_{\max} + v_{\max}\Delta t$ (or turned more than $\psi_{\max} + \omega_{\max}\Delta t$) since the previous scan holds integration until a scan agrees with the map again, and a scan during which `map`→`odom` changed (a correction in flight) is skipped. The grid side is capped. An optional refinement matches each scan on the current grid before integrating it; loop closure is future work.

D. Footprint-aware planning

The planning costmap is built from the map with radii from the profile: a cell closer than $r_{\text{lethal}} = r_{\text{in}} + m$ to an obstacle is lethal for every heading; up to $r_c + m$ a collision depends on the heading and the planner checks the *oriented* footprint at the heading the follower will use; beyond, the cost decays to 1 at the inflation radius R ,

$$c(d) = 1 + W \frac{e^{-k(d-r_\ell)} - e^{-k(R-r_\ell)}}{1 - e^{-k(R-r_\ell)}}, \quad r_\ell < d < R, \quad (10)$$

with $r_\ell = r_{\text{lethal}}$ and $W = 5$. A* [20] runs 8-connected without corner cutting and with the octile heuristic (admissible since $c \geq 1$). Line-of-sight shortcuts are accepted only when they stay as far from walls as the search (a line touching a cell corner counts both side cells, so it never slips through a diagonal pinch); smoothing is discarded when it touches a lethal cell. Where the footprint check fails along the result, the offending cells are blocked and the search repeats.

E. Path tracking, speed profile and path shift

All speed limits use one braking model: the distance to stop from speed v with reaction time t_r and deceleration a , and its inverse, the highest speed that stops within a room d ,

$$D(v) = v t_r + \frac{v^2}{2a}, \quad v^*(d) = a \left(-t_r + \sqrt{t_r^2 + 2d/a} \right). \quad (11)$$

The speed profile along the route takes the curvature limit $v_i \leq \sqrt{a_{\text{lat}}/|\kappa_i|}$, the approach speed at stop points and the end, a backward pass $v_i \leq \sqrt{v_{i+1}^2 + 2a_{\text{dec}}\Delta s_i}$, the braking model (11) before every stop and a forward pass $v_{i+1} \leq \sqrt{v_i^2 + 2a_{\text{acc}}\Delta s_i}$. The default holonomic controller steers toward a carrot a distance L ahead on the route, so the cross-track error decays as $e(t) = e_0 e^{-tv/L}$ (95% after $3L$ of travel, independent of speed) and the heading is controlled separately toward the route direction; five other controllers (pure pursuit [21], PID, LQR, Stanley [22], MPPI [23]) are plugins. Reactive avoidance outputs a heading offset (rad, a rotation of the direction of travel) and a yaw-rate bias (rad/s), applied the same way by every controller; the original single *steering bias* was a rate in the message and an angle in one controller. VFH blocks headings within $\arcsin((w+c)/d)$ of an obstacle at range d .

Path shift. For omnidirectional robots, obstacles of the 360° safety scan are expressed in route coordinates (s, d) [28]. A lateral offset o is clear when no point lies in the corridor $|d -$

$o| < w + c$ (w the half width, c a clearance) from the rear edge of the body to a look-ahead ℓ beyond its front edge. The offset is

$$o^* = \arg \min_{o \text{ clear}} |o| + \lambda |o - o_{\text{prev}}|, \quad (12)$$

with hysteresis and a delay before returning toward the route. Without a clear offset the robot keeps the one with the largest free distance. The committed offset follows o^* at most 0.25 m ahead of the robot. The speed limit is $v^*(d_{\text{free}} - m)$ with d_{free} the smallest free distance over the target, committed and measured offsets; while it holds the robot, the robot still slides where that is clear. Near a stop point it stays on the route.

F. Protective field and fail-safe sensing

The control multiplexer (Fig. 4) arbitrates navigation (AUTO) and one operator source (REMOTE) with watchdogs on the steady clock, clamps every command to the robot's envelope (non-finite values become zero) and limits it with a protective field on the 360° safety scan, a form of speed and separation monitoring [29], [30].

Translation. For the direction $\hat{u}$ of the commanded translation, the clearance d_t is the distance the footprint can move before touching a scan point (points ahead of the leading edge within the body corridor widened by a lateral margin, and zero for a point beside or inside the body). The speed is scaled to the highest v whose protective distance $D(v) + v_h(t_r + v/a)$ fits $d_t - m$, with the ISO/TS 15066 separation term of a person approaching at $v_h = 0.5$ m/s; the follower's guard does the same.

Rotation. Turning in place, the corners sweep the ring $r_{\text{in}} < r \leq r_c$ (0.28 to 0.49 m for the default profile), which the original field ignored. The free angle θ_f is found by rotating the ring points in 1° steps until one enters the footprint, and $|\omega| \leq \omega^*(\theta_f - m_\psi)$ with the same model and the yaw deceleration.

Combined motion. For a twist (v_x, v_y, ω) the robot keeps the command for t_r and then brakes both axes proportionally over $T_b = \max(v/a, |\omega|/\alpha)$; it covers the constant-twist arc up to the parameter $u = t_r + T_b/2$ (exactly $D(v)$ on a straight line). The footprint is checked at poses along the arc; if the first contact is at u_c , scaling the command by s keeps the curvature and covers $st_r + s^2 T_b/2$, so the largest clear scale is

$$s = \frac{-t_r + \sqrt{t_r^2 + 2T_b u_c}}{T_b}. \quad (13)$$

With $a = 0.6$ m/s², $t_r = 0.25$ s and $m = 0.15$ m, (11) allows 0.30, 0.52, 0.87 and 1.35 m/s at 0.3, 0.5, 1.0 and 2.0 m of free distance.

Fail-safe semantics. A sector with no data is not free space. The perception node reports every sector as BLOCKED when the LiDAR is blind (too few points, or an empty quarter of the view), when the sensor or levelling transform is missing, when processing fails or when no usable cloud arrives; blocked and NaN sectors are obstacles at the robot for every consumer. The mux accepts a scan only in the base frame with full-circle coverage and fresh receive and stamp ages; otherwise it

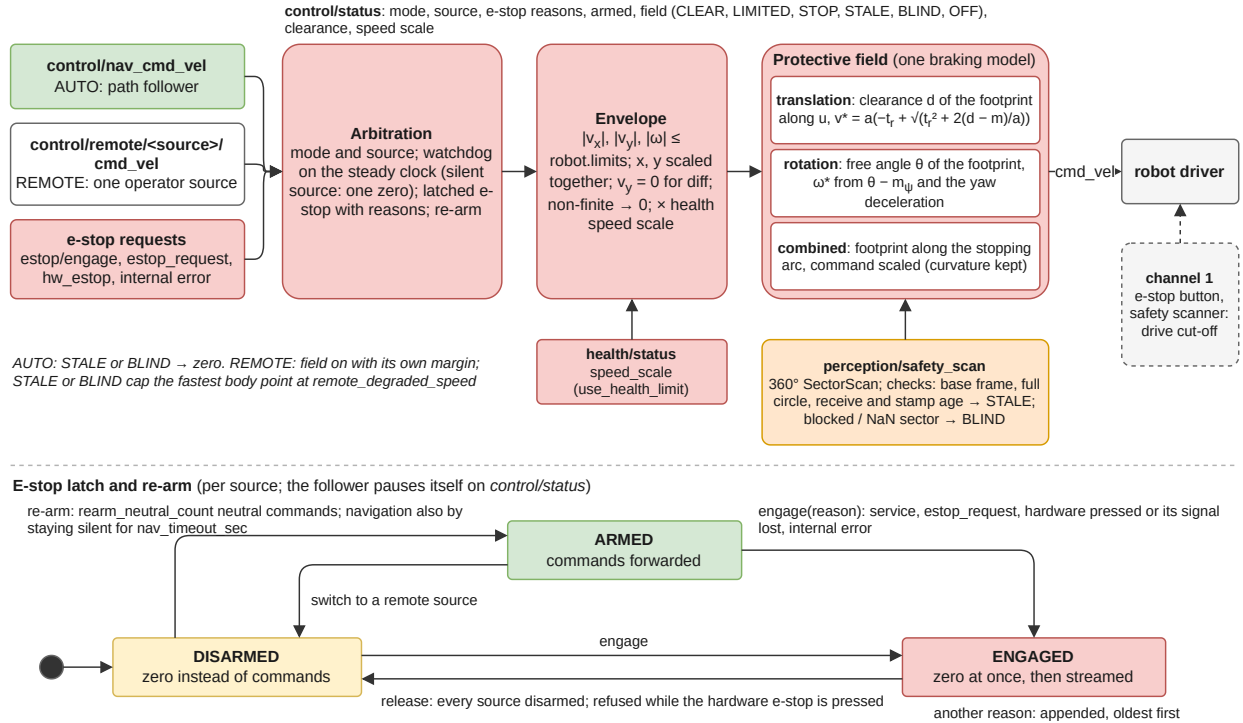


Fig. 4. Safety layer: arbitration, envelope and protective field in front of the robot's `cmd_vel`, and the e-stop latch with re-arm. Channel 1 (hardware) cuts the drive independently of any software.

is STALE. In AUTO a stale or blind scan stops the robot; in REMOTE the fastest point of the body is capped at a creep speed. Perception levels the points with roll and pitch of the base before the height band, since in `base_link` a floor point at range r enters the band when $r > (h + z_{\min}) / \sin \vartheta$: 2.9 m at a gait pitch of $\vartheta = 2^\circ$ with a 10 cm margin. A k -th nearest point per sector ($k = 2$) ignores single dust returns, and a short obstacle memory in `odom` keeps objects that drop below the lowest beam near the robot.

E-stop. The e-stop latches a list of reasons (service, topic, hardware input, lost hardware signal, internal error); zero is published at once and streamed while engaged. Release is refused while the hardware e-stop is pressed and disarms every source: a source drives again only after a number of consecutive neutral commands, so a stick held during a release never moves the robot. An exception in any callback engages the e-stop.

G. Health and the TF latency budget

The follower stops when the robot pose is older than a limit. `tf2` answers a lookup at the latest common time of the chain, so with E the dynamic edges and τ_e their stamps,

$$\begin{aligned} \text{age}(t) &= t_{\text{now}} - \min_{e \in E} \tau_e \\ &\approx \max_e (L_e^{\text{acq}} + L_e^{\text{proc}} + L_e^{\text{tx}} + L_e^{\text{q}} - \Delta_e) + \Delta_c + T_p, \end{aligned} \quad (14)$$

with acquisition, processing, transport and queueing latencies, the clock offsets Δ_e of the edge sources and Δ_c of the consumer, and T_p the time since the last sample. In Dharma the

LIO edge dominates: a 100 ms scan, 20–80 ms of processing and transport give a normal age of about 0.15–0.35 s against a 0.5 s limit, and an unsynchronized LiDAR clock adds its offset directly. The follower therefore distinguishes `TF_STARTING`, `TF_STALE` and `TF_CLOCK` (a pose from the future) and, for ages between 0.1 s and the limit, predicts the pose forward with the odometry twist instead of stopping. The health monitor checks topic rates and header stamp ages read from the serialized messages (a 10 Hz topic with 2 s old stamps is a fault), classifies each TF edge's recent ages as a source clock offset (late but steady), load (varying) or a stopped publisher, and watches the typed statuses, the safety scan, the expected nodes and the computer. Its speed scale for the mux is 0 on an error and 0.5 by default while the LiDAR, the estimator or TF is late.

V. SOFTWARE ENGINEERING

Pure cores, thin nodes. Every package has the same layout: `core/` holds the algorithms, state machines and configuration dataclasses in plain Python and numpy, without ROS imports; `utils/` holds the conversions between messages and core types; the nodes only declare parameters, topics, services and timers. The path follower, for example, is a pure state machine `Follower.tick(inputs) → Decision(command, status, stop events)` with an explicit table of allowed state transitions; the node feeds it once per control tick and publishes the decision, so every path through it, including zero commands on stop, e-stop, stale inputs, pause and shutdown,

TABLE I
ENGINEERING METRICS BEFORE (v0.1) AND AFTER (v0.2.0)

Metric	v0.1	v0.2.0
Largest node (lines)	1429	277
Copies of the node main boilerplate	15	1
Copies of the latched QoS definition	11	1
Robot geometry declared in (files)	≈ 6	1
Test functions (without lint)	158	595
Findings with a cited test	–	90 of 97
CI distributions	2	2

is unit-tested without ROS. The original follower node had 1429 lines; the new node has 277.

Shared building blocks. `dharma_utils` provides what packages used to copy: angles, quaternions, SE(2) and SE(3) helpers, filters (EMA, slew limiter, CUSUM), the typed configuration, the robot profile, the braking model and swept-footprint clearances, atomic files with backups and path guards, grid distance fields and the correlative matcher, the plugin registry, structured logging, and `DharmaNode` (typed parameters with live validation, TF cache with per-edge ages, exception-guarded timers, shutdown hooks).

Checks. A style test in every package enforces the conventions: modules of at most 300 lines (tests 400), no ROS imports in `core/`, no shadowed node members. Lint and format run through the same pre-commit hooks locally and in CI (ruff, clang-format, prettier, shellcheck). The CI workflow, identical in every repository, builds and tests on ROS 2 Foxy (Python 3.8, numpy 1.17), Humble (Python 3.10, numpy 1.21) and Jazzy (Python 3.12, numpy 1.26); Dharma dependencies are imported from `deps.repos` and checked out at the same branch when it exists, so a change that spans repositories is tested together before it is merged.

Traceability. Two reviews of the original stack produced findings with priorities (P0 safety, P1 robustness, P2 maintainability). Each one maps to a change and a regression test; the technical manual lists the full table.

VI. EVALUATION

Protocol. E1–E4 close the loop over the pure cores at the node rates: the simulator core (acceleration limits, collisions, Mid-360 ray casting, odometry drift $\xi \sim \mathcal{N}(\mathbf{0}, \Sigma \Delta s)$, 50/20 ms sensor/command latency), perception, the follower with VFH and the mux, all with package defaults and ideal localization except in E3. Each condition runs seeds 1–20 (mean $\pm$ std); `experiments/` in the hub reproduces them.

E1: stopping distance. At speed v a person appears on the robot’s axis $D(v) + m$ ahead, the last point from which (11) promises a stop with the margin. With $v_h = 0$ the robot stopped 0.06, 0.20, 0.40 and 0.68 m after the first scan showing it, against $D(v) = 0.08, 0.23, 0.45$ and 0.73 m at 0.2–0.8 m/s, braking 0.02–0.18 s after the appearance (inside t_r); with $v_h = 0.5$ m/s, 0.04–0.56 m; never closer than 0.162 m.

E2: blind LiDAR. At 0.5 m/s the LiDAR delivers clouds without points or with 50 points for 4 s: v0.2 commanded zero within 0.13 s and stood within 0.28 m; the v0.1 baseline (a

TABLE II
E3: POSITION RMSE (ATE), LARGEST POSITION ERROR, HEADING RMSE

Method	ATE (m)	max (m)	yaw (°)
Hard replace (v0.1)	0.408 ± 0.202	1.06 ± 0.30	2.85 ± 1.24
EKF, tags	0.078 ± 0.009	0.22 ± 0.04	0.92 ± 0.23
EKF, tags + scan match	0.029 ± 0.002	0.08 ± 0.01	0.47 ± 0.08

TABLE III
E4: COMPLETED RUNS, THEIR TIME AND STOPS; SMALLEST CLEARANCE, LARGEST OFFSET

people	avoidance	done	time (s)	stops	clear. (m)	offset (m)
on	path shift	20/20	70.5 ± 4.7	1.3 ± 1.0	0.18	0.80
on	reactive	0/20	–	–	0.53	0.17
on	none	0/20	–	–	0.44	0.00
beside	path shift	20/20	70.9 ± 1.5	1.2 ± 0.6	0.23	0.80
beside	reactive	20/20	72.3 ± 1.6	2.4 ± 0.9	0.20	0.15
beside	none	13/20	72.1 ± 1.7	2.2 ± 0.4	0.12	0.00

blind scan read as free) drove on at full speed. A silent LiDAR stopped both at the 0.5 s timeout; in REMOTE v0.2 crept at 0.15 m/s, v0.1 kept 0.5 m/s.

E3: localization. The follower drives three laps (86 m) of the simulator’s mall route with its drift scenario past four tags, detected at 5 Hz with the modelled noise plus 5 % outliers (5–25°); the estimators share all inputs (Table II). Hard replacement rejected 62 % of the good detections, as its 0.5 m threshold on `map→odom` grows with the distance from the odometry origin; the gate rejected 79 % of the outliers and 3 % of the good detections.

E4: path shift. The follower drives 20 m along a 3 m corridor past a person standing on the route (*on*) or two beside it, one walking ahead and one crossing; nobody gives way (Table III). A first run exposed defects, now fixed: path shift deadlocked or lost its offset while the robot stood (17/20 on the route, now 20/20), a walking person brushed the stopped robot (now 0.28 m), and a held robot was reported FOLLOWING. A second round swept the footprint along the command on the mux’s points: beside the route reactive avoidance passed in 20 and no avoidance in 13 of 20 runs (8 and 7 before; the rest leave 0.07–0.12 m beside the body). On the route VFH still finds no gap wide enough and stops.

End to end. On the real nodes (first round) the robot drove the 27 m mall route past two standing people in 83 s (79 s before), at least 0.23 m from them and 0.39 m from walls.

VII. LIMITATIONS

Software safety is not certified. The protective field, watchdogs and e-stop run in Python over DDS. They are a second channel; a robot in a public space needs a first channel that is safety-rated (hardware e-stop and a certified safety scanner or bumper wired to the drive’s safe torque off), designed and validated to ISO 13849-1 [31] or IEC 61508 [32], and a safety case (hazards, mitigations, verification). Dharma reads the hardware e-stop state so both channels latch together, but it does not replace it.

Perception gaps. A Mid-360 sees down to -7° : mounted 0.53 m above the floor, a 10 cm object is seen only beyond about 3.5 m. The obstacle memory keeps objects seen before, but an object that appears inside this near-field zone is not detected, and missing floor (stairs down, platform edges) is not an obstacle at all. A pitched mount or a near-field sensor is required where this matters. The scan is not deskewed (up to $v \cdot 0.1$ m of distortion per frame), which the field margins must cover.

No loop closure. Mapping integrates scans at the corrected odometry pose; large sites accumulate error between fiducials. A pose graph over scan matches with tags as landmarks would unify mapping and localization.

Sim-to-real gap. The evaluation is in simulation with modelled drift, latency and dropout. Braking deceleration and reaction time must be measured on each robot (worst-case floor, payload and battery) before the field limits can be trusted; the results do not transfer to another robot without repeating them, and real-robot trials are future work.

VIII. CONCLUSION

Dharma shows that a small autonomy stack can be robot-agnostic and explicit about its assumptions: one robot profile feeds every limit, the localization estimate carries its uncertainty and gates its corrections, and one braking model bounds every speed, from the route profile to a protective field that treats missing data as an obstacle. Pure cores and enforced conventions keep these models consistent across eleven repositories and three ROS 2 distributions. In simulation the field kept its margin and the gated filter cut the localization error fivefold; the defects the runs exposed are fixed, but reactive avoidance still stops short of a person on the route. Future work is a pose graph, deskewed near-field perception and a second robot model on the same stack.

REFERENCES

- [1] W. Burgard, A. B. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun, "Experiences with an interactive museum tour-guide robot," *Artificial Intelligence*, vol. 114, no. 1-2, 1999.
- [2] R. Triebel, K. Arras, R. Alami, L. Beyer, S. Breuers, R. Chatila, M. Chetouani, D. Cremers *et al.*, "SPENCER: A socially aware service robot for passenger guidance and help in busy airports," in *Field and Service Robotics*, ser. Springer Tracts in Advanced Robotics, 2016, pp. 607–622.
- [3] T. Kruse, A. K. Pandey, R. Alami, and A. Kirsch, "Human-aware robot navigation: A survey," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1726–1743, 2013.
- [4] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot operating system 2: Design, architecture, and uses in the wild," *Science Robotics*, 2022.
- [5] S. Macenski, F. Martín, R. White, and J. Ginés Clavero, "The Marathon 2: A navigation system," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2020.
- [6] S. Macenski, T. Moore, D. V. Lu, A. Merzlyakov, and M. Ferguson, "From the desks of ROS maintainers: A survey of modern & capable mobile robotics algorithms in the Robot Operating System 2," *Robotics and Autonomous Systems*, 2023.
- [7] S. Macenski, S. Singh, F. Martín, and J. Ginés, "Regulated pure pursuit for robot path tracking," *Autonomous Robots*, 2023.
- [8] T. Foote, "tf: The transform library," in *Proc. IEEE Int. Conf. Technologies for Practical Robot Applications (TePRA)*, 2013, pp. 1–6.
- [9] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang, "FAST-LIO2: Fast direct LiDAR-inertial odometry," *IEEE Transactions on Robotics*, 2022.
- [10] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and D. Rus, "LIO-SAM: Tightly-coupled lidar inertial odometry via smoothing and mapping," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2020.
- [11] G. Ou, D. Li, and H. Li, "Leg-KILO: Robust kinematic-inertial-lidar odometry for dynamic legged robots," *IEEE Robotics and Automation Letters*, vol. 9, no. 10, pp. 8194–8201, 2024.
- [12] E. Olson, "AprilTag: A robust and flexible visual fiducial system," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2011.
- [13] J. Wang and E. Olson, "AprilTag 2: Efficient and robust fiducial detection," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2016.
- [14] Y. Bar-Shalom, X.-R. Li, and T. Kirubarajan, *Estimation with Applications to Tracking and Navigation*. New York: John Wiley & Sons, 2001.
- [15] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT Press, 2005.
- [16] E. B. Olson, "Real-time correlative scan matching," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2009, pp. 4387–4393.
- [17] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, vol. 22, no. 6, pp. 46–57, 1989.
- [18] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2D LIDAR SLAM," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2016, pp. 1271–1278.
- [19] S. Macenski and I. Jambrecic, "SLAM toolbox: SLAM for the dynamic world," *Journal of Open Source Software*, vol. 6, no. 61, p. 2783, 2021.
- [20] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.
- [21] R. C. Coulter, "Implementation of the pure pursuit path tracking algorithm," The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-92-01, Jan. 1992.
- [22] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron *et al.*, "Stanley: The robot that won the DARPA Grand Challenge," *Journal of Field Robotics*, vol. 23, no. 9, pp. 661–692, 2006.
- [23] G. Williams, N. Wagener, B. Goldfain, P. Drews, J. M. Reh, B. Boots, and E. A. Theodorou, "Information theoretic MPC for model-based reinforcement learning," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2017, pp. 1714–1721.
- [24] J. Borenstein and Y. Koren, "The vector field histogram – fast obstacle avoidance for mobile robots," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 3, pp. 278–288, 1991.
- [25] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *The International Journal of Robotics Research*, vol. 5, no. 1, pp. 90–98, 1986.
- [26] V. Sezer and M. Gokasan, "A novel obstacle avoidance algorithm: 'Follow the Gap Method'," *Robotics and Autonomous Systems*, vol. 60, no. 9, pp. 1123–1134, 2012.
- [27] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, 1997.
- [28] M. Werling, J. Ziegler, S. Kammel, and S. Thrun, "Optimal trajectory generation for dynamic street scenarios in a Frenet frame," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2010, pp. 987–993.
- [29] ISO, "ISO/TS 15066:2016, robots and robotic devices – collaborative robots," International Organization for Standardization, 2016.
- [30] J. A. Marvel and R. J. Norcross, "Implementing speed and separation monitoring in collaborative robot workcells," *Robotics and Computer-Integrated Manufacturing*, vol. 44, pp. 144–155, 2016.
- [31] ISO, "ISO 13849-1:2023, safety of machinery – safety-related parts of control systems – part 1: General principles for design," International Organization for Standardization, 2023.
- [32] IEC, "IEC 61508-1:2010, functional safety of electrical/electronic/programmable electronic safety-related systems – part 1: General requirements," International Electrotechnical Commission, 2010.
- [33] ISO, "ISO 13482:2014, robots and robotic devices – safety requirements for personal care robots," International Organization for Standardization, 2014.
- [34] E. S. Page, "Continuous inspection schemes," *Biometrika*, vol. 41, no. 1-2, pp. 100–115, 1954.