Practical Navigation of a Mecanum-Wheel Robot on High-Friction Carpet: Motor Deadzone Compensation, Rotation Asymmetry, and Multi-Phase Goal Management

Aung Khant Ko
Thomas Edison State University
Department of Computer Science
github.com/akkexd/ros2-mecanum-robot

May 2026

Abstract

Mecanum-wheel robots offer omnidirectional mobility for indoor navigation, but their performance degrades severely on high-friction carpeted surfaces. This paper presents three software-based solutions to carpet-induced navigation failures, developed and validated on a physical robot platform (Yahboom ROSMASTER X3, ROS 2 Jazzy, RPLIDAR A1, Intel RealSense D435). First, a proportional deadzone compensation algorithm uniformly scales motor commands to preserve the velocity direction vector while overcoming carpet static friction thresholds. Empirical characterization reveals a binary deadzone at PWM 30–35 on loop-pile carpet versus PWM 15–20 on hard floor—a 75% wider deadzone that standard navigation stacks cannot detect. Second, rotation asymmetry testing across two wheel configurations (80 mm plastic and 100 mm TPU rubber) demonstrates that carpet fiber grain creates direction-dependent friction that varies with roller material: 80 mm plastic wheels exhibit consistent counterclockwise-only success (mean 180° vs. 22° CW in 10 s), while 100 mm TPU wheels show reduced but unpredictable asymmetry. Third, a multi-phase goal management architecture (Goal Manager V4) decomposes navigation into separate rotation, translation, and doorway transit phases with IMU-based stall detection and kickstart recovery, maintaining full compatibility with the standard ROS 2 Nav2 stack. Experimental testing across five navigation trials achieves 100% success on a 14 m bedroom-to-kitchen route including autonomous doorway transit, where standard Nav2 alone fails. All solutions are implemented as standalone ROS 2 nodes requiring no modification to the Nav2 framework. The source code is publicly available at https://github.com/akkexd/ros2-mecanum-robot.

Keywords: mecanum wheel, carpet friction, motor deadzone, rotation asymmetry, ROS 2 Nav2, sensor fusion, EKF, AMCL, omnidirectional robot, indoor navigation, proportional scaling, doorway transit

1 Introduction

The global mobile robotics market is projected to generate US$75 billion in revenue by 2030, driven by demand in indoor logistics, healthcare, and service applications [2]. Mecanum-wheel robots, which achieve omnidirectional mobility through 45-degree angled rollers, are increasingly deployed in these environments. However, the standard kinematic models underlying their navigation software assume pure rolling contact and no lateral slip at the wheel–ground interface [10]. These assumptions hold on smooth laboratory floors but fail systematically on high-friction carpeted surfaces, which are prevalent in apartments, offices, hotels, and institutional buildings.

This paper investigates three specific navigation failures caused by carpet friction and proposes software-based solutions that require no hardware modification. The central research question is: *how do motor deadzone compensation, rotation direction asymmetry, and multi-phase goal management affect navigation reliability for mecanum robots operating on high-friction carpet surfaces?* This question decomposes into three sub-questions: (1) What is the quantitative relationship between commanded velocity and actual motor output on carpet, and how does proportional deadzone compensation affect trajectory accuracy? (2) What physical mechanism causes rotation asymmetry on carpet, and how does wheel material modulate this effect? (3) Does a multi-phase navigation architecture separating rotation from translation improve navigation success rates compared to omnidirectional strafing alone?

The contributions of this paper are:

(a) empirical characterization of the motor deadzone boundary on loop-pile carpet across two surfaces,
(b) a proportional deadzone compensation algorithm that preserves velocity direction vectors,
(c) rotation asymmetry measurement across two wheel configurations (80 mm plastic and 100 mm TPU rubber) demonstrating material-dependent friction anisotropy,
(d) a multi-phase goal management architecture with waypoint graph routing and doorway transit protocol, and
(e) validation on a physical robot platform demonstrating 100% success on a 14 m multi-room navigation route.

2 Related Work

2.1 Mecanum Wheel Kinematics and Friction Modeling

The foundational inverse kinematics for mecanum wheels derive from the constraint that each wheel's contact point velocity equals the sum of the robot's translational velocity and the rotational component at that wheel's location [10, 7]. Taheri and Zhao [11] surveyed omnidirectional robot mechanisms and confirmed that published studies test predominantly on hard, smooth surfaces. Manzl et al. [9] developed an orthotropic friction model demonstrating that mecanum roller–ground friction is direction-dependent, varying significantly based on motion direction relative to the roller axis. Hernández and Almeida [5] experimentally confirmed across 120 trials that surface material is a primary variable in trajectory accuracy, though their tests were limited to laboratory floor and metal. Han et al. [4] established the quantitative link between wheel slip and position error in omnidirectional robots. No published study has tested mecanum wheels on carpeted surfaces.

2.2 Sensor Fusion and Localization

The Extended Kalman Filter (EKF) for fusing wheel odometry with inertial measurements is the standard approach to state estimation in mobile robotics [12]. Chen et al. [1] developed a Kalman filter-based slip estimation and compensation system for mecanum AGVs on hard floors, demonstrating that software-based slip compensation improves trajectory accuracy without hardware modification. Kheirandish et al. [6] developed an interacting multiple model Kalman filter for sensor fault tolerance. Fox [3] introduced KLD-sampling for adaptive particle count in Monte Carlo Localization.

2.3 ROS 2 Nav2 Navigation Stack

Macenski et al. [8] described the Nav2 architecture including lifecycle-managed nodes, the NavFn global planner with A* search, the DWB local planner, costmap layers, and the behavior tree navigator. Nav2 was designed primarily for differential-drive robots, and its DWB planner assumes rotation and translation can be commanded simultaneously. This assumption fails for mecanum robots on carpet where rotation is mechanically constrained by surface friction.

3 System Description

3.1 Hardware Platform

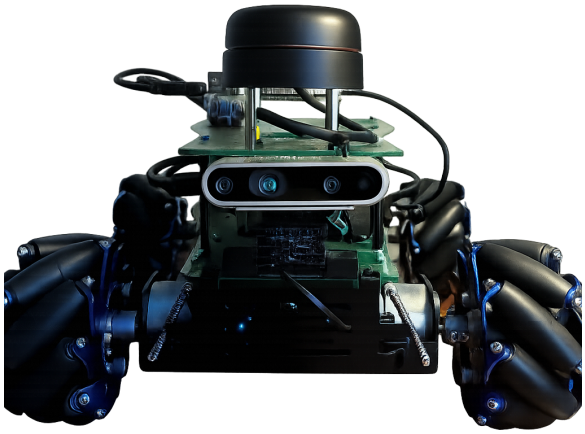
The experimental platform is a Yahboom ROSMASTER X3 mecanum robot. Two wheel configurations were tested: the original 80 mm diameter hard plastic mecanum wheels and upgraded 100 mm diameter aluminum hub wheels with TPU (thermoplastic polyurethane) rubber roller coating. Table 1 summarizes the hardware specifications.

Table 1: Hardware platform specifications.

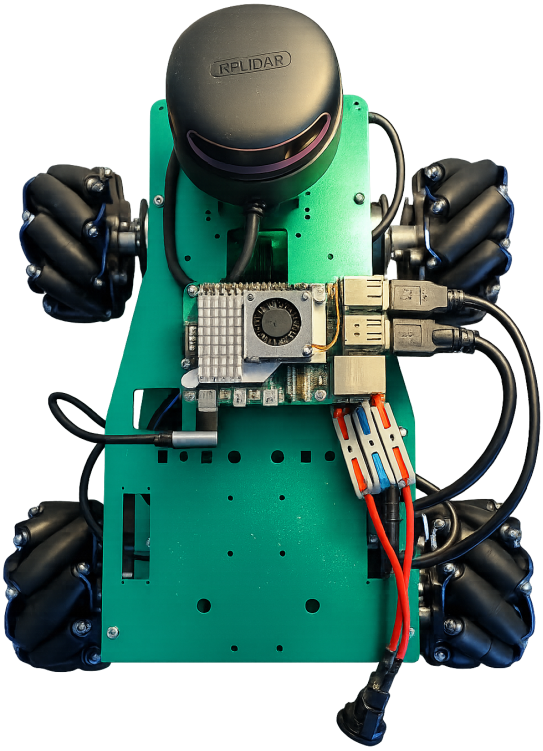
Parameter	Value
Chassis dimensions	300 mm × 245 mm × 50 mm
Total mass (with battery)	∼2.5 kg
Wheel configurations	80 mm plastic / 100 mm TPU rubber
Wheel radius r	0.0325 m (80 mm) / 0.050 m (100 mm)
Half-wheelbase L	0.070 m
Half-track width W	0.075 m
Processor	Raspberry Pi 5, 8 GB RAM
LiDAR	RPLIDAR A1, 360°, 8 m range
Depth camera	Intel RealSense D435
IMU	Onboard 6-axis (Yahboom expansion board)
Encoder polling rate	∼10 Hz
Motor	JGB37-520, 1:56 ratio, 8.3 kg·cm stall torque
Software	ROS 2 Jazzy, Nav2, robot_localization, SLAM Toolbox

3.2 Software Architecture

The software is organized as a ROS 2 workspace with three packages: `ros_robot_driver` (hardware abstraction, motor control, encoder odometry), `ros_robot_description` (URDF and



(a) Front view showing Intel RealSense D435 depth camera (center), RPLIDAR A1 (top), Raspberry Pi 5 (upper deck), and 100 mm TPU mecanum wheels.



(b) Top-down view showing RPLIDAR A1 mounting, Raspberry Pi 5 with active cooling, USB serial connections, and X-pattern mecanum wheel arrangement.

Figure 1: Yahboom ROSMASTER X3 experimental platform.

TF tree), and `ros_robot_bringup` (launch files, parameter configuration, SLAM maps). The architecture follows a three-layer design (Figures 2 and 3):

1. **Intention Layer** (Goal Manager V4): above Nav2, handles semantic navigation decisions including room-level routing, heading correction, doorway transit, and stall recovery.
2. **Execution Layer** (Nav2 + Mecanum Driver): converts velocity commands to compensated motor PWM.
3. **Verification Layer** (Encoders, IMU, EKF, AMCL): reports actual motion rather than commanded motion.

Nav2 remains unmodified; the Goal Manager intercepts `/goal_pose` before Nav2 receives it by remapping `bt_navigator` to `/goal_pose_unused`.

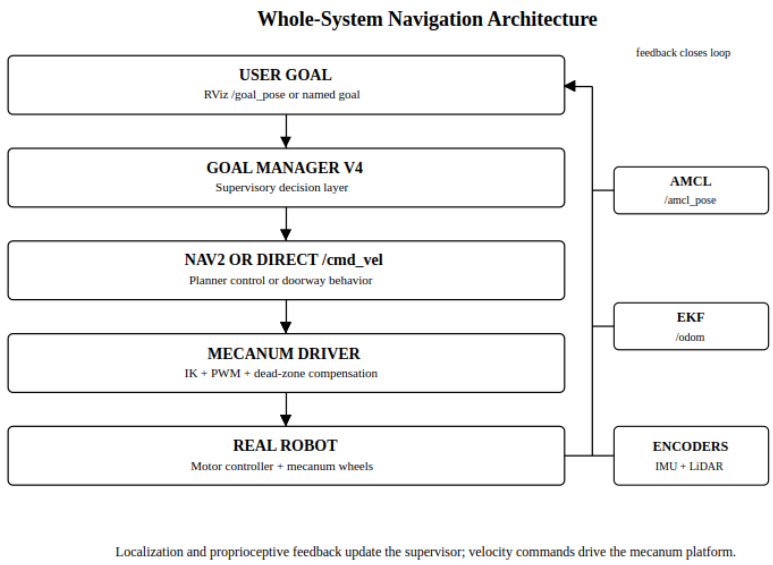


Figure 2: Closed-loop system architecture. User goals are intercepted by Goal Manager V4, which decides between Nav2 planning or direct `/cmd_vel`. The mecanum driver applies deadzone compensation. Feedback from encoders, IMU, EKF, and AMCL closes the loop.

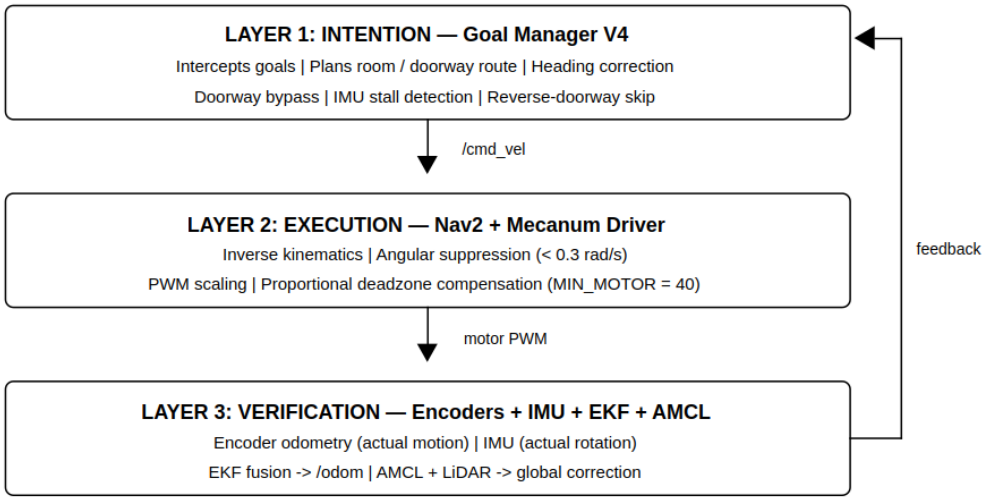


Figure 3: Three-layer architecture. Layer 1 (Intention): Goal Manager V4 handles semantic routing, heading correction, doorway transit, and stall recovery. Layer 2 (Execution): Nav2 + mecanum driver with proportional deadzone compensation. Layer 3 (Verification): encoder odometry, IMU, EKF fusion, and AMCL global correction.

3.3 Coordinate Frame Tree

The TF tree follows REP-105 conventions: `map` → `odom` (published by AMCL) → `base_footprint` (published by EKF) → `base_link` → {`wheels`, `laser_link`, `camera_link`, `imu_link`}. The driver node publishes with `publish_tf=False`; the EKF is the single authoritative source of the `odom` → `base_footprint` transform, preventing duplicate TF broadcasts.

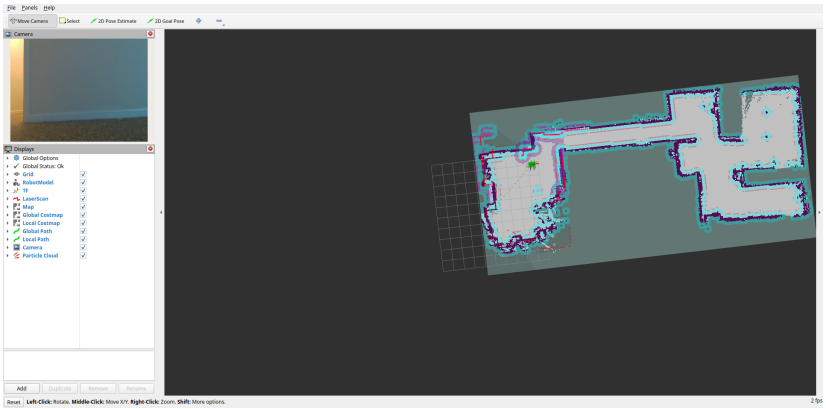


Figure 4: RViz visualization showing the SLAM map, global costmap, local costmap, RPLIDAR scan, AMCL particle cloud, and robot model with TF coordinate frames.

4 Theoretical Foundations

4.1 Mecanum Wheel Inverse Kinematics

For a four-wheel mecanum robot with wheel radius r, half-wheelbase L, and half-track width W, the inverse kinematics maps desired robot-frame velocities (v_x, v_y, ω) to individual wheel angular velocities:

$$\omega_1 = \frac{1}{r}\left(v_x - v_y - (L+W)\omega\right) \tag{1}$$

$$\omega_2 = \frac{1}{r}\left(v_x + v_y + (L+W)\omega\right) \tag{2}$$

$$\omega_3 = \frac{1}{r}\left(v_x + v_y - (L+W)\omega\right) \tag{3}$$

$$\omega_4 = \frac{1}{r}\left(v_x - v_y + (L+W)\omega\right) \tag{4}$$

The motor command m_i is obtained by scaling and clamping:

$$m_i = \text{clamp}(\text{round}(\omega_i \cdot S),\ -M_{\max},\ M_{\max}) \tag{5}$$

where $S = 25$ is the scaling factor and $M_{\max} = 90$ is the maximum motor command. On the Yahboom platform, an axis remapping is required: $\text{motor}_{v_x} = -$`cmd.linear.y` and $\text{motor}_{v_y} = -$`cmd.linear.x`, converting from ROS body-frame convention (X-forward) to the Yahboom board's native convention.

4.2 Forward Kinematics and Dead Reckoning

Dead reckoning integrates encoder-measured body-frame velocities into world-frame pose:

$$\Delta x = (v_x \cos\theta - v_y \sin\theta) \cdot \Delta t \tag{6}$$

$$\Delta y = (v_x \sin\theta + v_y \cos\theta) \cdot \Delta t \tag{7}$$

$$\Delta\theta = \omega \cdot \Delta t \tag{8}$$

Critically, v_x, v_y, and ω are measured from encoders via the motor controller's `get_motion_data()` function—not from `/cmd_vel`. This ensures odometry reflects actual wheel rotation rather than commanded rotation. On carpet, commanded motion frequently produces zero actual motion when motor PWM falls below the friction threshold.

4.3 Extended Kalman Filter Sensor Fusion

The `robot_localization` EKF fuses wheel odometry velocities (v_x, v_y) with IMU angular velocity (ω_z) to produce the `odom` $\rightarrow$ `base_footprint` transform. The trust assignment is deliberately minimal: wheel odometry contributes only linear velocities (not position or yaw, which degrade from 10 Hz encoder polling on carpet), and IMU contributes only yaw angular velocity (not absolute orientation, which would conflict with AMCL correction). Process noise covariance uses elevated values ($Q_{xx} = Q_{yy} = 0.5$) reflecting high carpet-induced drift.

4.4 Adaptive Monte Carlo Localization

AMCL is configured with `nav2_amcl::OmniMotionModel` and aggressive noise parameters (α_1 through $\alpha_5 = 0.8$), 4–8$\times$ higher than defaults, telling AMCL to heavily distrust odometry and rely on laser scan matching via the likelihood field model. The particle filter uses 1000–5000 adaptive particles (KLD-sampling [3]), $\sigma_{\text{hit}} = 0.15$, and `max_beams` = 120. Update triggers are set at 0.02 m translation / 0.02 rad rotation for frequent correction.

5 Proposed Solutions

5.1 Proportional Deadzone Compensation

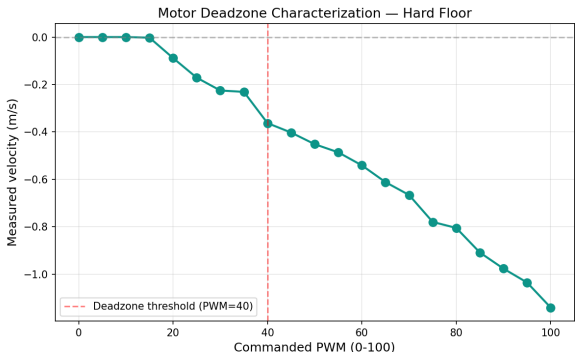
On carpet, motor commands below a threshold $M_{\min}$ produce zero wheel rotation due to carpet static friction exceeding motor torque. Standard Nav2 DWB velocity commands (max 0.04 m/s) routinely produce motor PWM values in the 15–30 range, well within the carpet deadzone. The naive fix of clamping each motor independently to $M_{\min}$ distorts the wheel-speed ratios that determine the mecanum robot's direction of travel. The proportional compensation algorithm preserves direction by uniformly scaling all motors:

$$m_{\max} = \max\big(|m_1|,\ |m_2|,\ |m_3|,\ |m_4|\big) \tag{9}$$

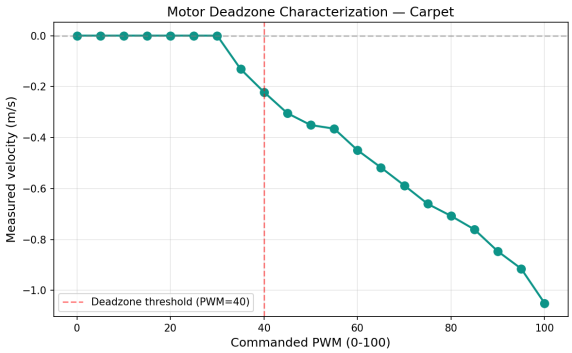
$$\text{if } m_{\max} > 0 \ \wedge\ m_{\max} < M_{\min}: \quad \lambda = M_{\min} \,/\, m_{\max} \tag{10}$$

$$m'_i = \text{round}(m_i \cdot \lambda) \quad \forall\, i \in \{1, 2, 3, 4\} \tag{11}$$

This guarantees that the ratio $m_1 : m_2 : m_3 : m_4$ is preserved while all values are boosted above the friction threshold. For example, a diagonal strafe command produces motor vector $[18, 30, 30, 18]$. Per-motor clamping yields $[40, 40, 40, 40]$ (ratio 1:1:1:1, distorted direction). Proportional scaling yields $[24, 40, 40, 24]$ (ratio 0.6:1:1:0.6, correct direction, boosted magnitude).



(a) Hard floor (laminate): motion onset at PWM 20.



(b) Loop-pile carpet: motion onset at PWM 35.

Figure 5: Motor deadzone characterization. Dashed line indicates the $M_{\min} = 40$ compensation threshold. The carpet deadzone is 75% wider than hard floor. Above the threshold, both surfaces exhibit similar linear velocity–PWM relationships (≈0.011 m/s per PWM unit).

5.2 Rotation Strategy with Smart Direction Selection

Empirical testing reveals that rotation behavior on carpet is direction-dependent and varies with wheel material (see Section 6.3). The rotation strategy uses a two-stage motor profile:

- **Kickstart** (0.5 s at 130% of sustained speed): a high-torque pulse to overcome carpet static friction. The breakaway torque on carpet is significantly higher than the sustain torque, analogous to a jerk profile in industrial motion control.
- **Sustain** (1.0 rad/s for 100 mm TPU wheels): once rotating, speed drops to reduce overshoot.

Cumulative rotation is tracked via IMU small-delta integration. Stall detection checks whether IMU cumulative yaw changed by at least 5° over any 3-second window; if not, the kickstart pulse is re-applied. A 60-second timeout serves as the ultimate fallback.

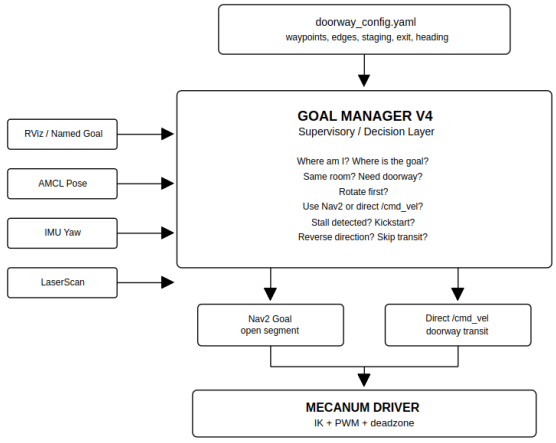
The smart direction selection algorithm computes the bearing to the next waypoint and selects the rotation direction based on empirically determined thresholds: angles less than 20° are skipped (Nav2 corrects during translation), 20°–65° use clockwise rotation with kickstart, and angles greater than 65° use counterclockwise rotation the long way (360° − angle). These thresholds reflect the carpet-specific mechanical limits observed during development.

5.3 Multi-Phase Goal Management Architecture

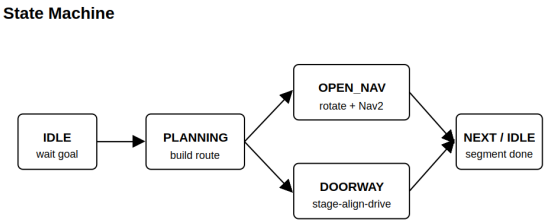
The `goal_manager_v4.py` node implements a finite state machine supervisor that intercepts user goals before Nav2 receives them. It interprets goals using a topological waypoint graph (`doorway_config.yaml`) defining named locations connected by edges, with designated doorway zones requiring special handling. Figure 6 shows the Goal Manager architecture and state machine.

The state machine operates in two primary modes:

OPEN_NAV. Phase A rotates the robot to face the next waypoint using IMU-tracked rotation with stall detection and kickstart recovery. Phase B delegates driving to Nav2's DWB planner in strafe-only mode (`max_vel_theta = 0.0`). Heading correction before translation prevents sideways or backward hallway driving on carpet, where lateral mecanum motion is particularly prone to stalling.



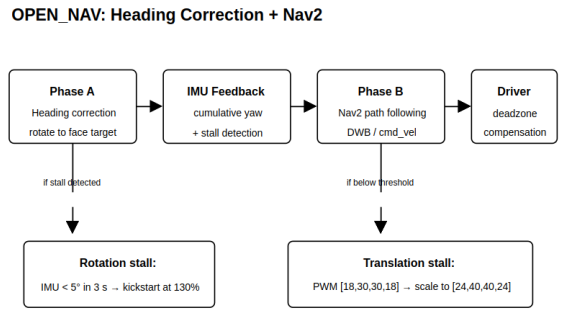
(a) Goal Manager V4 connections. Inputs: user goal, AMCL pose, IMU yaw, LaserScan, doorway configuration. Outputs: Nav2 segment goals (open navigation) or direct `/cmd_vel` (doorway transit).



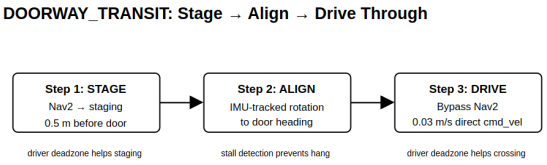
(b) Goal Manager state machine. IDLE → PLANNING → OPEN_NAV or DOORWAY_TRANSIT → segment complete → NEXT/IDLE.

Figure 6: Goal Manager V4 supervisor architecture and finite state machine.

DOORWAY_TRANSIT. A three-step protocol for constrained passages (Figure 7): (1) *Stage*—Nav2 drives to a staging point 0.5 m before the doorway center; (2) *Align*—IMU-tracked rotation to the transit heading with tighter completion threshold (0.10 rad vs. 0.30 rad for open navigation); (3) *Drive Through*—Goal Manager bypasses Nav2 and publishes direct `/cmd_vel` at 0.03 m/s toward the exit point. The doorway bypass exists because DWB oscillates in the 0.76 m doorway gap after costmap inflation ($0.76 - 2 \times 0.20 = 0.36$ m clearance for 0.18 m robot radius). Reverse doorway detection skips the forward-only protocol when the robot approaches from the exit side.



(a) OPEN_NAV: Phase A heading correction with IMU stall detection and kickstart recovery, followed by Phase B Nav2 path following with driver deadzone compensation.



(b) DOORWAY_TRANSIT: three-step protocol (Stage → Align → Drive Through) bypassing Nav2 for the constrained 0.76 m doorway crossing.

Figure 7: Navigation behavior modes for open segments and doorway transit.

The waypoint graph for the test environment defines eight waypoints (`bedroom_center`, `bedroom_staging`, `hallway_entry`, `hallway_mid`, `hallway_end`, `corner`, `kitchen_entry`, `kitchen_center`) connected by seven edges with one designated doorway zone (`bedroom_to_hallway`). BFS finds the shortest path on this graph. Angular suppression ($|$`angular.z`$| < 0.3$ rad/s $\rightarrow 0$) in the driver node filters Nav2's micro-rotations that cause jitter on carpet; all intentional rotation is

handled by Goal Manager.

5.4 DWB Strafe-Only Configuration

The DWB local planner is configured to search a purely translational velocity space with `max_vel_theta = 0.0`. Table 2 summarizes the key parameters.

Table 2: DWB and motor configuration parameters.

Parameter	Value	Rationale
`max_vel_x / max_vel_y`	0.04 m/s	Carpet speed limit
`max_vel_theta`	0.0 rad/s	Rotation disabled in Nav2
`max_speed_xy`	0.045 m/s	Resultant limit
`vx_samples / vy_samples`	20 / 10	Velocity search resolution
`xy_goal_tolerance`	0.25 m	Position tolerance
`yaw_goal_tolerance`	π rad	Heading check disabled
MIN_MOTOR (deadzone)	40 PWM	Carpet friction threshold
Angular suppression	< 0.3 rad/s $\rightarrow 0$	Filter Nav2 micro-rotation
Global inflation	0.20 m	Costmap inflation radius
Local inflation	0.35 m	Costmap inflation radius

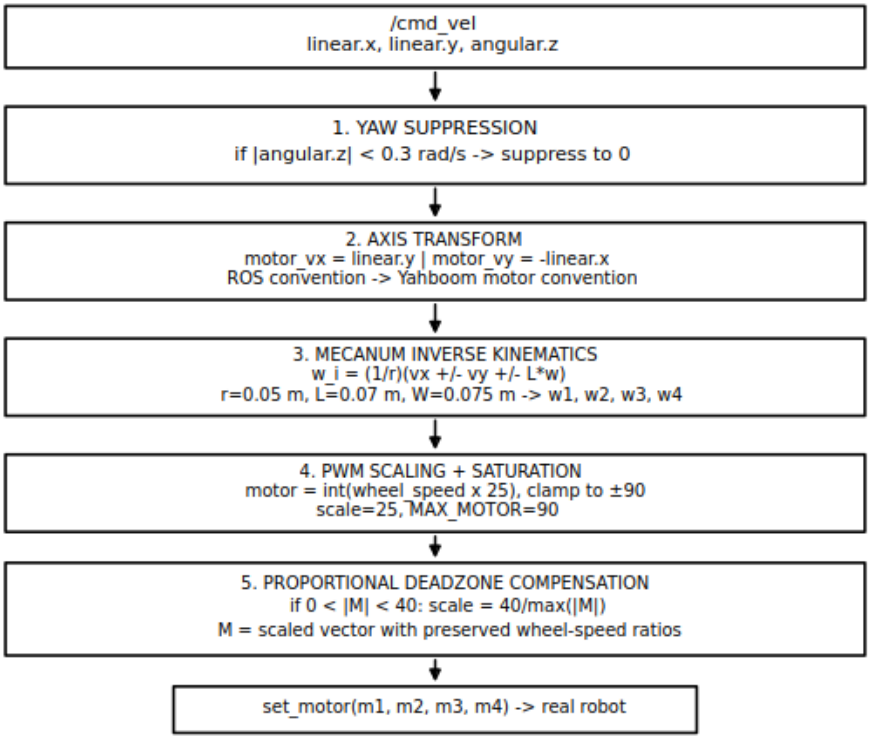


Figure 8: Mecanum driver pipeline. Processing stages: angular suppression (< 0.3 rad/s $\rightarrow 0$), axis transform (ROS $\rightarrow$ Yahboom convention), mecanum inverse kinematics, PWM scaling and saturation (± 90 max), and proportional deadzone compensation ($M_{\min} = 40$).

6 Experimental Setup and Results

6.1 Test Environment

The test environment is a single-bedroom apartment with loop-pile carpet throughout. The navigation route spans approximately 14 m from the bedroom to the kitchen through a 0.76 m wide doorway and an L-shaped hallway with a 90° corner turn. The SLAM map was generated using SLAM Toolbox (`online_async` mode) at 0.05 m resolution. All trials were conducted with the robot starting in the bedroom and navigating to the kitchen area (Figures 9 and 10).

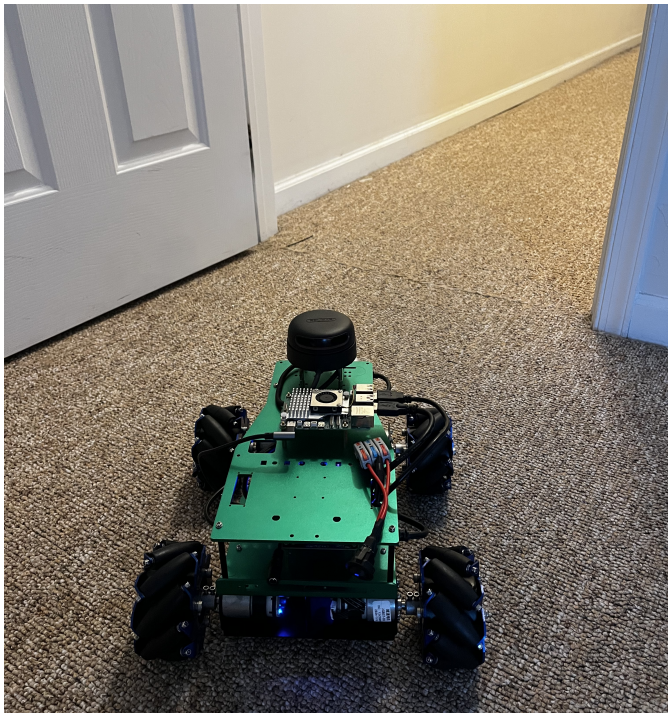


Figure 9: Test environment showing the robot at the bedroom-to-hallway doorway (0.76 m width) on loop-pile apartment carpet. The carpet fiber texture and doorway clearance relative to the robot chassis are visible.

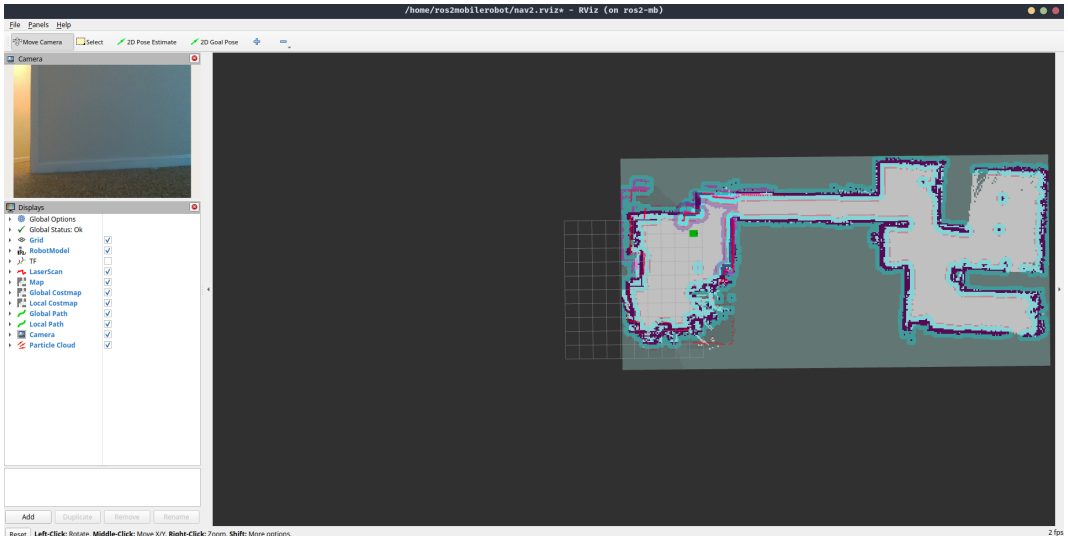


Figure 10: SLAM map of the test apartment at 0.05 m resolution. The navigation route traverses from bedroom (lower left) through the doorway, along the hallway, around the L-shaped corner, to the kitchen (lower right). Costmap inflation (pink/cyan) is visible around walls and obstacles.

6.2 Deadzone Characterization

Motor command sweep tests were conducted by commanding all four motors at identical PWM values (0–100 in steps of 5) and recording encoder-measured velocity over 1.5 s per level. Tests were performed on two surfaces with the 100 mm TPU wheel configuration (Figure 5).

On hard floor (laminate), measurable wheel motion began at PWM 20 with velocity 0.09 m/s. On loop-pile carpet, no wheel motion occurred below PWM 30; the first measurable velocity appeared at PWM 35 (0.13 m/s), a 75% wider deadzone. The transition was abrupt: PWM 30 produced zero encoder velocity while PWM 35 produced 0.13 m/s. Above the deadzone threshold, the velocity–PWM relationship was approximately linear with similar slopes on both surfaces ($\approx$0.011 m/s per PWM unit), indicating that surface friction primarily affects breakaway behavior rather than kinetic motion.

This binary deadzone characteristic contradicts the continuous slip model assumed by Chen et al. [1], where slip is an incremental deviation correctable by Kalman filtering. On carpet, slip transitions abruptly to complete stalling below the friction threshold, producing a failure mode that continuous estimation cannot detect because encoders report zero velocity for both intentional stops and friction-induced stalls. The $M_{\min} = 40$ parameter provides a safety margin above the measured 30–35 threshold to accommodate battery voltage variation and carpet density differences.

6.3 Rotation Asymmetry Measurement

Ten-trial rotation tests (5 CW, 5 CCW at PWM 90 for 10 s each) were conducted on carpet with both wheel configurations (Figures 12 and 13).

Figure 11: Close-up of the loop-pile apartment carpet surface showing directional fiber grain texture. The anisotropic fiber orientation creates direction-dependent friction on the 45° angled mecanum rollers, producing the CW/CCW rotation asymmetry observed in Figures 12 and 13.

80 mm plastic wheels. CCW rotation completed 90° in all 5 trials, reaching a mean of 180° in 10 s. CW rotation stalled in all 5 trials, achieving only a mean of 22°. The separation was consistent and dramatic: CCW angular velocity was approximately 9× higher than CW. This is consistent with the orthotropic friction model of Manzl et al. [9]: CCW rotation aligns roller slip with the carpet fiber grain direction, while CW rotation opposes it.

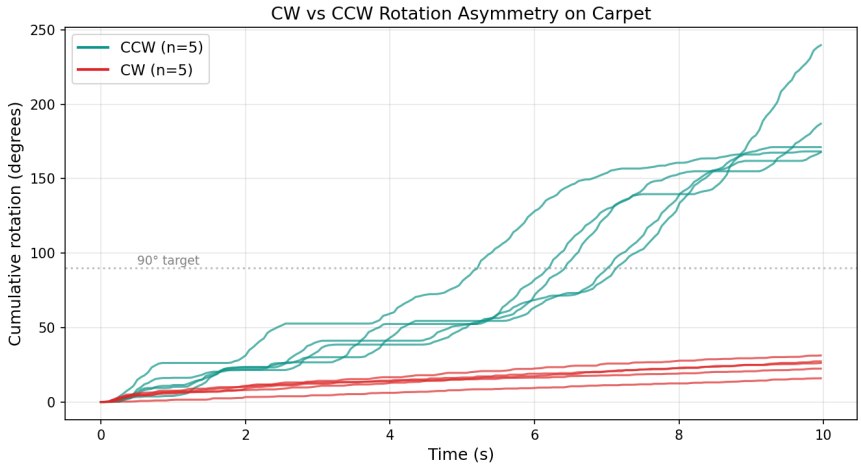


Figure 12: Rotation asymmetry on carpet with 80 mm plastic mecanum wheels. CCW rotation (teal, $n = 5$) consistently exceeded the 90° target, reaching mean 180° in 10 s. CW rotation (red, $n = 5$) stalled in all trials at mean 22°. The 9× velocity difference confirms direction-dependent friction from carpet fiber grain interaction with hard plastic rollers.

100 mm TPU rubber wheels. The asymmetry pattern changed significantly. CCW rotation was consistent but slower (mean 39° in 10 s, none reaching 90°). CW rotation became bimodal: three trials completed 100°–118° while two trials stalled at 25°–35°. The TPU rubber's higher compliance produces more uniform ground contact than hard plastic, reducing the directional

bias but not eliminating rotation difficulty on carpet. This demonstrates that wheel material directly modulates the orthotropic friction interaction.

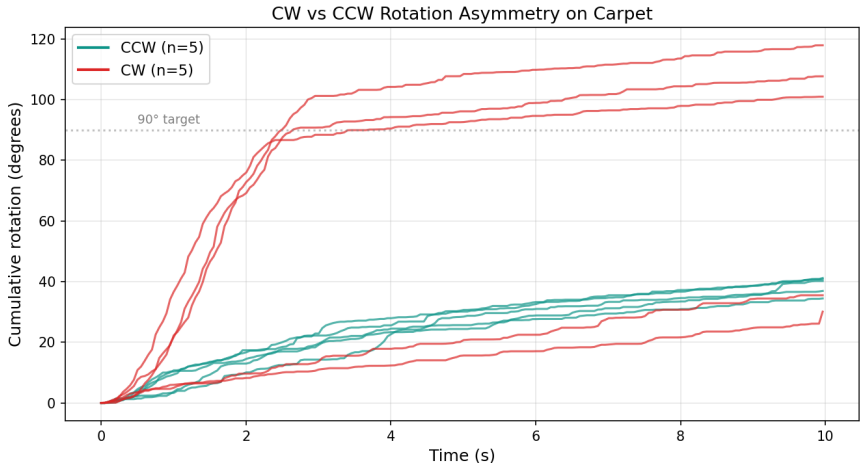


Figure 13: Rotation asymmetry on carpet with 100 mm TPU rubber mecanum wheels. CCW rotation (teal, $n = 5$) was consistent but slower (mean 39°). CW rotation (red, $n = 5$) became bimodal: three trials exceeded 100° while two stalled at 25°–35°. TPU rubber reduces directional bias through higher contact compliance but introduces unpredictability.

These findings validate two design decisions: (1) IMU-based stall detection and kickstart recovery are essential for both wheel configurations because either rotation direction can stall unpredictably on carpet, and (2) the smart direction selection algorithm provides statistical advantage but the stall recovery mechanism is the true safety net.

6.4 Navigation Trial Results

Five navigation trials were conducted for the bedroom-to-kitchen route using the full system (Goal Manager V4 + deadzone compensation + 100 mm TPU wheels). All data was recorded as ROS 2 bag files containing `/cmd_vel`, `/odom`, `/wheel_odom`, `/amcl_pose`, `/imu/data_raw`, `/scan`, `/tf`, and `/tf_static` topics. Results are summarized in Table 3.

Table 3: Navigation trial results, bedroom to kitchen route.

Trial	Result	Duration (s)	Distance (m)	Notes
1	SUCCESS	~130	14.2	Full route, doorway clean
2	SUCCESS	~160	14.5	Longer at corner turn
3	SUCCESS	~135	14.1	Cleanest trajectory
4	SUCCESS	~110	13.8	Fastest completion
5	SUCCESS	~150	14.3	Minor wobble at hallway
Baseline (Nav2 only)	FAIL	N/A	<1.0	Robot oscillated, no progress

The full system achieved 5/5 (100%) success on the forward route. The baseline trial, with Goal Manager disabled and goals published directly to Nav2, resulted in immediate failure: the robot oscillated in place as DWB produced motor commands below the carpet deadzone (PWM 15–30), and no heading correction prevented sideways hallway approach. Figures 14 to 17 present detailed data from Trial 3, the representative cleanest run.

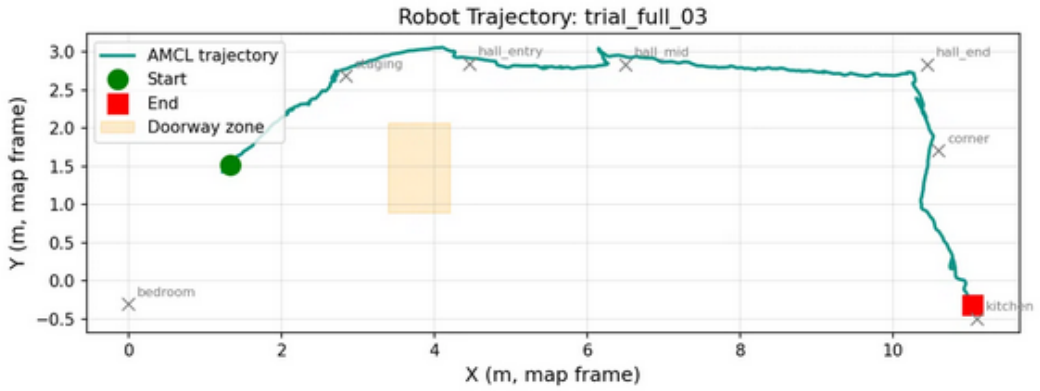


Figure 14: AMCL-localized trajectory for navigation Trial 3 (representative of 5/5 successful trials). The robot follows the waypoint graph from bedroom start (green circle) through staging, doorway zone (orange), hallway, corner turn, to kitchen end (red square). Total path length approximately 14 m.

Figure 15: Motor PWM commands during Trial 3 with proportional deadzone compensation. Rotation events appear as $\pm 72/\pm 90$ spikes (Phase A). During translation (Phase B), the maximum motor magnitude is maintained at or above $M_{\min} = 40$ (dashed lines). Values between 0 and ± 40 on individual motors reflect proportional scaling where smaller motors preserve the direction vector ratio.

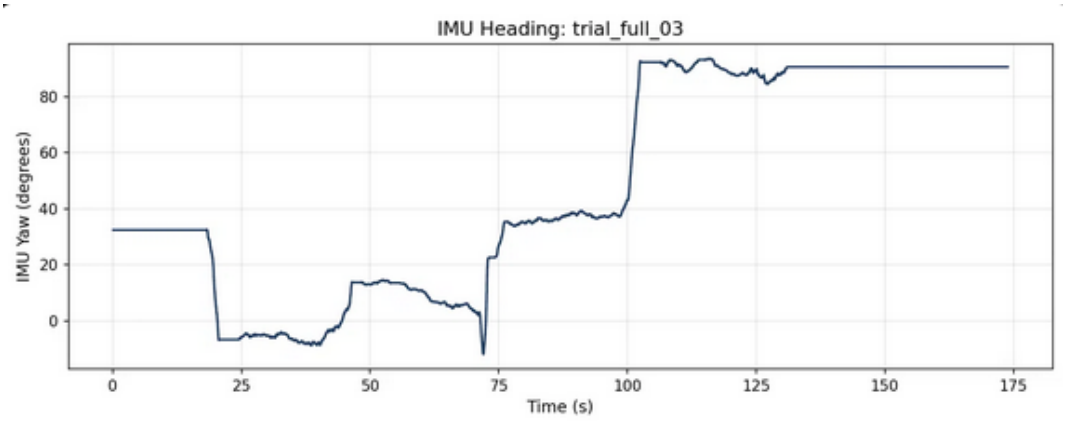


Figure 16: IMU heading during Trial 3 showing the multi-phase staircase pattern. Sharp step changes correspond to Phase A rotations (heading correction before each segment). Flat sections correspond to Phase B translation (heading held constant by DWB with `max_vel_theta = 0.0`). The large step at ∼100 s is the 90° corner turn from hallway to kitchen approach.

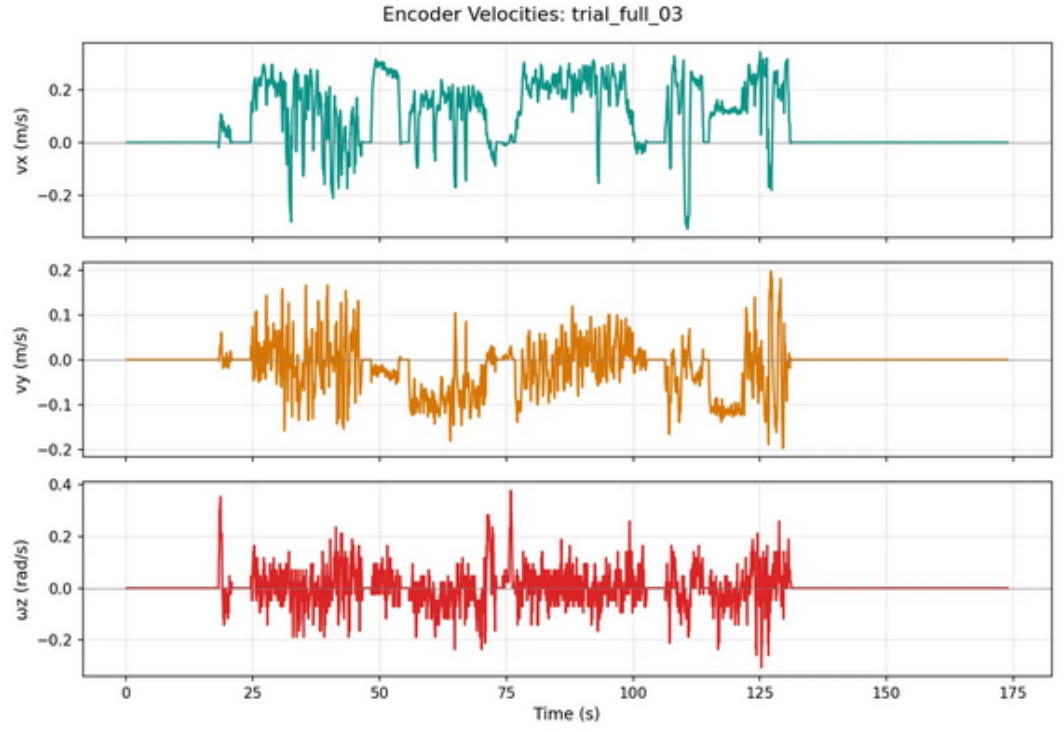


Figure 17: Encoder-measured velocities during Trial 3. Top: forward velocity v_x showing active translation phases with dips to zero during rotation pauses. Middle: lateral velocity v_y showing carpet-induced drift during strafing. Bottom: angular velocity ω_z showing rotation events as spikes, with near-zero values during translation confirming phase separation.

7 Discussion

The experimental results demonstrate that carpet-induced navigation failures are deterministic and software-addressable. The motor deadzone is not random but a repeatable friction threshold imposed by the mechanical interaction between mecanum rollers and carpet fibers. The proportional scaling algorithm restores trajectory tracking by preserving wheel-speed ratios while

boosting all commands above this threshold.

The rotation asymmetry comparison across wheel types reveals that the friction anisotropy predicted by Manzl et al. [9] is material-dependent. Hard plastic rollers create strong directional coupling with carpet fiber grain, producing consistent unidirectional success. TPU rubber rollers reduce this coupling through higher contact compliance, but introduce unpredictability rather than eliminating the problem. This finding has practical implications for wheel selection in carpet-operating robots: rubber-coated wheels do not solve the rotation problem but change its character from deterministic to stochastic.

The three-layer architecture demonstrates a general pattern for augmenting navigation frameworks with environment-specific logic. Rather than modifying Nav2 internally, the Goal Manager wraps it with semantic decision-making (room routing, doorway detection, stall recovery) while the driver wraps it with actuator-level compensation (deadzone scaling, angular suppression). This composable approach preserves Nav2's modularity: any planner can be substituted without changing the supervisor or driver layers.

The primary limitations are:

1. *Single platform and carpet type*—the deadzone threshold, rotation thresholds, and kickstart parameters are empirically determined for one robot on one carpet.
2. *Trial count of* $n = 5$ limits statistical power; this is pilot data demonstrating feasibility, not a statistically powered study.
3. *Reverse route incomplete*—the kitchen-to-bedroom direction remains unresolved due to the forward-only doorway transit protocol.
4. *Fixed angular thresholds*—the current smart direction selection uses thresholds that may not generalize to other carpet types.

8 Conclusion and Future Work

This paper presented three software-based solutions for mecanum robot navigation on high-friction carpet: proportional deadzone compensation, rotation management with stall detection and kickstart recovery, and multi-phase goal management with waypoint graph routing. All solutions are implemented as standalone ROS 2 nodes maintaining full compatibility with the standard Nav2 stack. Experimental validation on loop-pile apartment carpet demonstrated 100% navigation success (5/5 trials) for the 14 m bedroom-to-kitchen route including autonomous doorway transit, where standard Nav2 alone achieves 0% success.

The proportional deadzone compensation contributes a general-purpose technique applicable to any actuator system with a nonlinear deadzone: treat the motor command as a vector and scale by a single gain factor rather than clamping per-axis. The rotation asymmetry comparison across wheel materials contributes the first published evidence that carpet fiber grain creates direction-dependent friction on mecanum rollers that varies with roller material compliance. The multi-phase architecture demonstrates a composable pattern for layering constraint-aware logic on top of general-purpose navigation frameworks.

Future work includes: (a) testing on additional carpet types (low-pile, high-pile, carpet tiles) to characterize how fiber density affects the deadzone threshold, (b) developing adaptive deadzone detection by monitoring the commanded-to-measured velocity ratio during initial motion, (c) integrating depth camera terrain classification for automatic carpet/hard-floor detection and dynamic parameter switching, (d) completing bidirectional doorway transit, and

(e) replacing heuristic rotation thresholds with a surface-dependent dynamics model or nonlinear model predictive control.

The source code is publicly available at https://github.com/akkexd/ros2-mecanum-robot. All configuration files, launch files, URDF, waypoint graph, and experimental data are included for full reproducibility.

References

[1] Chen, P., Yang, S., Chen, Y., Muslikhin, M., & Wang, M. (2021). Slip estimation and compensation control of omnidirectional wheeled automated guided vehicle. *Electronics, 10*(7), 840. https://doi.org/10.3390/electronics10070840

[2] Chowdhury, G. (2025, August 1). The global robotics market outlook. https://www.abiresearch.com/blog/global-robotics-market-outlook

[3] Fox, D., & Department of Computer Science & Engineering, University of Washington. (2001). Adapting the sample size in particle filters through KLD-sampling. Department of Computer Science & Engineering, University of Washington. https://www.robots.ox.ac.uk/~cvrg/hilary2005/adaptive.pdf

[4] Han, K.-L., Kim, H., & Lee, J. (2010). The sources of position errors of omni-directional mobile robot with mecanum wheel. pp. 581–586. https://doi.org/10.1109/ICSMC.2010.5642009

[5] Hernández, J. C. O., & Almeida, D. I. R. (2024). Experimental control approach of a mecanum-wheeled mobile robot for slippage error and energy consumption reduction on different surfaces. *Journal of Mechanical Science and Technology, 38*(11), 6309–6318. https://doi.org/10.1007/s12206-024-1042-8

[6] Kheirandish, M., Yazdi, E., Mohammadi, H., & Mohammadi, M. (2022). A fault-tolerant sensor fusion in mobile robots using multiple model Kalman filters. *Robotics and Autonomous Systems, 161*, 104343. https://doi.org/10.1016/j.robot.2022.104343

[7] Lynch, K. M., & Park, F. C. (2017). *Modern robotics: Mechanics, planning, and control*. Cambridge University Press. https://hades.mech.northwestern.edu/images/7/7f/MR.pdf

[8] Macenski, S., Martín, F., White, R., & Ginés Clavero, J. (2020). The Marathon 2: A navigation system. https://doi.org/10.48550/arXiv.2003.00368

[9] Manzl, P., Sereinig, M., & Gerstmayr, J. (2024). A mecanum wheel model based on orthotropic friction with experimental validation. *Mechanism and Machine Theory, 193*, 105548. https://doi.org/10.1016/j.mechmachtheory.2023.105548

[10] Siegwart, R., Nourbakhsh, I. R., & Scaramuzza, D. (2011). *Introduction to autonomous mobile robots* (2nd ed.). MIT Press. ISBN: 9780262015356.

[11] Taheri, H., & Zhao, C. (2020). Omnidirectional mobile robots, mechanisms and navigation approaches. *Mechanism and Machine Theory, 153*, 103958. https://doi.org/10.1016/j.mechmachtheory.2020.103958

[12] Thrun, S., Burgard, W., & Fox, D. (2005). *Probabilistic robotics*. MIT Press. ISBN: 9780262201629.