# Reliability-Aware LiDAR Odometry Fusion for Mobile Robot Localization in Warehouse Environments

by

Maryam Salehi

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Mechanical and Mechatronics Engineering

Waterloo, Ontario, Canada, 2026

© Maryam Salehi 2026

## Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

**Abstract**

Accurate self-localization is essential for reliable navigation of autonomous mobile robots in structured indoor environments such as warehouses and logistics facilities. In these settings, global positioning systems are unavailable, wheel odometry accumulates drift, and LiDAR scan matching can become unreliable when temporary obstacles or map changes create scan-to-map mismatch. Therefore, localization systems must combine the short-term continuity of odometry with the corrective value of map-referenced sensing while adapting to changes in sensor reliability.

This thesis investigates encoder- and LiDAR-based localization for a differential-drive warehouse robot using a progressive development and evaluation pipeline spanning MATLAB simulation and ROS 2/Gazebo implementation. First, a unified MATLAB framework is developed to compare four localization strategies under consistent conditions: encoder dead reckoning, LiDAR scan-to-map registration using Iterative Closest Point (ICP), Extended Kalman Filter (EKF) fusion of encoder prediction with ICP pose measurements, and particle-filter localization using a LiDAR likelihood model. The results show that encoder-only odometry is computationally efficient but accumulates drift, while scan-based and probabilistic methods can reduce long-term error by anchoring the pose estimate to environmental geometry. EKF fusion provides a lightweight and continuous estimate that combines encoder smoothness with LiDAR-based correction.

Second, the localization workflow is implemented in ROS 2 and evaluated in Gazebo under simulated-time operation, with RViz used for visualization. The implementation establishes a modular architecture for simulation, LiDAR preprocessing, local ICP odometry, map-based localization, EKF fusion, trust monitoring, and visualization. RTAB-Map is used for LiDAR-based local ICP odometry, AMCL is used for map-based probabilistic localization, and the `robot_localization` EKF fuses wheel odometry with ICP odometry to produce a smoothed `/odometry/filtered` estimate. A custom online ICP trust monitor provides a heuristic diagnostic score for local scan-matching quality. The ROS 2 results demonstrate the expected qualitative behavior of wheel odometry, local LiDAR odometry, and EKF fusion while also reflecting practical middleware effects such as asynchronous sensor updates, transform handling, and timing variation.

Third, this thesis introduces a reliability-aware localization enhancement for handling scan-to-map mismatch and accumulated odometry drift. When ICP becomes unreliable due to temporary obstacles or degraded scan matching, its influence is reduced to limit the effect of incorrect scan alignments on the estimate. When ICP remains consistently reliable, it is used as a geometric anchor to gradually reduce accumulated encoder drift. A

Mamdani fuzzy inference system maps ICP registration RMSE, position innovation, and yaw innovation to a continuous trust score. Simulation results show that the proposed strategy maintains low estimation error during the tested map-mismatch condition and reduces encoder–ICP divergence during stable periods without abrupt switching. These results are limited to the simulated environments, trajectories, and mismatch conditions considered in this thesis.

Overall, the contributions of this thesis are: (i) a unified MATLAB evaluation framework for encoder-, ICP-, EKF-, and particle-filter localization in a representative warehouse environment; (ii) a reliability-aware localization strategy that adapts LiDAR trust based on scan-matching quality and innovation consistency; and (iii) a modular ROS 2/Gazebo implementation that integrates wheel odometry, local LiDAR odometry, AMCL, EKF fusion, and online ICP trust monitoring as a foundation for future hardware deployment. Future work will focus on physical robot validation, richer sensor-artifact modeling, systematic evaluation under a wider range of operating conditions, and full integration of the reliability-aware trust framework into the ROS 2 fusion loop.

**Acknowledgements**

I would like to express my sincere gratitude to my supervisor, Professor Amir Khajepour, for his guidance, support, and valuable feedback throughout my graduate studies and the development of this thesis.

I am grateful to my colleagues, friends, and everyone who supported me throughout this journey.

Finally, I would like to express my deepest appreciation to my family for their continuous encouragement, patience, and unconditional support throughout my studies.

# Table of Contents

**Author's Declaration** ii

**Abstract** iii

**Acknowledgements** v

**List of Figures** xi

**List of Tables** xvi

**1 Background and Literature** 1

1.1 Introduction . . . . . 1
1.2 Mobile Robot Localization in Warehouse Systems . . . . . 2
1.3 Odometry-Based Localization . . . . . 3
1.4 Probabilistic Localization and Bayesian Estimation . . . . . 4
1.5 LiDAR-Based Localization and Scan Matching . . . . . 5
1.6 Fusion of Odometry and LiDAR . . . . . 6
1.7 Broader Localization Frameworks in Context . . . . . 7
1.8 Challenges in Structured Indoor Environments . . . . . 8
1.9 Summary and Research Gap . . . . . 8

**2 System Modeling and Simulation in MATLAB** **11**
2.1 Introduction . . . 11
2.2 Simulation Environment and Experimental Setup . . . 13
2.2.1 Warehouse Map and Waypoints . . . 13
2.2.2 Robot Model and Sensors . . . 15
2.3 Encoder-Based Odometry . . . 17
2.3.1 Kinematic Formulation . . . 17
2.3.2 Noise Model and Drift Accumulation . . . 17
2.3.3 Performance and Limitations . . . 18
2.4 LiDAR-Based Localization (ICP Scan-to-Map) . . . 20
2.4.1 Concept and Cost Function . . . 20
2.4.2 Implementation Notes . . . 20
2.4.3 Results and Discussion . . . 21
2.5 Particle Filter-Based Localization . . . 22
2.5.1 Bayesian Formulation . . . 22
2.5.2 Motion and Measurement Models . . . 23
2.5.3 Simulation Results and Diagnostics . . . 23
2.5.4 Discussion . . . 23
2.6 Extended Kalman Filter Fusion of Encoder and ICP . . . 25
2.6.1 State, Process, and Measurement Models . . . 25
2.6.2 Adaptive Measurement Covariance from ICP RMSE . . . 26
2.6.3 EKF Prediction and Correction . . . 27
2.6.4 Baseline Results . . . 27
2.6.5 Comparison with a Conventional EKF under Temporary Map Mismatch . . . 30
2.6.6 Quantitative Summary for the Baseline Trajectory . . . 31
2.6.7 Additional Trajectory Validation . . . 32
2.7 Evaluation Metrics and Experimental Protocol . . . 42
2.8 Runtime and Computational Considerations . . . 42
2.9 Limitations, Threats to Validity, and Chapter Summary . . . 43

**3 Reliability-Aware Localization Enhancements** **46**
3.1 Motivation and Problem Statement . . . 46
3.2 Experimental Configuration for the Reliability Tests . . . 47
3.3 Reliability Metrics and Decision Logic . . . 49
3.3.1 ICP Registration Error (RMSE) . . . 50
3.3.2 Innovation Relative to Encoder-Based Prediction . . . 51
3.3.3 Trust Mapping via Fuzzy Inference System (FIS) . . . 53
3.3.4 Decision Logic for Unreliable and Sustained-Good ICP . . . 57
3.4 Encoder Drift Correction Using Sustained High-Quality ICP . . . 59
3.4.1 Why Correct the Encoder Pose? . . . 60
3.4.2 Detection of Sustained High-Quality ICP . . . 60
3.4.3 Gradual Encoder Correction Toward ICP . . . 61
3.4.4 Interaction With the Controller and EKF . . . 63
3.5 Quantitative Evaluation Metrics . . . 63
3.5.1 Pose Error Definitions . . . 63
3.5.2 Aggregate Metrics . . . 64
3.5.3 Mismatch-Window Metrics . . . 65
3.5.4 Encoder Anchoring Diagnostics . . . 65
3.6 Results and Discussion . . . 66
3.6.1 Trajectory Overlays and Closed-Loop Behavior . . . 66
3.6.2 ICP Quality and EKF Weighting During Map Mismatch . . . 68
3.6.3 Impact of Fuzzy Trust on Estimation Error . . . 69
3.6.4 Encoder Drift Correction During Sustained-Good ICP . . . 70
3.6.5 Summary of Quantitative Results . . . 71
3.6.6 Risks and Practical Considerations . . . 72
3.7 Limitations and Threats to Validity . . . 73
3.8 Chapter Summary . . . 74

**4 ROS 2-Based Implementation in Gazebo and RViz** **75**
4.1 Introduction and Objectives . . . 75
4.2 Operational Modes and Experimental Workflow . . . 76
4.3 System Architecture Overview . . . 78
4.3.1 Overall Data Flow . . . 78
4.3.2 Key Topics and Data Interfaces . . . 80
4.3.3 TF Frame Hierarchy . . . 80
4.4 Package Structure and Workspace Organization . . . 80
4.5 Robot Description and Simulation Environment . . . 81
4.5.1 Robot Model . . . 81
4.5.2 Planar LiDAR Model . . . 82
4.5.3 Gazebo Warehouse Environment . . . 82
4.6 Occupancy-Grid Mapping with SLAM Toolbox . . . 83
4.7 Local LiDAR Odometry and RTAB-Map Integration . . . 83
4.7.1 Local ICP Odometry . . . 83
4.7.2 Practical Configuration . . . 84
4.7.3 ICP Diagnostic Information . . . 84
4.7.4 Online Local ICP Trust Monitor . . . 84
4.8 AMCL Map Localization . . . 86
4.9 EKF-Based State Estimation . . . 87
4.9.1 Configuration and State Variables . . . 88
4.9.2 Covariance Calibration . . . 88
4.10 Results and Performance Analysis . . . 90
4.10.1 Evaluation Procedure . . . 90
4.10.2 MATLAB Post-Processing and Synchronization . . . 90
4.10.3 Interpretation of ICP Diagnostics . . . 91
4.10.4 Quantitative Metrics and ICP Tracking Availability . . . 92
4.10.5 Computational Behavior . . . 94
4.11 Limitations . . . 94
4.12 Discussion and Reflection . . . 95

**5 Conclusion and Future Work** **97**

5.1 Conclusion . . . . . 97

5.2 Future Work . . . . . 100

**References** **102**

**APPENDICES** **110**

**A Jacobian Derivations for the EKF** **111**

A.1 First-Order Discrete-Time Motion Model . . . . . 111

A.2 Measurement Jacobian . . . . . 112

**B ROS 2 Workspace Details** **113**

**C ROS 2 Configuration Excerpts** **115**

C.1 Robot Model (URDF/Xacro) Excerpt . . . . . 115

C.2 Gazebo World File Excerpt . . . . . 118

C.3 EKF Configuration Excerpt . . . . . 119

# List of Figures


2.1 General structure of the localization and evaluation framework. A simulated robot follows a predefined trajectory in a warehouse map while multiple estimators produce pose estimates. Each estimator is evaluated using common error metrics. . . . . 13

2.2 Simulated warehouse environment represented as a binary occupancy grid with the baseline reference trajectory. . . . . 15

2.3 Visualization of the simulated robot and LiDAR rays in the warehouse map. 16

2.4 Encoder-based odometry compared with the ground-truth trajectory. Small dead-reckoning errors accumulate over time and become more visible after the turn. . . . . 19

2.5 Absolute Position Error (APE) over time for encoder-only odometry. . . . 19

2.6 ICP scan-to-map alignment visualization on the warehouse map. . . . . 21

2.7 Registration RMSE over time. Transient increases indicate harder frames, such as poor overlap or ambiguous geometry. . . . . 22

2.8 Absolute Position Error (APE) for the PF run compared with encoder dead reckoning using the same ground-truth trajectory. . . . . 24

2.9 PF diagnostics: effective sample size $N_{\text{eff}}$ and weighted scan RMSE. . . . . 24

2.10 Baseline tracking error (GT vs desired) for closed-loop runs driven by different estimator feedback. Top: position error. Bottom: yaw error. . . . . 28

2.11 Baseline estimation error (estimator vs GT). Top: position error. Bottom: yaw error. . . . . 28

2.12 Enlarged view of baseline estimation errors excluding encoder-only odometry. Position and yaw errors for ICP, EKF, and PF are shown on a reduced vertical scale to improve visibility. . . . . 29

2.13 Estimation error comparison between a conventional EKF and the proposed adaptive EKF under a temporary map-mismatch scenario. The dashed vertical lines indicate the interval in which an extra obstacle was added to the live map. . . . . 31

2.14 Additional corridor-style trajectory used to examine localization behavior during extended straight-line motion in a warehouse aisle. . . . . 33

2.15 Additional repeated-turn trajectory used to examine localization behavior under multiple heading changes. . . . . 34

2.16 Repeated-loop stress-test trajectory used to examine long-duration accumulation effects without resetting the estimator state between laps. . . . . 34

2.17 Estimator trajectory overlay for the corridor-style path. Encoder-only localization shows the largest deviation, while ICP and EKF remain closer to the desired path. . . . . 35

2.18 Estimator trajectory overlay for the repeated-turn path. The same overall trend as in the baseline case is preserved, with encoder-only localization deviating more than the LiDAR-assisted methods. . . . . 36

2.19 Estimator error for the corridor-style path. The position-error trend shows larger accumulated error for encoder-only localization, while ICP and EKF remain better bounded. . . . . 37

2.20 Estimator error for the repeated-turn path. Encoder-only localization again shows the largest cumulative drift, while ICP and EKF maintain lower error. 37

2.21 Enlarged view of estimation errors for the corridor-style trajectory, showing ICP, EKF, and PF without encoder-only odometry. . . . . 38

2.22 Estimator error for the repeated-loop stress test. Encoder error increases progressively over successive laps, while the LiDAR-assisted methods remain well bounded over the longer traversal duration. . . . . 39

2.23 Enlarged view of estimation errors for the repeated-turn trajectory, showing ICP, EKF, and PF without encoder-only odometry. . . . . 39

2.24 Enlarged view of estimation errors for the four-lap repeated-loop stress test, showing ICP, EKF, and PF without encoder-only odometry. Dashed vertical lines indicate lap boundaries. . . . . 40

3.1 Reliability-aware localization concept used in this chapter. Finite but unreliable ICP measurements are down-weighted within the EKF, fuzzy trust controls controller-level feedback blending, and sustained high-quality ICP permits gradual encoder anchoring within FUZZY-BLEND. . . . . 48

3.2 ICP registration RMSE over time. Dashed vertical lines mark the interval during which a temporary obstacle is present in the live map but absent from the ICP reference map. The mismatch increases registration inconsistency, while the comparison with the pose-error results shows why RMSE should not be interpreted as a complete correctness test by itself. . . . . 51

3.3 Innovation metrics relative to the encoder-based prediction: (top) position innovation $\nu_k^{\text{pos}}$, and (bottom) yaw innovation $\nu_k^{\theta}$. Increased innovation indicates stronger disagreement between the ICP measurement and the predicted state and is used as an indicator of reduced measurement consistency. 52

3.4 Input and output membership functions used in the Mamdani FIS. The three input variables use overlapping Small, Medium, and Large sets, while the output trust variable uses Low, Medium, and High sets. . . . . 54

3.5 Filtered fuzzy trust score $\bar{t}_k$. The score decreases when the ICP reliability indicators become less consistent and recovers smoothly when scan-to-map consistency returns. . . . . 56

3.6 Example Mamdani FIS trust surface with yaw innovation fixed at approximately 0.650 rad. The maximum defuzzified value is approximately 0.85, rather than 1.0, because centroid defuzzification is applied to the High output membership function $[0.60, 0.85, 1, 1]$. . . . . 56

3.7 Encoder–ICP disagreement within FUZZY-BLEND with and without encoder anchoring. During unreliable ICP, anchoring is disabled. When reliable scan-to-map registration persists long enough to satisfy the sustained-good gate, the anchoring mechanism gradually reduces the encoder–ICP position and yaw disagreement. . . . . 59

3.8 Diagnostics for the sustained-good ICP gate. (Top) ICP RMSE compared with the good-RMSE threshold. (Bottom) consecutive-good counter $c_k$ compared with $N_{\text{good}}$. A sample contributes to the counter only when the ICP result is finite and the RMSE, position innovation, and yaw innovation satisfy their respective limits. . . . . 61

3.9 Ground-truth robot trajectories under ENC, ICP, EKF, and FUZZY-BLEND localization feedback. The standalone ENC run is a raw encoder-only baseline and is not ICP-anchored. The zoomed view highlights the region affected by the temporary scan-to-map mismatch. . . . . 67

3.10 ICP reliability diagnostics from the EKF run. (Top) ICP registration RMSE. (Bottom) covariance-scaling factor $\alpha$, where $\mathbf{R}_{\text{eff}} = \alpha\mathbf{R}_{\text{base}}$. The vertical lines indicate the temporary mismatch interval. . . . . 68

3.11 Estimator error versus GT over the full run. (Top) position error, (middle) yaw error, and (bottom) FUZZY-BLEND filtered trust. Reduced trust shifts controller-level weight away from ICP and toward encoder feedback while the EKF independently applies its covariance-scaling logic. . . . . 69

3.12 Encoder anchoring diagnostics within FUZZY-BLEND. From top to bottom: encoder–ICP position gap (m), encoder–ICP yaw gap (rad), applied encoder position correction per step (m), and applied encoder yaw correction per step (rad). Position and angular quantities are shown separately to avoid mixing units. . . . . 70

4.1 AGV5 robot model in Gazebo, showing the differential-drive base, passive caster wheels, and front-mounted planar LiDAR sensor. . . . . 78

4.2 High-level ROS 2 localization architecture. Gazebo supplies wheel odometry and planar LiDAR returns; RTAB-Map provides local ICP odometry and diagnostics; the EKF provides a filtered estimate; and the custom monitor publishes a diagnostic ICP trust score. AMCL operates separately against the saved SLAM Toolbox map. . . . . 79

4.3 Gazebo warehouse environment containing shelves, open regions, and corridor-like structures used in the ROS 2 localization experiments. . . . . 82

4.4 Local ICP odometry and diagnostic flow. RTAB-Map processes `/scan`, publishes `/icp_odom` and `/odom_info`, and the custom monitor converts selected diagnostic fields into `/icp_trust`. . . . . 86

4.5 AMCL localization in RViz, showing the saved occupancy grid, live LiDAR scan, robot model, and particle-based pose estimate. . . . . 87

4.6 ROS 2 topic flow for wheel/ICP EKF estimation. The diagnostic trust output is logged separately and does not alter the EKF measurement weighting. 89

4.7 Representative ROS 2/Gazebo localization trajectories post-processed and plotted in MATLAB. When methods originate from different operating modes, the overlay is interpreted as a qualitative cross-mode comparison. . . . . 91

# List of Tables


2.1 Conventional EKF versus adaptive EKF under temporary map mismatch. . 31

2.2 Baseline tracking RMSE (GT vs Desired) for each estimator-driven closed-loop run. . . . . 32

2.3 Baseline estimation RMSE (Estimator vs GT). . . . . 32

2.4 Summary of additional open-path trajectory results. . . . . 41

2.5 Overall repeated-loop stress-test results over four laps. . . . . 41

2.6 Lap-by-lap repeated-loop stress-test results for encoder-only localization and EKF fusion. . . . . 42

3.1 FIS breakpoint parameters used to define the input membership functions. 53

3.2 Complete Mamdani FIS rule base implemented in MATLAB. . . . . 55

3.3 Overall estimator accuracy over the full trajectory. Each estimate is evaluated against the GT pose from its corresponding closed-loop run. . . . . 71

3.4 Position RMSE before, during, and after the temporary map-mismatch interval. . . . . 71

4.1 Summary of ROS 2 operational modes used in the simulation. . . . . 77

4.2 Major ROS 2 topics and their roles in the Gazebo implementation. . . . . 80

4.3 Main packages and their roles in the localization framework. . . . . 81

4.4 Principal launch and configuration files in the ROS 2 workspace. . . . . 81

4.5 Localization metrics from the representative ROS 2/Gazebo evaluations. Wheel odometry, ICP, and EKF were evaluated simultaneously, while AMCL was evaluated separately under the same nominal trajectory and initial condition. The final column reports the one-second translational displacement-error RMSE. . . . . 93

# Chapter 1

# Background and Literature

## 1.1 Introduction

Accurate self-localization is a fundamental requirement for autonomous mobile robots. A robot can only follow a path, avoid obstacles, or complete delivery tasks if it can estimate its position and orientation with sufficient accuracy. In indoor environments such as warehouses, this problem is particularly challenging because global positioning signals are typically unavailable, while onboard sensors are affected by noise, drift, occlusion, and environmental change. As a result, practical mobile-robot localization usually relies on a combination of proprioceptive sensing, such as wheel encoders, and exteroceptive sensing, such as LiDAR or vision [11, 63, 76].

Among indoor mobile robots, warehouse platforms present a particularly relevant application domain. These robots operate in structured but not perfectly static environments, often characterized by long aisles, repeated geometry, temporary obstacles, and changing layouts. In such settings, wheel odometry provides smooth short-term motion estimates but accumulates drift over time, while environment-based localization methods can correct that drift but may become unreliable when sensor observations degrade or the live environment differs from the reference map. For this reason, localization systems for warehouse robots commonly rely on probabilistic fusion methods that combine the continuity of odometry with the corrective value of external sensing [11, 63, 52].

This thesis focuses on that practical problem. In particular, it investigates localization for a differential-drive warehouse robot using wheel encoders and LiDAR-based estimation, with emphasis on two recurring failure modes: long-term odometric drift and reduced trustworthiness of LiDAR-based corrections under map mismatch or degraded scan matching.

Accordingly, this chapter does not attempt to provide an exhaustive review of all localization and SLAM methodologies. Instead, it presents a focused background on the concepts most relevant to the thesis: odometry, probabilistic localization, LiDAR-based correction, and multi-sensor fusion. Broader frameworks such as SLAM, AMCL, and graph-based optimization are introduced only briefly to position the present work within the wider robotics literature.

The remainder of this chapter is organized as follows. First, the role of localization in mobile robotics is introduced in the context of warehouse automation. Next, odometry-based localization and probabilistic state estimation are reviewed. This is followed by a focused discussion of LiDAR-based localization and the role of fusion methods such as the Extended Kalman Filter (EKF) and particle filters. Finally, the chapter summarizes the limitations of current approaches in structured indoor environments and identifies the research gap addressed in the subsequent chapters.

## 1.2 Mobile Robot Localization in Warehouse Systems

Mobile robots have evolved from simple guided platforms into autonomous systems capable of navigating warehouses, factories, and other indoor facilities with limited human intervention. In warehouse environments, autonomous mobile robots (AMRs) and automated guided vehicles (AGVs) are now widely used for material transport, inventory movement, and coordination between workstations. Although these systems differ in sensing and navigation sophistication, they share a common requirement: reliable localization.

Historically, many industrial AGVs relied on highly structured guidance methods such as wires, magnetic strips, or predefined tracks. These approaches reduced the localization burden because the robots' motion was physically constrained. Modern warehouse robots, by contrast, are expected to move more flexibly, re-route around obstacles, and operate in larger shared spaces. This shift has made onboard localization and navigation much more important [70, 76].

Despite advances in autonomy, warehouse localization remains difficult for several practical reasons. First, indoor environments are generally GNSS-denied, so robots cannot depend on satellite-based positioning. Second, wheel slip, encoder quantization, and calibration errors cause dead-reckoning methods to drift over time. Third, warehouse layouts often contain repeated structures such as parallel aisles and similar shelving, which can make environment-based localization ambiguous. Finally, temporary changes in the scene, such as moved pallets or nearby obstacles, can degrade scan matching or map-based cor-

rections. These challenges motivate localization systems that combine multiple sensing sources and explicitly reason about changing measurement reliability.

For these reasons, warehouse robots often use a layered localization strategy. Wheel odometry provides continuous local motion information, while exteroceptive sensors such as LiDAR provide geometric observations of the environment that can be used to constrain drift. A fusion framework then combines these information sources to produce a smoother and more reliable pose estimate than either source alone. This principle forms the core technical motivation of the present thesis.

## 1.3 Odometry-Based Localization

Odometry-based localization estimates the motion of a robot by integrating incremental measurements over time. For ground robots, these measurements commonly come from wheel encoders and, in some systems, inertial sensors. Odometry is attractive because it is simple, computationally efficient, and available at high update rates. For this reason, it often serves as the backbone of real-time motion estimation.

For a differential-drive robot, the planar pose is commonly represented as

$$\mathbf{p} = \begin{bmatrix} x & y & \theta \end{bmatrix}^T, \tag{1.1}$$

where $(x, y)$ denotes the robot position and $\theta$ denotes the heading angle. Using the incremental displacements of the right and left wheels, the motion over one sampling interval can be approximated by

$$\Delta\mathbf{p} = \begin{bmatrix} \frac{\Delta s_r + \Delta s_l}{2} \cos\theta \\ \frac{\Delta s_r + \Delta s_l}{2} \sin\theta \\ \frac{\Delta s_r - \Delta s_l}{b} \end{bmatrix}, \tag{1.2}$$

where $\Delta s_r$ and $\Delta s_l$ are the right and left wheel displacements and $b$ is the wheel baseline [70].

This formulation is straightforward and useful, but it has a major limitation: errors accumulate over time. Small inaccuracies in wheel radius, encoder measurement, wheel baseline, or floor contact lead to growing pose errors as the robot travels. These errors are often classified as systematic errors, such as unequal wheel diameters or uncertain geometry, and non-systematic errors, such as slip or floor irregularity [9, 70]. Since odometry is obtained by integration, even small errors can eventually lead to large deviations between the estimated pose and the true pose.

Because of this, wheel odometry alone is usually insufficient for long-term localization. Still, it remains valuable because it provides smooth local motion information and short-term continuity. In practice, odometry is often treated as a prediction source that must be corrected by environment-based sensing. This exact role is central to this thesis, where encoder odometry provides the local motion estimate but must be corrected when drift grows.

In some systems, odometry is combined with inertial sensing to improve short-term motion estimation. Such fusion can reduce the effect of transient wheel slip and improve heading consistency. However, even enhanced odometry remains a dead-reckoning process and therefore cannot guarantee long-term global accuracy without external correction [52, 12].

## 1.4 Probabilistic Localization and Bayesian Estimation

A key development in robotics localization was the transition from deterministic pose tracking to probabilistic state estimation. Instead of treating the robot pose as a single exact value, probabilistic localization represents it as a belief that incorporates uncertainty. This is important because robot motion and sensor observations are inherently noisy.

The general Bayesian localization update can be written as

$$bel(\mathbf{x}_t) = \eta\, p(\mathbf{z}_t|\mathbf{x}_t) \int p(\mathbf{x}_t|\mathbf{u}_t, \mathbf{x}_{t-1})\, bel(\mathbf{x}_{t-1})\, d\mathbf{x}_{t-1}, \tag{1.3}$$

where $\mathbf{x}_t$ is the state, $\mathbf{u}_t$ is the motion input, $\mathbf{z}_t$ is the measurement, and $\eta$ is a normalization constant [75]. This equation expresses the two main steps used in recursive localization: prediction and correction. First, the previous belief is propagated using the motion model. Then, the estimate is corrected using new sensor information.

Several well-known localization methods follow this principle. The Kalman Filter (KF) provides an optimal solution for linear Gaussian systems. Since mobile-robot motion and measurement models are generally nonlinear, the Extended Kalman Filter (EKF) is often used instead, where nonlinear models are linearized around the current estimate. The Unscented Kalman Filter (UKF) improves nonlinear propagation by using deterministic sampling. Another major class is the Particle Filter (PF), also known as Monte Carlo Localization, which represents the belief using a set of weighted particles and is well suited for multi-modal uncertainty and global re-localization [75, 20, 23].

These methods differ in computational cost, representation power, and sensitivity to modeling assumptions. EKF-based methods are efficient and widely used in real-time systems, especially when the state uncertainty can be reasonably approximated as Gaussian. Particle filters are more flexible and can recover from larger pose ambiguity, but they are typically more computationally expensive. In this thesis, both EKF-based fusion and particle filtering are considered in simulation, not as equally emphasized final solutions, but as useful reference points for comparing different localization behaviors under warehouse conditions.

The probabilistic view is especially important for this work because it provides the framework for reasoning about sensor reliability. When odometry and LiDAR do not agree, the estimator must decide how much to trust each source. This is naturally expressed through covariance weighting, innovation analysis, and related trust measures.

## 1.5 LiDAR-Based Localization and Scan Matching

LiDAR is widely used in indoor mobile robotics because it provides direct geometric information about the surrounding environment. In structured settings such as warehouses, LiDAR scans can capture walls, shelves, and corridor boundaries with relatively high precision. This makes LiDAR particularly useful for correcting the long-term drift of odometry.

One common approach is scan matching, in which a newly observed scan is aligned with either a previous scan or a reference map. A widely used technique for this purpose is the Iterative Closest Point (ICP) algorithm. ICP estimates the rigid transformation that best aligns two point sets by iteratively associating points and minimizing an alignment error. When the initial pose guess is reasonable and the environment contains enough geometric structure, ICP can provide useful pose corrections. However, ICP is also sensitive to local minima, ambiguous geometry, poor initialization, and map mismatch.

These limitations are important in warehouse environments. Repeated aisles and similar shelf structures can reduce geometric uniqueness. Temporary obstacles or environmental changes can produce scans that no longer match the stored reference well. Under such conditions, the LiDAR-based estimate may become unreliable even though it still produces an output. For this reason, LiDAR correction should not always be accepted blindly. Instead, its reliability should be evaluated before it is heavily trusted by the fusion framework.

This issue is central to the present thesis. The work does not treat LiDAR as a perfect source of truth. Rather, it studies how LiDAR-based localization can improve odometric drift when scan quality is good, while reducing its influence when scan matching becomes

questionable. The contribution is not the general idea of adaptive sensor weighting, which already exists in the localization literature, but the particular reliability-aware combination developed and evaluated later in this thesis for the warehouse localization problem.

## 1.6 Fusion of Odometry and LiDAR

Because odometry and LiDAR have complementary strengths and weaknesses, combining them is a natural strategy for indoor localization. Odometry provides smooth, high-rate local motion updates but drifts over time. LiDAR-based localization provides environment-referenced correction but may degrade when scan matching is poor. Fusion aims to obtain a better estimate than either source alone.

A common model-based fusion method is the Extended Kalman Filter (EKF). In this setting, odometry is typically used in the prediction step, while LiDAR-based pose information is used in the measurement update. The EKF recursively combines the two based on their assumed uncertainty. If the measurement covariance is well chosen, the filter can balance short-term continuity with external correction and produce a stable estimate.

However, this balance depends strongly on how the measurement uncertainty is modeled. In many practical systems, fixed noise covariances are used. This simplifies implementation, but it can be problematic when sensor reliability changes during operation. If LiDAR corrections become unreliable because of map mismatch or degraded scan alignment, a fixed-covariance filter may continue to trust them too much. This can lead to estimator degradation rather than improvement.

For that reason, recent research has increasingly emphasized adaptive or reliability-aware fusion methods, in which sensor influence is adjusted online according to quality indicators such as innovation size, matching score, or uncertainty estimates [52, 60]. In LiDAR-based localization, such indicators may include scan-matching residuals, ICP root-mean-square error (RMSE), innovation magnitude, or consistency between the predicted and corrected pose. When these indicators suggest that the LiDAR correction is consistent with the motion estimate, the fusion framework can assign it greater influence. Conversely, when scan matching becomes unreliable because of map mismatch, occlusion, or weak geometric structure, the measurement covariance can be increased so that the estimator depends more strongly on odometric continuity.

The present thesis follows this direction in a focused warehouse setting using wheel encoders and LiDAR, but it does not rely on a single adaptive mechanism. The reliability-aware method developed later combines three complementary elements: (i) ICP RMSE

and innovation checks that adapt the effective LiDAR measurement covariance within the EKF, (ii) a multi-input fuzzy trust score that controls controller-level blending among encoder, EKF, and ICP estimates, and (iii) a separate sustained-good condition that permits gradual encoder anchoring only after ICP has remained valid and consistent for multiple consecutive samples. This combination is evaluated under temporary scan-to-map mismatch and accumulated odometric drift.

Particle filters can also be used for LiDAR-based localization, particularly when global ambiguity or multi-modal uncertainty is important. In this thesis, particle filtering is considered as a comparative method in simulation. Nevertheless, the main emphasis is on LiDAR-encoder fusion through a probabilistic framework that remains computationally practical and easier to integrate into the overall warehouse robot pipeline.

## 1.7 Broader Localization Frameworks in Context

The wider robotics literature contains many localization and mapping frameworks beyond the direct scope of this thesis. Simultaneous Localization and Mapping (SLAM) addresses the problem of building a map while localizing within it. Classical SLAM approaches include EKF-SLAM, particle-filter-based methods such as FastSLAM, and graph-based optimization methods [11, 27, 53]. These methods are especially important in exploration or map-building settings where no prior map is available.

In practical ROS-based navigation systems, Adaptive Monte Carlo Localization (AMCL) is also widely used for localization on a known 2D occupancy map. In this thesis, SLAM-related methods and AMCL provide useful implementation and comparison context, particularly in the ROS 2/Gazebo portion of the work. The methodological focus remains on the narrower problem of combining encoder odometry with LiDAR-based localization under changing measurement reliability.

This scope is intentional. The thesis does not attempt to develop a complete SLAM system or a full multi-modal autonomy framework. Instead, it focuses on a practically important warehouse localization problem: preserving the short-term continuity of encoder odometry while using LiDAR-based corrections when they are reliable and reducing their influence when scan consistency deteriorates.

## 1.8 Challenges in Structured Indoor Environments

Although warehouse environments are more structured than many outdoor settings, localization in such spaces still presents important challenges. One challenge is the gradual accumulation of odometric drift, especially during long travel distances or repeated turning maneuvers. Another is the difficulty of maintaining reliable scan matching in environments with repeated geometry, partial occlusion, or temporary layout changes. These factors can reduce the quality of LiDAR-based corrections even when a map is available.

A further challenge is that practical sensor reliability is not constant. The confidence that should be assigned to a LiDAR correction may vary significantly from one time step to the next. Yet many fusion methods assume fixed noise characteristics. This creates a mismatch between theory and practice: the estimator may behave well under nominal conditions but less reliably when the environment becomes inconsistent with the assumptions embedded in the filter.

Structured indoor environments also introduce ambiguity. Long corridors, similar shelving rows, and sparse distinctive landmarks can make different locations appear geometrically similar. In such cases, localization methods that rely too strongly on one sensing cue may become brittle. This is why combining motion continuity with environment-based correction remains valuable, provided that the fusion mechanism can react appropriately when one source becomes less trustworthy.

These challenges motivate a localization framework that is not only probabilistic, but also reliability-aware. Such a framework should preserve the short-term continuity of encoder odometry, exploit LiDAR-based correction when it is trustworthy, and reduce dependence on LiDAR measurements when scan consistency deteriorates.

## 1.9 Summary and Research Gap

This chapter reviewed the background most relevant to the present thesis: odometry-based localization, probabilistic state estimation, LiDAR scan matching, and the fusion of internal and external sensing for indoor mobile robots. The literature shows a consistent trade-off. Wheel odometry is computationally efficient and locally smooth but accumulates drift. LiDAR-based localization can provide map-referenced correction, yet its reliability depends on geometric distinctiveness, scan quality, initialization, and consistency between the live environment and the reference map. Probabilistic fusion methods such as the EKF

provide an effective framework for combining these sources, but their behavior depends strongly on how measurement uncertainty is represented.

Existing adaptive and reliability-aware localization methods already address the general problem of time-varying sensor confidence by modifying covariance, rejecting measurements, or using consistency indicators. Therefore, the research gap addressed in this thesis is not simply that fixed sensor trust can fail. Instead, the gap considered here is the need for a practical warehouse-localization architecture that combines multiple reliability interventions at different levels of the localization and control pipeline and evaluates their complementary roles under controlled failure conditions.

Specifically, the methodology developed in this thesis combines three mechanisms:

1. **Adaptive EKF measurement weighting:** ICP registration RMSE and innovation consistency are used to adjust the effective ICP measurement covariance so that finite but questionable scan-matching updates have reduced influence on the EKF.

2. **Multi-input fuzzy feedback blending:** ICP RMSE, position innovation, and yaw innovation are combined into a continuous trust score that adjusts the controller-level contribution of encoder, EKF, and ICP estimates without abrupt source switching.

3. **Sustained-good encoder anchoring:** when ICP remains valid and consistent for a required number of consecutive samples, a separate conservative mechanism gradually reduces accumulated encoder offset by anchoring the encoder state toward the trusted map-referenced pose.

The value of this combination is evaluated in MATLAB under both temporary scan-to-map mismatch and long-duration odometric drift. The MATLAB study also compares encoder-only, ICP-based, EKF-based, and particle-filter localization within a common simulation framework so that the reliability-aware method can be interpreted relative to established alternatives.

The later ROS 2/Gazebo implementation serves a different but complementary purpose. It transfers the supporting localization architecture to a modular real-time robotics environment containing wheel odometry, local LiDAR odometry, EKF-based estimation, map-based localization, transform management, logging, visualization, and online ICP diagnostics. The complete Chapter 3 fuzzy blending and reliability-dependent covariance adaptation are not reproduced in the ROS 2 fusion loop in the present work. The ROS 2 stage should therefore be interpreted as a system-level implementation and integration

foundation for future deployment of the complete reliability-aware method, rather than as a direct real-time replication of all MATLAB mechanisms.

Accordingly, the thesis addresses a focused problem rather than the full scope of SLAM or multi-modal autonomy: how to preserve the short-term continuity of encoder odometry, reduce the influence of unreliable LiDAR corrections, and make use of sustained reliable scan matching to limit long-term encoder drift in structured indoor warehouse environments. The following chapters develop, evaluate, and implement the components of this reliability-aware localization framework.

# Chapter 2

# System Modeling and Simulation in MATLAB

## 2.1 Introduction

Accurate localization is a fundamental requirement for autonomous mobile robots operating in structured indoor environments such as warehouses, factories, and logistics hubs. A robot that cannot estimate its position and orientation with sufficient precision cannot reliably follow its assigned path, avoid collisions, or coordinate with other systems in shared spaces.

This thesis focuses on a warehouse-style mobile robot with a **differential-drive** base. Differential drive is chosen because it is mechanically simple, widely deployed in industrial AGVs/AMRs, and well suited to structured warehouse navigation with corridors, aisles, and frequent 90° turns. Compared to omnidirectional platforms, it reduces mechanical and control complexity; compared to Ackermann steering, it supports tight in-place turns and low-speed maneuvers common in indoor logistics.

The most direct way to estimate motion is through *dead reckoning*, which integrates incremental displacements derived from wheel encoder measurements. Encoder-based odometry is computationally inexpensive, but its accuracy deteriorates over time due to wheel slip, floor irregularities, encoder quantization, calibration uncertainty, and accumulated heading error.

To mitigate drift, mobile robots employ *exteroceptive sensors* such as LiDAR and cameras, which observe the environment and provide geometric constraints. In this work, a 2-D

LiDAR is emphasized because it is less sensitive to lighting changes and can provide dense geometric information. In real warehouses, LiDAR scans may also contain *ghost reflections* or *mixed pixels* at corners, glass, or specular surfaces. While the present simulation models LiDAR uncertainty primarily as Gaussian range noise, these real-world effects further motivate the use of robust estimation and fusion rather than relying solely on idealized dead reckoning.

MATLAB was selected as the prototyping environment because it supports rapid iteration and provides strong tool support for robotics mapping, scan processing, and estimation workflows. The algorithms implemented here follow standard robotics foundations, including scan matching and probabilistic localization, with emphasis placed on repeatable evaluation and practical diagnostics.

In this chapter, multiple localization strategies are implemented and evaluated within a unified MATLAB simulation environment:

1. **Encoder-Based Odometry (Dead Reckoning):** pose estimation derived purely from wheel encoder integration.
2. **LiDAR Scan-to-Map Localization (ICP):** pose estimation from scan registration using Iterative Closest Point (ICP).
3. **Extended Kalman Filter (EKF) Fusion (Encoder + ICP):** encoder-based prediction corrected by ICP pose measurements with adaptive measurement covariance.
4. **Particle Filter (PF) Localization (Monte Carlo Localization):** sampling-based localization using an encoder motion model and a LiDAR measurement likelihood against the map.

Figure 2.1 summarizes the evaluation structure. The robot follows a reference trajectory in a simulated warehouse map. Each algorithm produces an estimated pose, and performance is quantified using time-history errors and trajectory-level statistics.

The objectives of this chapter are:

- To implement and validate multiple localization methods in a consistent simulation environment.
- To evaluate accuracy in both *tracking performance* (ground truth (GT) vs desired trajectory) and *estimation performance* (estimator vs ground truth (GT)).

- To discuss trade-offs between encoder-only odometry, scan matching, probabilistic localization, and model-based fusion.

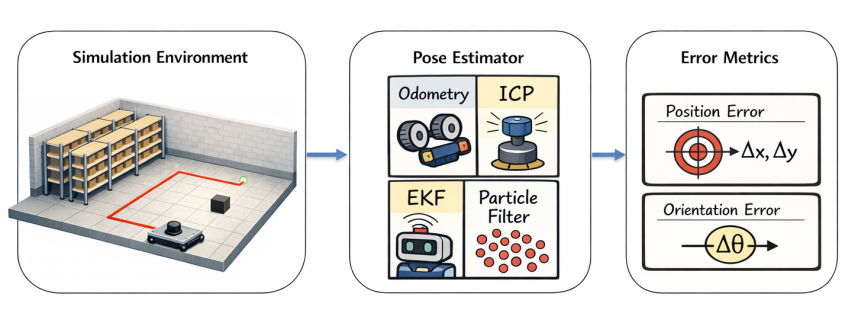



Figure 2.1: General structure of the localization and evaluation framework. A simulated robot follows a predefined trajectory in a warehouse map while multiple estimators produce pose estimates. Each estimator is evaluated using common error metrics.

The remainder of this chapter is organized as follows. Section 2.2 describes the simulation environment, robot, sensors, and trajectory design. Section 2.3 presents encoder-based odometry and drift behavior. Section 2.4 introduces scan-to-map ICP localization. Section 2.5 presents particle filter localization and diagnostics. Section 2.6 details EKF fusion using encoder prediction and ICP correction. Section 2.7 summarizes evaluation metrics and experimental protocol. Finally, Sections 2.8–2.9 discuss runtime considerations, limitations, and provide a chapter summary.

## 2.2 Simulation Environment and Experimental Setup

A controlled MATLAB simulation framework was developed to compare localization methods under consistent conditions. The setup models a differential-drive autonomous mobile robot operating inside a structured warehouse environment, equipped with wheel encoders and a simulated 2-D LiDAR. Repeatability is ensured by fixing the reference trajectory and controlling the pseudorandom seed for stochastic components.

### 2.2.1 Warehouse Map and Waypoints

The warehouse layout is represented as a **binary occupancy grid** with a resolution of 10 cells/m (cell size 0.10 m), as shown in Figure 2.2. Obstacles such as racks and walls are

encoded as occupied cells, while free cells represent navigable space. The map dimensions are approximately $16 \times 18$ m. The map is hand-constructed to provide a warehouse-style environment containing walls, aisles, rack-like obstacles, and corridor structures suitable for controlled and repeatable localization experiments.

In the baseline case, waypoints were selected to create a warehouse-style navigation task consisting of an L-shaped path that includes a straight segment and a $90^\circ$ turn. To reduce dependence on a single path geometry, two additional warehouse-style trajectories were later evaluated using the same nominal simulation configuration: a corridor-dominant path emphasizing extended straight-line motion and a repeated-turn path emphasizing multiple heading changes.

A separate repeated-loop stress test was conducted by traversing a closed rectangular trajectory over multiple laps without resetting the estimator state. This test intentionally differed from the nominal open-path cases in the encoder error model. In addition to stochastic wheel-speed noise, a small persistent left–right wheel-scale asymmetry was applied using scale factors of 0.99 and 1.01, respectively. The purpose was to introduce a controlled systematic odometric bias and expose long-duration drift accumulation over repeated laps. The repeated-loop experiment is therefore interpreted as a stress test rather than as an identical-condition comparison with the baseline, corridor, and repeated-turn trajectories.

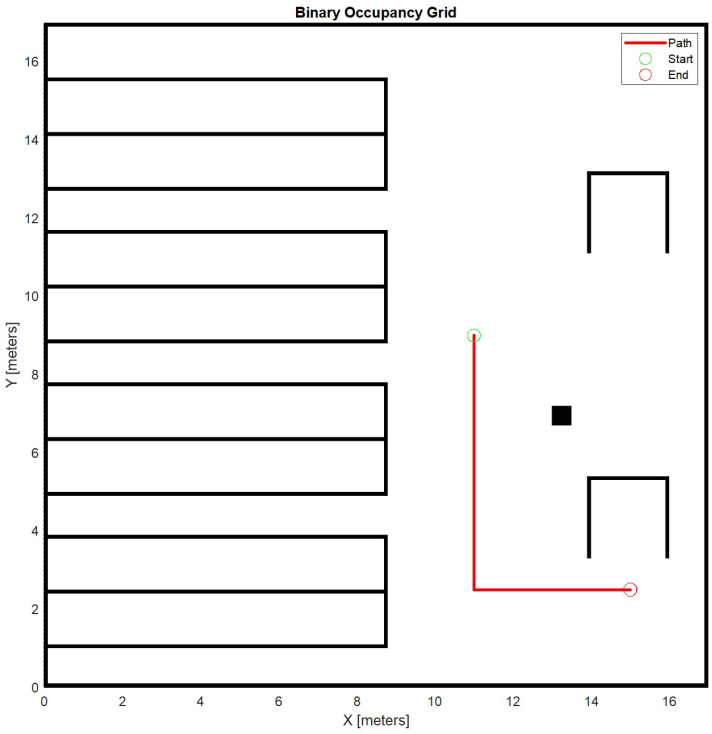



Figure 2.2: Simulated warehouse environment represented as a binary occupancy grid with the baseline reference trajectory.

### 2.2.2 Robot Model and Sensors

The robot follows a differential-drive kinematic model. The robot pose in the global map frame is

$$\mathbf{x} = [x,\, y,\, \theta]^T,$$

where $(x, y)$ is position and $\theta$ is heading. Control inputs are linear velocity $v$ and angular velocity $\omega$. For wheel radius $r$ and wheelbase $b$, the relationship to wheel angular speeds $\omega_L$ and $\omega_R$ is

$$v = \frac{r}{2}(\omega_R + \omega_L), \qquad \omega = \frac{r}{b}(\omega_R - \omega_L).$$

The wheel radius and wheelbase used in the MATLAB simulation were set to $r = 0.10$ m and $b = 0.50$ m, respectively. The detailed ROS 2/Gazebo model used the same wheel radius and a wheel separation of 0.46 m. This small difference reflects the simplified MATLAB model and does not affect comparisons performed within each implementation.

The robot is equipped with:

- **Wheel encoders:** used to generate incremental motion inputs and encoder-based odometry. In the baseline, corridor, and repeated-turn experiments, drift is induced using zero-mean measurement noise on the wheel-speed signals and no explicit slip event is injected. The repeated-loop stress test additionally introduces a small persistent left–right wheel-scale asymmetry (0.99 and 1.01) to represent a controlled systematic odometric bias.
- **2-D LiDAR:** simulated with a 360° field of view, angular resolution 0.4°, and maximum range 30 m. Range noise is modeled as zero-mean Gaussian with a standard deviation of 0.10 m, while angular noise has a standard deviation of 0.2°. Although Gaussian noise is used for repeatability, real warehouse scans may also exhibit corner artifacts and occasional ghost returns, which further motivates robust estimation and fusion.

All sensors are simulated at a fixed sampling time of $\Delta t = 0.1$ s (10 Hz) to maintain consistent time alignment across all algorithms.

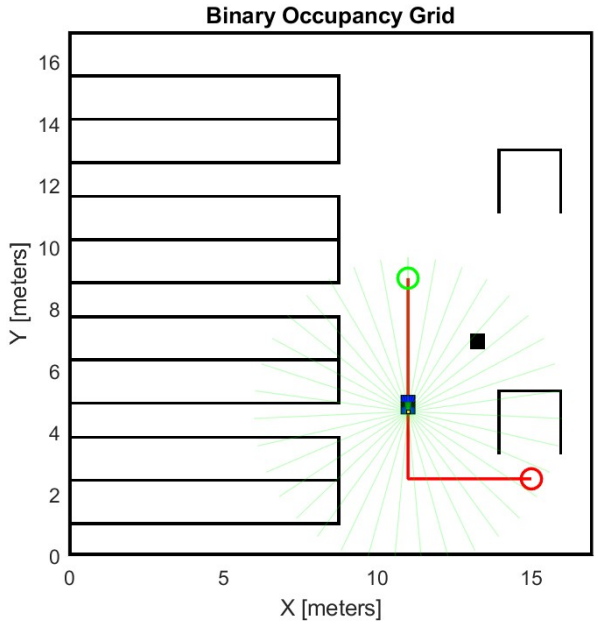



Figure 2.3: Visualization of the simulated robot and LiDAR rays in the warehouse map.

## 2.3 Encoder-Based Odometry

Encoder-based odometry estimates robot motion by integrating wheel encoder increments over time. It is computationally efficient and provides smooth short-term pose estimates, but it suffers from drift because no external reference is used.

### 2.3.1 Kinematic Formulation

For a differential-drive robot, the continuous-time unicycle model is

$$\dot{x} = v\cos\theta, \tag{2.1}$$
$$\dot{y} = v\sin\theta, \tag{2.2}$$
$$\dot{\theta} = \omega. \tag{2.3}$$

Discretizing with sampling time $\Delta t$, and using encoder-derived velocity estimates $(v_k, \omega_k)$, gives

$$x_{k+1} = x_k + v_k \Delta t \cos\theta_k, \tag{2.4}$$
$$y_{k+1} = y_k + v_k \Delta t \sin\theta_k, \tag{2.5}$$
$$\theta_{k+1} = \mathrm{wrap}(\theta_k + \omega_k \Delta t). \tag{2.6}$$

In simulation, encoder uncertainty is introduced by perturbing the wheel speed measurements, which generates noisy estimates of $v_k$ and $\omega_k$ for odometry integration. Given the small sampling interval used in this study ($\Delta t = 0.1$ s), first-order Euler integration was considered sufficient for kinematic propagation.

### 2.3.2 Noise Model and Drift Accumulation

In practice, encoder-odometry errors may arise from wheel slip, calibration uncertainty, quantization, and timing mismatch. In the present simulation, encoder uncertainty is modeled by adding independent zero-mean Gaussian noise directly to the left and right wheel angular-speed measurements:

$$\omega_{L,k}^{\mathrm{meas}} = \omega_{L,k} + \eta_{L,k}, \tag{2.7}$$
$$\omega_{R,k}^{\mathrm{meas}} = \omega_{R,k} + \eta_{R,k}, \tag{2.8}$$

where

$$\eta_{L,k}, \eta_{R,k} \sim \mathcal{N}\left(0, \sigma_\omega^2\right), \qquad \sigma_\omega = 0.10 \text{ rad/s}.$$

The noisy wheel-speed measurements are converted into measured linear and angular velocities according to

$$v_k^{\text{meas}} = \frac{r}{2}\left(\omega_{R,k}^{\text{meas}} + \omega_{L,k}^{\text{meas}}\right), \tag{2.9}$$

$$\omega_k^{\text{meas}} = \frac{r}{b}\left(\omega_{R,k}^{\text{meas}} - \omega_{L,k}^{\text{meas}}\right). \tag{2.10}$$

No explicit wheel-slip event is injected in the baseline, corridor, or repeated-turn simulations. In those nominal cases, the encoder error model is therefore zero-mean at the wheel-speed measurement level. The repeated-loop stress test is intentionally different: the left and right wheel signals are additionally scaled by 0.99 and 1.01, respectively, before odometric propagation. This persistent asymmetry produces a systematic linear/angular motion bias that accumulates from lap to lap. Because odometry integrates both stochastic and systematic motion errors without an external reference, position and especially heading error can grow substantially over long repeated traversals.

### 2.3.3 Performance and Limitations

Figure 2.4 compares encoder-only odometry against ground truth along the baseline warehouse trajectory. Although the estimate remains smooth, small heading and velocity errors accumulate over time, leading to a gradual deviation from the ground-truth path that becomes more visible after the turn.

To quantify this drift, Figure 2.5 shows the Absolute Position Error (APE) over time for encoder-only odometry. The error grows progressively throughout the run, reaching its largest values near the end of the trajectory. This behavior is consistent with the cumulative nature of dead reckoning, where small motion-estimation errors are integrated over time and cannot self-correct without an external reference.

Encoder odometry remains attractive because of its simplicity and low computational cost, but by itself it is not sufficient for long-term localization accuracy. In practice, it benefits from periodic correction using exteroceptive sensors or sensor-fusion frameworks.

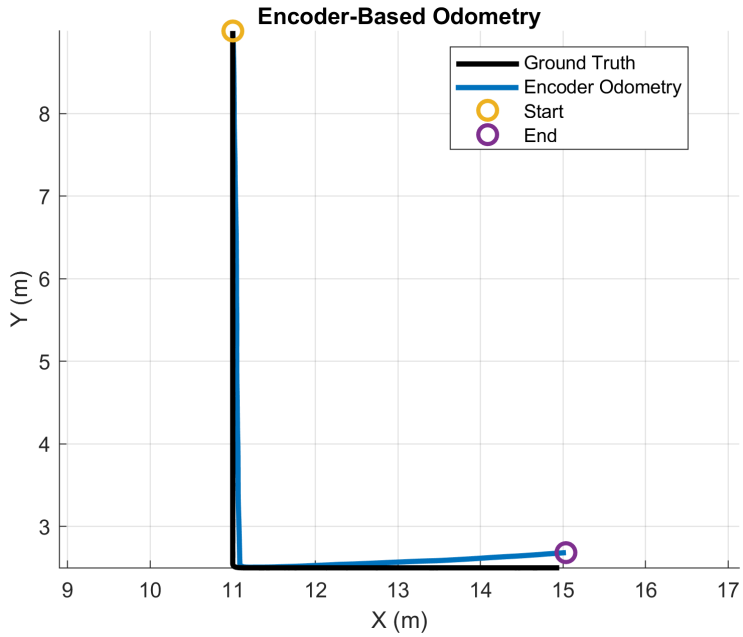



Figure 2.4: Encoder-based odometry compared with the ground-truth trajectory. Small dead-reckoning errors accumulate over time and become more visible after the turn.

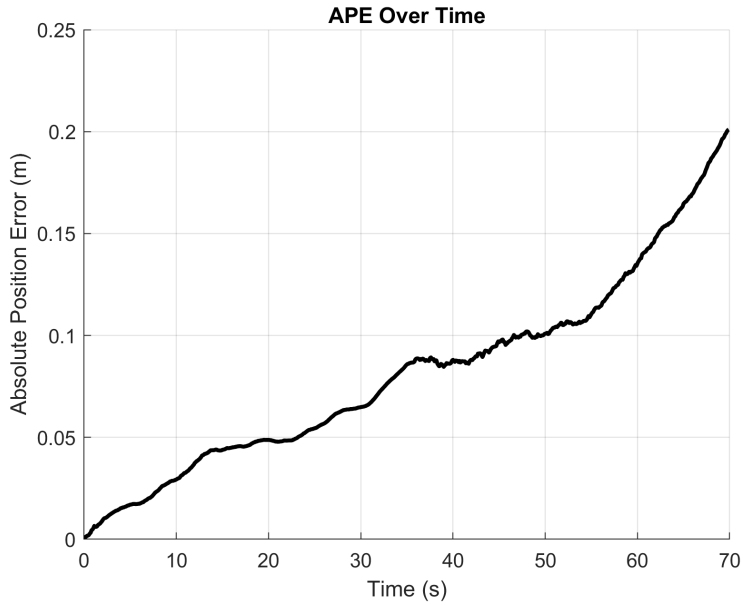



Figure 2.5: Absolute Position Error (APE) over time for encoder-only odometry.

## 2.4 LiDAR-Based Localization (ICP Scan-to-Map)

LiDAR-based localization reduces drift by aligning the current LiDAR scan against a reference map. In this work, ICP performs scan-to-map registration between the current scan point set and a dense point set extracted from occupied cells of the occupancy grid. In this chapter, the term *live scan* refers to the LiDAR observation generated from the robot's current simulated environment, while the *reference map* refers to the occupancy-grid map used as the fixed scan-matching target.

### 2.4.1 Concept and Cost Function

Given a scan point set $\mathcal{Q} = \{\mathbf{q}_i\}$ and map point set $\mathcal{P} = \{\mathbf{p}_j\}$, ICP solves the point-to-point objective

$$\min_{\mathbf{R},\mathbf{t}} \sum_{i \in \mathcal{I}} \|\mathbf{p}_i - (\mathbf{R}\mathbf{q}_i + \mathbf{t})\|^2, \tag{2.11}$$

where $\mathbf{R} \in \mathrm{SO}(2)$ and $\mathbf{t} \in \mathbb{R}^2$, and $\mathcal{I}$ denotes correspondences typically chosen via nearest neighbors.

### 2.4.2 Implementation Notes

In the MATLAB implementation, scan-to-map registration is performed using `pcregistericp`. The scan is converted to Cartesian points and registered to the map point set using the point-to-point metric, consistent with the ICP objective in Equation 2.11. ICP is warm-started using the previous ICP pose estimate as the initial guess. This is a critical practical detail: ICP has a finite basin of convergence, and if the robot moves too far between consecutive frames, the initial guess may fall outside this basin and ICP may converge to an incorrect local minimum.

### 2.4.3 Results and Discussion

In the tested MATLAB scenarios, ICP substantially reduced drift relative to encoder-only odometry, particularly when the environment provided sufficiently distinctive geometric features. Figure 2.6 illustrates the scan-to-map alignment outcome. Although ICP can be highly accurate when geometry is sufficiently distinctive, its reliability depends on scan quality, environmental structure, and initialization.

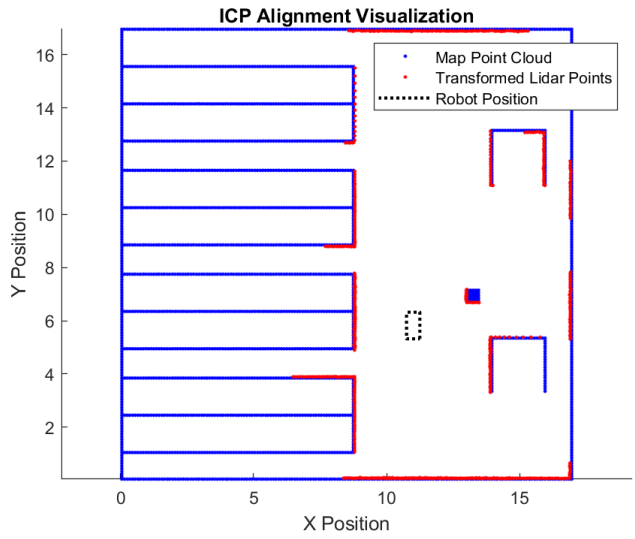



Figure 2.6: ICP scan-to-map alignment visualization on the warehouse map.

Figure 2.7 reports the per-step registration RMSE. This RMSE is a registration residual after convergence, not the global pose error. In the tested baseline simulation, the observed RMSE remains close to 0.10 m on average. Transient increases correspond to harder registration frames, such as reduced overlap, partial observability, or less-informative geometry.

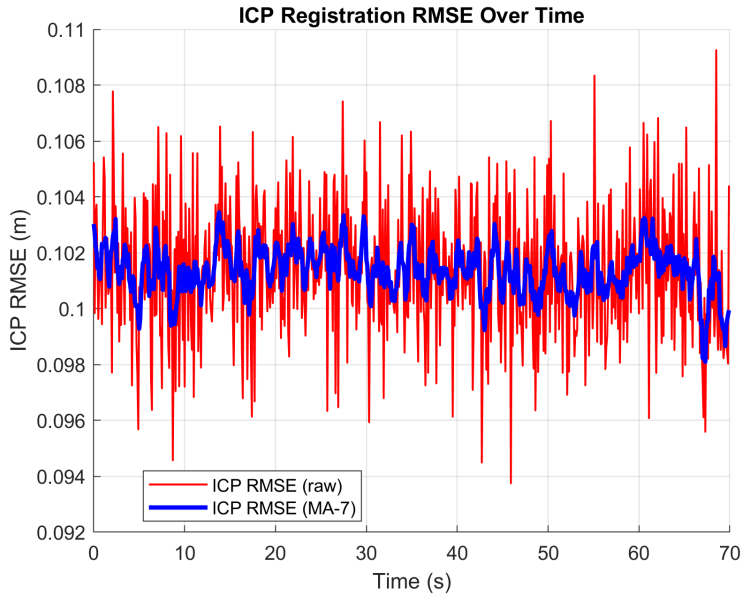



Figure 2.7: Registration RMSE over time. Transient increases indicate harder frames, such as poor overlap or ambiguous geometry.

These considerations motivate fusion approaches that combine model-based prediction from encoders with ICP corrections, as discussed in Section 2.6.

## 2.5 Particle Filter-Based Localization

Encoder odometry provides smooth short-term pose estimates but accumulates drift, while scan matching may become less reliable in feature-poor regions. These limitations motivate a probabilistic localization framework. The particle filter represents the robot belief as a set of weighted pose hypotheses that are propagated by a motion model and corrected by LiDAR measurements.

### 2.5.1 Bayesian Formulation

At time step $k$, the posterior over pose $\mathbf{x}_k = [x,\, y,\, \theta]^T$ is

$$p(\mathbf{x}_k \mid \mathbf{z}_{1:k}, \mathbf{u}_{1:k}),$$

where $\mathbf{u}_{1:k}$ are encoder-derived inputs and $\mathbf{z}_{1:k}$ are LiDAR measurements. Using the recursive Bayes filter,

$$p(\mathbf{x}_k \mid \mathbf{z}_{1:k}, \mathbf{u}_{1:k}) \propto p(\mathbf{z}_k \mid \mathbf{x}_k) \int p(\mathbf{x}_k \mid \mathbf{u}_k, \mathbf{x}_{k-1})\, p(\mathbf{x}_{k-1} \mid \mathbf{z}_{1:k-1}, \mathbf{u}_{1:k-1})\, d\mathbf{x}_{k-1}. \tag{2.12}$$

### 2.5.2 Motion and Measurement Models

**Motion model.** Particles are propagated using differential-drive kinematics with additive Gaussian noise.

**Measurement likelihood.** A scan likelihood is computed by comparing observed ranges with expected ranges simulated from the map. The particle-filter likelihood parameter was set to $\sigma_z = 0.10$ m. This parameter controls the width of the particle-filter measurement likelihood and is conceptually distinct from the simulated LiDAR range-noise standard deviation. In the experiments reported here, both happen to have the same numerical value of 0.10 m; however, one is an estimator-tuning parameter and the other specifies the noise applied to the simulated LiDAR measurements.

To reduce computation, beam subsampling is used. With a 0.4° angular resolution over 360°, each scan contains approximately 900 beams. Using a stride of 3 evaluates approximately 300 beams per update.

In the baseline implementation, the particle filter uses $N = 250$ particles, and resampling is triggered when the effective sample size falls below $0.5N$.

### 2.5.3 Simulation Results and Diagnostics

Figure 2.8 shows that the particle filter maintains much lower position error than encoder-only odometry for most of the baseline run, indicating that LiDAR-based weighting helps correct long-term drift. Figure 2.9 provides internal diagnostics through the effective sample size, $N_{\text{eff}}$, and a weighted scan RMSE diagnostic.

### 2.5.4 Discussion

PF reduces drift by maintaining multiple pose hypotheses and re-weighting them using LiDAR map consistency. The main trade-off is computational cost, which scales approximately with the number of particles and the number of beams evaluated per update.

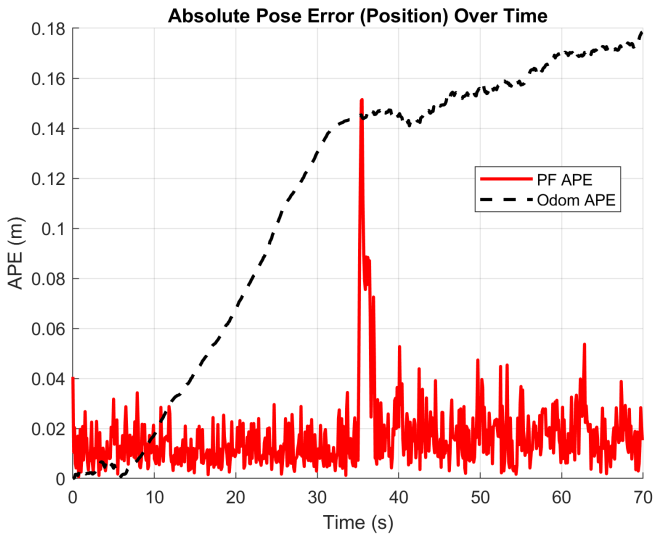



Figure 2.8: Absolute Position Error (APE) for the PF run compared with encoder dead reckoning using the same ground-truth trajectory.

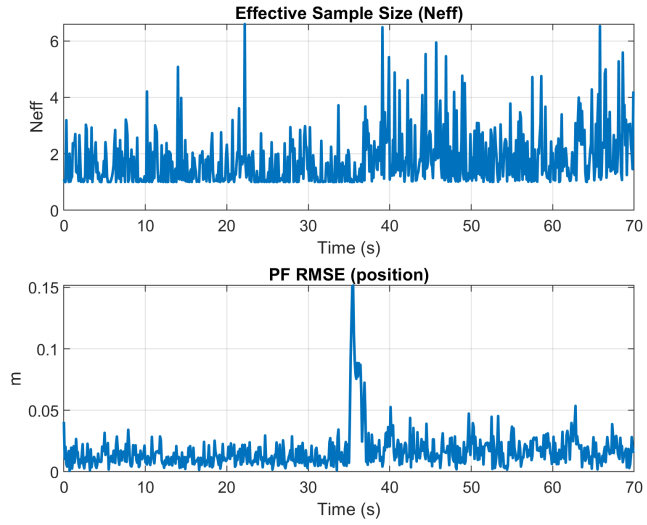



Figure 2.9: PF diagnostics: effective sample size $N_{\text{eff}}$ and weighted scan RMSE.

Beam sub-sampling and resampling triggered by $N_{\text{eff}}$ reduce cost while maintaining stable localization. A known advantage of PF methods is their ability to represent non-Gaussian and multi-modal uncertainty, which makes them suitable for problems such as global localization. Although global localization is not the primary focus of this chapter, the PF formulation supports it.

In some trajectory cases, the particle filter exhibited larger transient error during the early portion of the run before improving after additional motion. This behavior is consistent with the need for the particle set to concentrate around the correct pose using successive LiDAR observations. The effect can be more noticeable when the initial local geometry is less distinctive, such as along aisle-like regions where nearby poses may generate similar range patterns.

## 2.6 Extended Kalman Filter Fusion of Encoder and ICP

Encoder odometry provides smooth short-term motion estimates but drifts, while ICP provides map-anchored pose updates but can occasionally become less reliable when overlap or structure is weak. The EKF combines both by using encoder odometry as the prediction model and ICP pose as the measurement for correction. In addition, ICP measurement trust is adapted online by scaling the measurement covariance based on ICP RMSE and innovation gating.

### 2.6.1 State, Process, and Measurement Models

The EKF state is the planar pose

$$\mathbf{x}_k = [x_k,\, y_k,\, \theta_k]^T.$$

**Process model (encoder-driven).** Using encoder-derived velocities $(v_k, \omega_k)$, the nonlinear process model is

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, v_k, \omega_k) + \mathbf{w}_k, \qquad \mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}),$$

with

$$f(\mathbf{x}_k, v_k, \omega_k) = \begin{bmatrix} x_k + v_k \Delta t \cos\theta_k \\ y_k + v_k \Delta t \sin\theta_k \\ \text{wrap}(\theta_k + \omega_k \Delta t) \end{bmatrix}.$$

Linearizing about $(\theta_k, v_k)$ yields

$$\mathbf{F}_k = \begin{bmatrix} 1 & 0 & -v_k \Delta t \sin\theta_k \\ 0 & 1 & v_k \Delta t \cos\theta_k \\ 0 & 0 & 1 \end{bmatrix}.$$

**Measurement model (ICP pose).** ICP provides a pose measurement

$$\mathbf{z}_k = \begin{bmatrix} x_k^{\mathrm{ICP}} \\ y_k^{\mathrm{ICP}} \\ \theta_k^{\mathrm{ICP}} \end{bmatrix} = \mathbf{H}\mathbf{x}_k + \mathbf{v}_k, \qquad \mathbf{H} = \mathbf{I}_3, \qquad \mathbf{v}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_{\mathrm{eff},k}).$$

### 2.6.2 Adaptive Measurement Covariance from ICP RMSE

To reduce sensitivity to occasional unreliable ICP frames, the EKF uses an adaptive measurement covariance

$$\mathbf{R}_{\mathrm{eff},k} = \alpha_k \, \mathbf{R}_{\mathrm{base}}.$$

The implemented values are

$$\mathbf{Q} = \mathrm{diag}(0.02,\ 0.02,\ 0.01), \qquad \mathbf{R}_{\mathrm{base}} = \mathrm{diag}(0.05,\ 0.05,\ 0.30).$$

In the MATLAB implementation, $\alpha_k$ is computed from the ICP registration RMSE:

$$\alpha_k = \min\left(\alpha_{\max},\ \max\left(1, \left(\frac{\mathrm{RMSE}_k}{\mathrm{RMSE}_0}\right)^2\right)\right).$$

In the implementation used for this chapter, the reference value was set to $\mathrm{RMSE}_0 = 0.10$ m and the maximum scaling factor was set to $\alpha_{\max} = 50$.

The innovation is defined as

$$\boldsymbol{\nu}_k = \mathbf{z}_k - \hat{\mathbf{x}}_k^-, \qquad \nu_{k,3} \leftarrow \mathrm{wrap}(\nu_{k,3}),$$

with innovation position norm $\|\nu_{k,1:2}\|$ and yaw innovation $|\nu_{k,3}|$. The gate thresholds were set to 1.0 m for position innovation and 1.0 rad for yaw innovation. If either threshold is exceeded, the scaling factor is further increased according to

$$\alpha_k \leftarrow \max(\alpha_k,\ 25).$$

### 2.6.3 EKF Prediction and Correction

**Prediction.**

$$\hat{\mathbf{x}}_k^- = f(\hat{\mathbf{x}}_{k-1}, v_k, \omega_k), \tag{2.13}$$

$$\mathbf{P}_k^- = \mathbf{F}_k \mathbf{P}_{k-1} \mathbf{F}_k^\top + \mathbf{Q}. \tag{2.14}$$

**Correction.**

$$\mathbf{S}_k = \mathbf{H}\mathbf{P}_k^- \mathbf{H}^\top + \mathbf{R}_{\text{eff},k}, \tag{2.15}$$

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}^\top \mathbf{S}_k^{-1}, \tag{2.16}$$

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k \boldsymbol{\nu}_k, \tag{2.17}$$

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{H}) \mathbf{P}_k^-. \tag{2.18}$$

Angles are wrapped to keep $\theta \in (-\pi, \pi]$.

### 2.6.4 Baseline Results

Figure 2.10 shows the tracking error (GT vs desired) for closed-loop runs driven by different estimator feedback, while Figure 2.11 shows the estimation error (estimator vs GT) within each run. These two figures highlight different aspects of performance: tracking error reflects how closely the robot follows the desired path, whereas estimation error reflects how accurately each method estimates the true pose.

In the baseline results, encoder feedback leads to the largest position error growth, while ICP, EKF, and PF maintain lower position error overall. In the estimation results, ICP, EKF, and PF all significantly reduce pose error relative to encoder-only odometry. Although no single method achieves the lowest error for every metric, EKF fusion provides low estimation error while combining the smooth short-term motion continuity of encoder prediction with map-referenced ICP correction. Together with its relatively low computational cost, EKF provides a favorable balance among estimation accuracy, smooth state estimation, and computational efficiency in the tested MATLAB scenario.

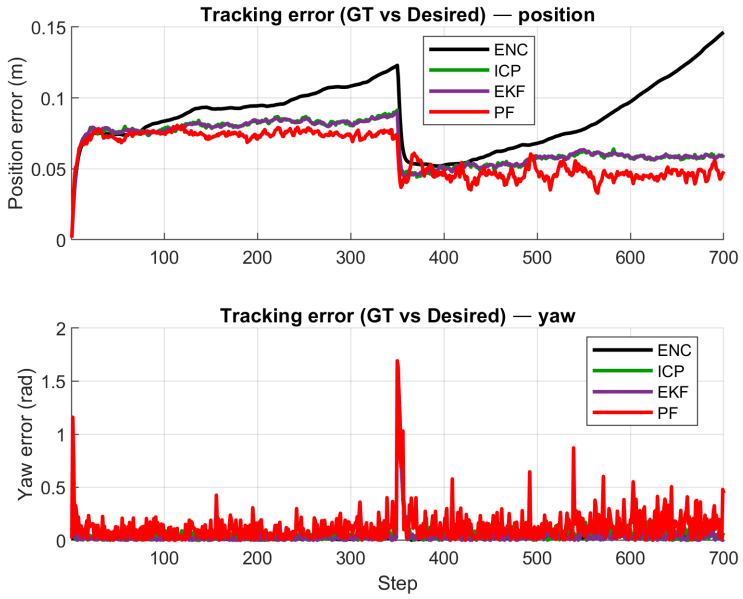



Figure 2.10: Baseline tracking error (GT vs desired) for closed-loop runs driven by different estimator feedback. Top: position error. Bottom: yaw error.

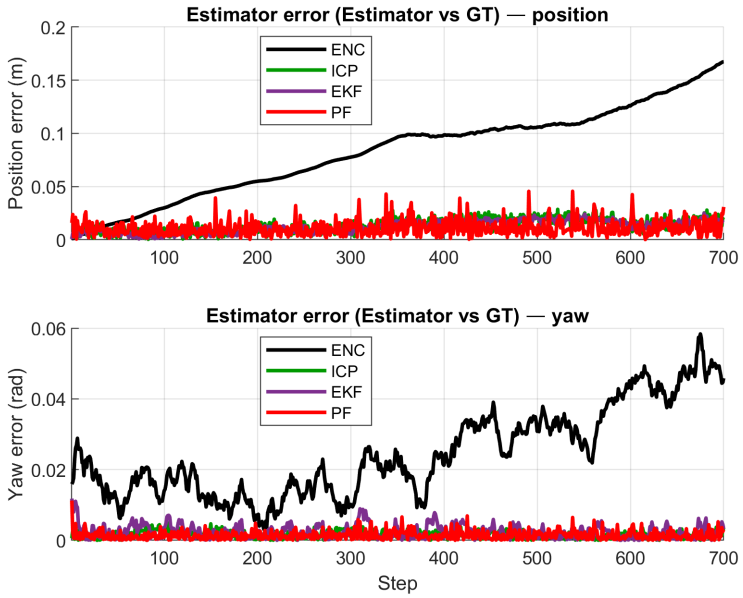



Figure 2.11: Baseline estimation error (estimator vs GT). Top: position error. Bottom: yaw error.

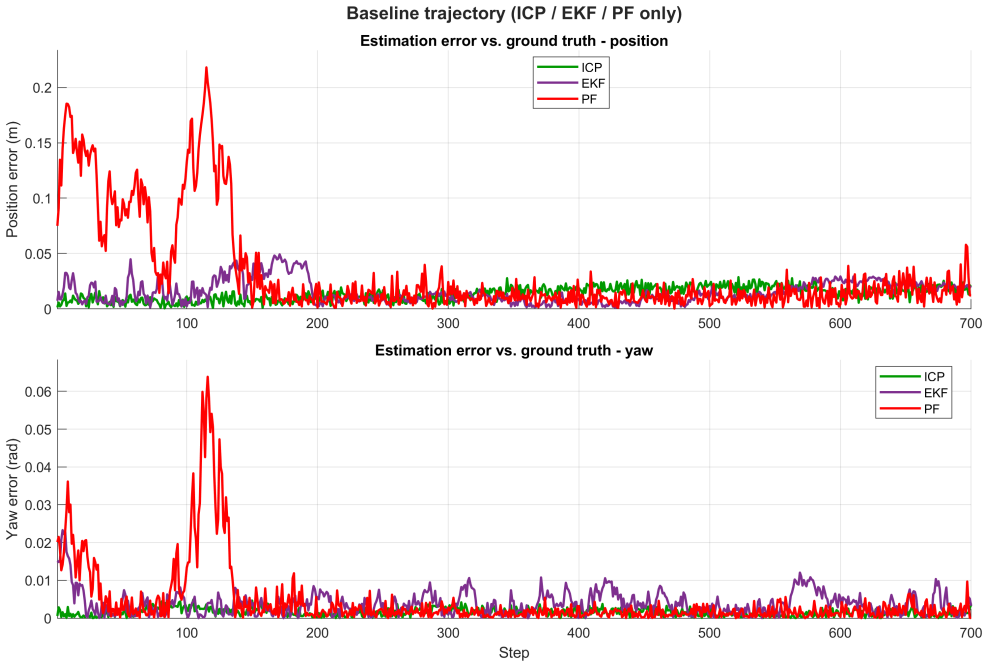



Figure 2.12: Enlarged view of baseline estimation errors excluding encoder-only odometry. Position and yaw errors for ICP, EKF, and PF are shown on a reduced vertical scale to improve visibility.

### 2.6.5 Comparison with a Conventional EKF under Temporary Map Mismatch

To isolate the benefit of adaptive measurement trust scaling, an additional comparison was performed between the proposed adaptive EKF and a conventional EKF using a fixed measurement covariance. Both filters used the same encoder-driven prediction model and the same ICP pose correction, but differed in how the ICP measurement covariance was handled. In the conventional EKF, the measurement covariance remained constant at $\mathbf{R}_{\text{base}}$. In the adaptive EKF, the effective covariance was scaled online according to ICP registration RMSE and innovation magnitude.

To make this comparison meaningful, a controlled map-mismatch scenario was introduced. A temporary extra obstacle was added to the *live* map used to generate LiDAR scans, while the reference map used by ICP remained unchanged. This created a short period of map mismatch lasting approximately 5 s, from 23.2 s to 28.2 s.

Figure 2.13 compares the estimation errors of the conventional EKF and the adaptive EKF over the full trajectory. Before the mismatch interval, both filters performed similarly. During the mismatch interval, however, the conventional EKF showed a sharper increase in both position and yaw error, indicating that it continued to trust corrupted ICP updates too strongly. By contrast, the adaptive EKF responded more cautiously by inflating the measurement covariance when ICP RMSE and innovation increased.

Quantitatively, the adaptive EKF reduced the overall estimation RMSE from 0.0751 m to 0.0591 m in position and from 0.0088 rad to 0.0051 rad in yaw. Tracking RMSE also improved, from 0.0967 m to 0.0885 m in position and from 0.2087 rad to 0.1853 rad in yaw. During the mismatch window itself, the position RMSE decreased from 0.2735 m for the conventional EKF to 0.2059 m for the adaptive EKF.

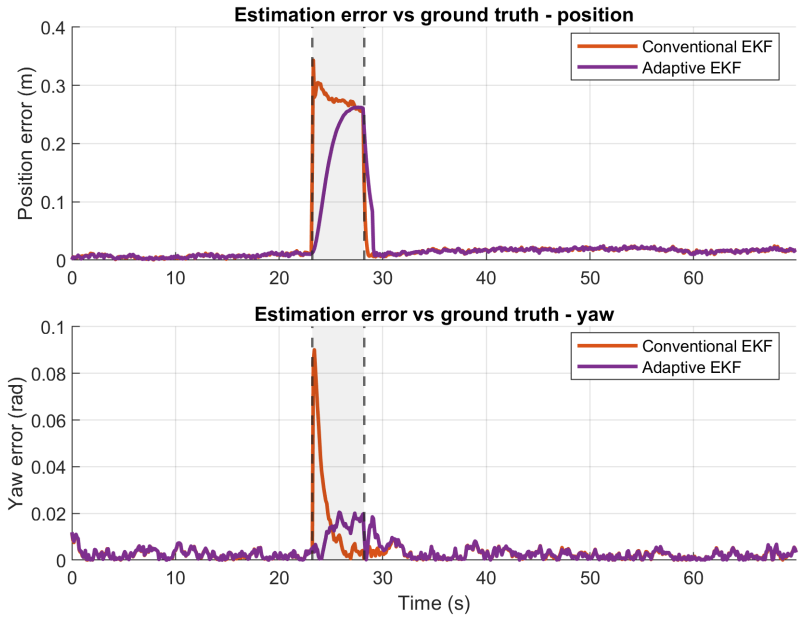



Figure 2.13: Estimation error comparison between a conventional EKF and the proposed adaptive EKF under a temporary map-mismatch scenario. The dashed vertical lines indicate the interval in which an extra obstacle was added to the live map.

Table 2.1: Conventional EKF versus adaptive EKF under temporary map mismatch.

| **Method** | **RMSE Pos (m)** | **RMSE Yaw (rad)** |
|---|---|---|
| Conventional EKF | 0.0751 | 0.0088 |
| Adaptive EKF | 0.0591 | 0.0051 |

### 2.6.6 Quantitative Summary for the Baseline Trajectory

Two complementary metrics are reported:

- **Tracking RMSE (GT vs Desired):** closed-loop path-following performance.
- **Estimation RMSE (Estimator vs GT):** estimator accuracy relative to the ground-truth trajectory of the corresponding run.

Table 2.2: Baseline tracking RMSE (GT vs Desired) for each estimator-driven closed-loop run.

| **Method** | **RMSE Pos (m)** | **RMSE Yaw (rad)** |
|---|---|---|
| Encoder (ENC) | 0.0958 | 0.1193 |
| ICP | 0.0693 | 0.1330 |
| Adaptive EKF (Enc+ICP) | 0.0692 | 0.1213 |
| PF | 0.0619 | 0.1981 |

Table 2.3: Baseline estimation RMSE (Estimator vs GT).

| **Method** | **RMSE Pos (m)** | **RMSE Yaw (rad)** |
|---|---|---|
| Encoder (ENC) | 0.0939 | 0.0359 |
| ICP | 0.0148 | 0.0019 |
| Adaptive EKF (Enc+ICP) | 0.0140 | 0.0045 |
| PF | 0.0130 | 0.0020 |

These baseline results show that scan-based and probabilistic methods achieve substantially lower estimation error than encoder-only odometry. In the tested closed-loop runs, the LiDAR-assisted estimators generally also produced lower position tracking RMSE than encoder-only feedback. Among the compared baseline methods, EKF remains particularly attractive because it provides both low error and a smooth state estimate suitable for closed-loop control.

### 2.6.7 Additional Trajectory Validation

To broaden the evaluation beyond the original representative trajectory, two additional warehouse-style motion patterns were simulated using the nominal MATLAB configuration. The first was a corridor-dominant path with a long straight segment followed by a turn into a second aisle. The second was a repeated-turn path containing several heading changes across the warehouse map. These two open-path tests retained the nominal stochastic encoder-noise model.

A separate repeated-loop stress test used a closed rectangular trajectory repeated over multiple laps without resetting the estimator state. For this stress test, the encoder model intentionally included the additional persistent wheel-scale factors 0.99 (left) and 1.01 (right) together with stochastic wheel-speed noise. The purpose was to make systematic odometric drift observable over a long repeated traversal, not to claim that the repeated-

loop run used identical encoder-error assumptions to the nominal tests. Overall, the additional experiments were intended as scenario-based validation rather than exhaustive testing.

Figure 2.14 and Figure 2.15 show the added open-path trajectories used for this expanded validation. Figure 2.16 shows the repeated-loop stress-test path.




Figure 2.14: Additional corridor-style trajectory used to examine localization behavior during extended straight-line motion in a warehouse aisle.




Figure 2.15: Additional repeated-turn trajectory used to examine localization behavior under multiple heading changes.




Figure 2.16: Repeated-loop stress-test trajectory used to examine long-duration accumulation effects without resetting the estimator state between laps.

Figures 2.17 and 2.18 show the estimator trajectory overlays for the two additional open-path cases. For conciseness, only the most informative trajectory and estimator-error plots are shown for these cases. The discussion focuses primarily on the position-error trends, while the corresponding quantitative summaries are provided in Tables 2.4–2.6.

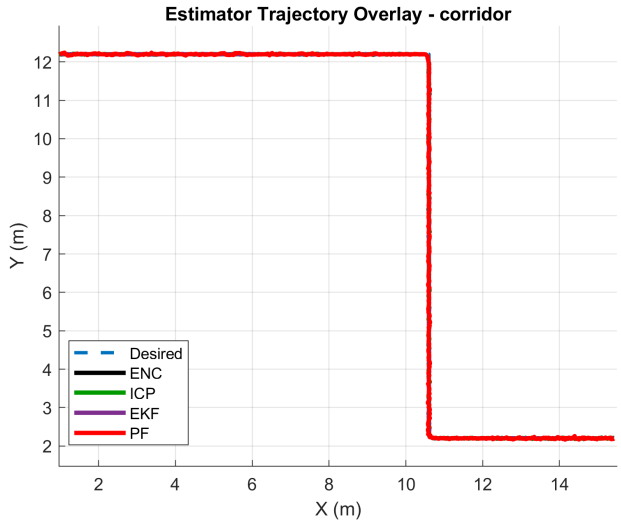



Figure 2.17: Estimator trajectory overlay for the corridor-style path. Encoder-only localization shows the largest deviation, while ICP and EKF remain closer to the desired path.

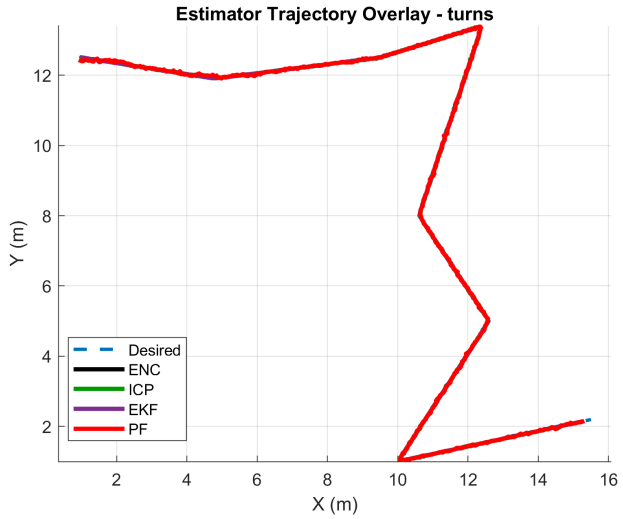



Figure 2.18: Estimator trajectory overlay for the repeated-turn path. The same overall trend as in the baseline case is preserved, with encoder-only localization deviating more than the LiDAR-assisted methods.

Figures 2.19 and 2.20 summarize the estimator error behavior for the corridor-style and repeated-turn cases. In both cases, encoder-only localization again shows the most pronounced error growth, while ICP and EKF remain better bounded. Figure 2.22 shows the repeated-loop stress-test result, where encoder error increases progressively over successive laps while EKF remains substantially better bounded over the longer traversal duration.

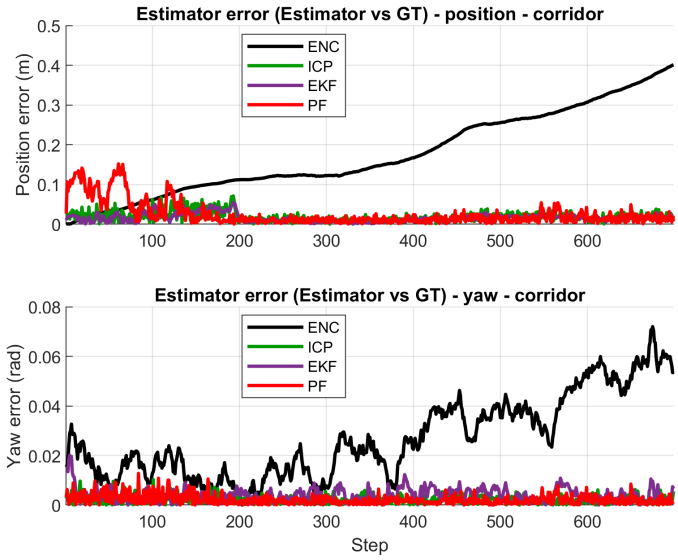



Figure 2.19: Estimator error for the corridor-style path. The position-error trend shows larger accumulated error for encoder-only localization, while ICP and EKF remain better bounded.

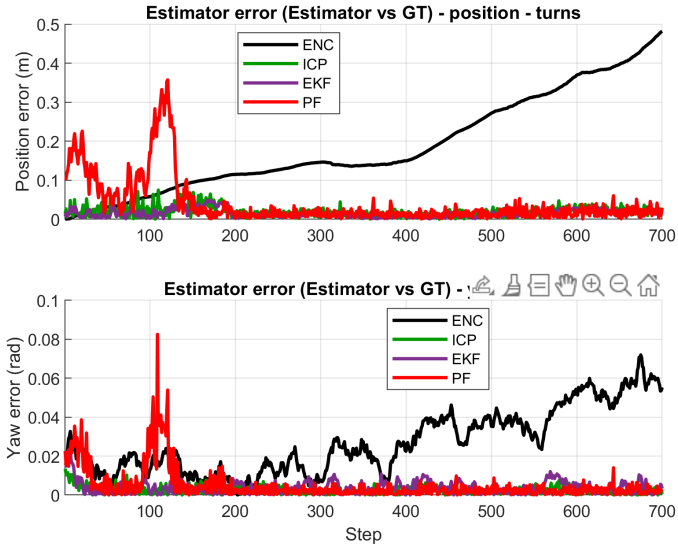



Figure 2.20: Estimator error for the repeated-turn path. Encoder-only localization again shows the largest cumulative drift, while ICP and EKF maintain lower error.

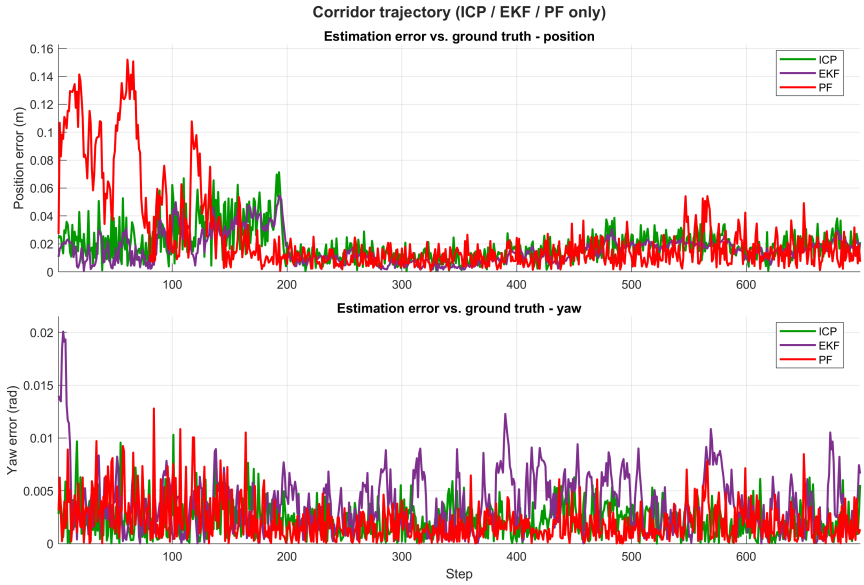



Figure 2.21: Enlarged view of estimation errors for the corridor-style trajectory, showing ICP, EKF, and PF without encoder-only odometry.

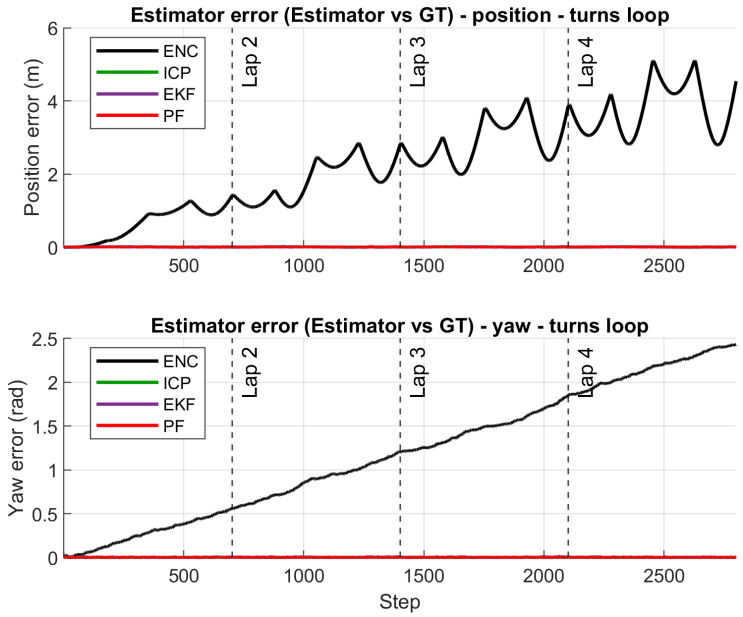



Figure 2.22: Estimator error for the repeated-loop stress test. Encoder error increases progressively over successive laps, while the LiDAR-assisted methods remain well bounded over the longer traversal duration.

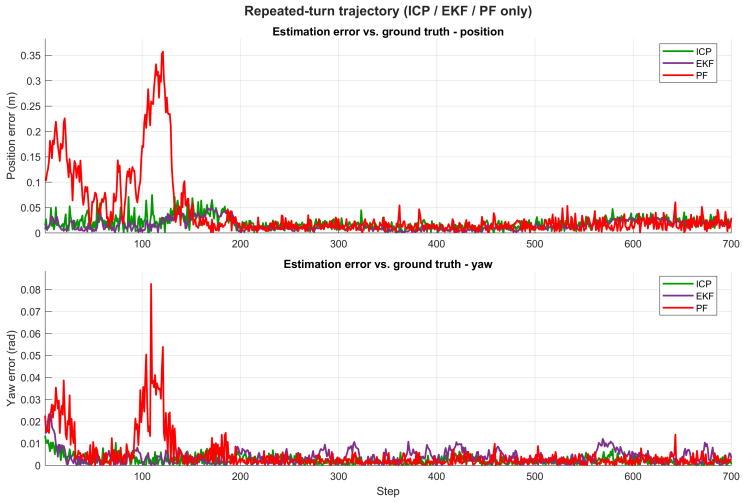



Figure 2.23: Enlarged view of estimation errors for the repeated-turn trajectory, showing ICP, EKF, and PF without encoder-only odometry.

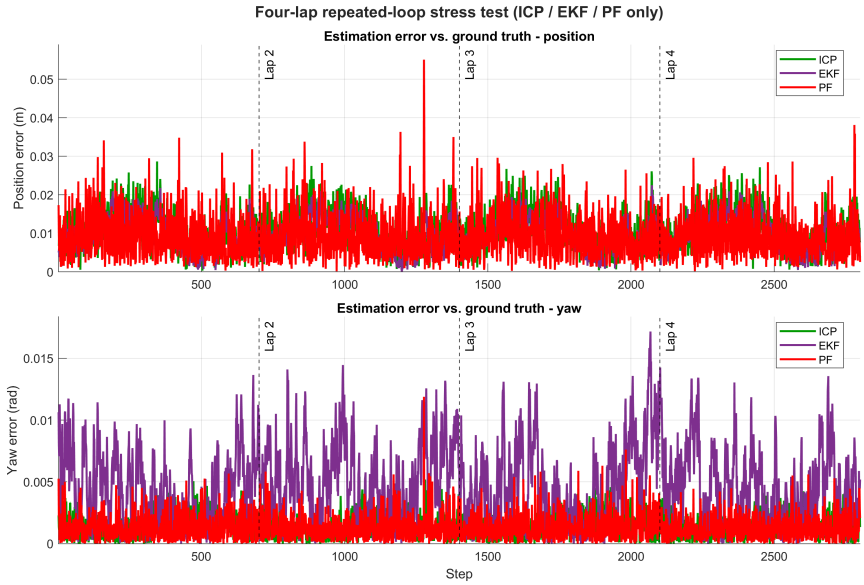



Figure 2.24: Enlarged view of estimation errors for the four-lap repeated-loop stress test, showing ICP, EKF, and PF without encoder-only odometry. Dashed vertical lines indicate lap boundaries.

Across the baseline, corridor, and repeated-turn trajectories, the same overall qualitative behavior was observed. Encoder-only localization accumulated the largest error, particularly in the corridor case and in the repeated-turn case where heading changes occurred more frequently. ICP-based localization generally reduced this drift by anchoring the estimate to the map geometry, while EKF fusion provided the most consistently smooth overall behavior across the tested cases.

The repeated-loop stress test further clarified long-duration behavior. Unlike the nominal open-path tests, this experiment intentionally included a small persistent left–right wheel-scale mismatch in addition to stochastic encoder noise. Because the two wheel signals were scaled differently, the resulting odometric error contained a systematic component rather than only zero-mean random variation. Repeated integration of this bias produced the large and steadily increasing encoder position and heading errors observed from lap to lap.

By contrast, the LiDAR-assisted methods repeatedly referenced the map and remained well bounded over the same traversal. In the final repeated-loop configuration, ICP, EKF, and PF maintained nearly constant position RMSE across laps, whereas encoder-only localization showed strong cumulative growth. The result should therefore be interpreted as an intentional stress test of long-duration odometric bias, not as evidence that zero-mean

encoder noise alone necessarily produces the same rate of yaw-error growth.

The particle-filter results in the additional trajectory cases remained more sensitive to the selected tuning than the encoder, ICP, and EKF cases, particularly during sharper turning transitions. For that reason, the interpretation of the added-path experiments is based primarily on the comparative trends of encoder odometry, ICP localization, and EKF fusion.

Table 2.4: Summary of additional open-path trajectory results.

| **Trajectory** | **Method** | **Track Pos RMSE (m)** | **Track Yaw RMSE (rad)** | **Est Pos RMSE (m)** | **Est Yaw RMSE (rad)** |
|---|---|---|---|---|---|
| Corridor | ENC | 0.2563 | 0.1525 | 0.2051 | 0.0316 |
| Corridor | ICP | 0.1536 | 0.1568 | 0.0230 | 0.0028 |
| Corridor | EKF | 0.1539 | 0.1542 | 0.0189 | 0.0048 |
| Corridor | PF | 0.1598 | 0.1952 | 0.0398 | 0.0028 |
| Repeated-turns | ENC | 0.2903 | 0.2213 | 0.2263 | 0.0316 |
| Repeated-turns | ICP | 0.1870 | 0.2282 | 0.0230 | 0.0030 |
| Repeated-turns | EKF | 0.1867 | 0.2241 | 0.0183 | 0.0050 |
| Repeated-turns | PF | 0.1983 | 0.2656 | 0.0727 | 0.0095 |

Table 2.5: Overall repeated-loop stress-test results over four laps.

| **Method** | **Track Pos RMSE (m)** | **Track Yaw RMSE (rad)** | **Est Pos RMSE (m)** | **Est Yaw RMSE (rad)** |
|---|---|---|---|---|
| ENC | 2.5806 | 1.3819 | 2.6204 | 1.3939 |
| ICP | 0.0908 | 0.2210 | 0.0122 | 0.0018 |
| EKF | 0.0907 | 0.2181 | 0.0110 | 0.0056 |
| PF | 0.0892 | 0.2286 | 0.0107 | 0.0017 |

Table 2.6: Lap-by-lap repeated-loop stress-test results for encoder-only localization and EKF fusion.

| **Lap** | **ENC Track Pos RMSE (m)** | **EKF Track Pos RMSE (m)** | **ENC Est Pos RMSE (m)** | **EKF Est Pos RMSE (m)** |
|---|---|---|---|---|
| Lap 1 | 0.7218 | 0.0905 | 0.7710 | 0.0109 |
| Lap 2 | 1.8364 | 0.0907 | 1.8955 | 0.0110 |
| Lap 3 | 2.8846 | 0.0907 | 2.9346 | 0.0110 |
| Lap 4 | 3.7979 | 0.0908 | 3.8297 | 0.0111 |

These additional trajectory tests broaden the evaluation beyond a single representative path and increase confidence that the observed estimator behavior is not limited to one specific geometry. However, because all results in this chapter are obtained in controlled simulation, they should still be interpreted as scenario-based evidence rather than as comprehensive proof of robustness in real warehouse operation.

## 2.7 Evaluation Metrics and Experimental Protocol

Localization accuracy is evaluated using position and orientation errors relative to ground truth. Let $\hat{\mathbf{x}}_k = [\hat{x}_k, \hat{y}_k, \hat{\theta}_k]^\top$ denote an estimated pose and $\mathbf{x}_k^{\mathrm{GT}} = [x_k^{\mathrm{GT}}, y_k^{\mathrm{GT}}, \theta_k^{\mathrm{GT}}]^\top$ denote the corresponding ground-truth pose. The position error and wrapped yaw error are defined as

$$e_k^{\mathrm{pos}} = \sqrt{(\hat{x}_k - x_k^{\mathrm{GT}})^2 + (\hat{y}_k - y_k^{\mathrm{GT}})^2}, \tag{2.19}$$

$$e_k^{\theta} = \mathrm{wrap}\big(\hat{\theta}_k - \theta_k^{\mathrm{GT}}\big). \tag{2.20}$$

In experiments where each method is executed as a separate closed-loop run, $\mathbf{x}_k^{\mathrm{GT}}$ refers to the ground-truth trajectory of the corresponding run. We report time histories as well as trajectory-level RMSE. All methods use the same reference trajectory and sampling time $\Delta t = 0.1$ s. For stochastic components, a fixed pseudorandom seed is used for reproducibility.

## 2.8 Runtime and Computational Considerations

Encoder odometry and EKF updates are lightweight in the present 3-state formulation, whereas ICP scan-to-map registration and PF updates introduce the main computational

burden. In particular, ICP runtime depends on point cloud size, correspondence search, and iteration count, while PF cost scales approximately as $\mathcal{O}(N\,B)$, where $N$ is the number of particles and $B$ is the number of LiDAR beams evaluated after sub-sampling.

For practical engineering comparison, runtime was measured as the mean milliseconds per update over the full simulation loop on the development platform used for this study. The observed runtimes were approximately ENC: 6.04 ms/update, ICP: 6.83 ms/update, EKF: 6.52 ms/update, and PF: 338.23 ms/update. These values are hardware- and implementation-dependent and include the shared simulation framework in addition to the estimator-specific computations. Nevertheless, they make the relative computational trade-off clear: encoder odometry, ICP, and EKF remain comparatively lightweight in this implementation, whereas PF is substantially more expensive due to particle propagation and scan-likelihood evaluation.

These runtime results complement the accuracy comparisons presented earlier. In particular, PF can achieve low estimation error, but at a substantially higher computational cost, while EKF provides a favorable balance between accuracy and efficiency for the simulated warehouse localization problem considered in this chapter. At the sensor update rate used in this study (10 Hz, corresponding to a 100 ms time budget per update), the current MATLAB implementation of the PF is not real-time capable. In this chapter, PF is therefore used primarily as a simulation benchmark rather than a deployment-ready real-time solution.

## 2.9 Limitations, Threats to Validity, and Chapter Summary

**Limitations.** (i) The occupancy map is idealized; real environments include clutter and dynamic obstacles. (ii) LiDAR noise is modeled primarily as Gaussian range noise and does not explicitly simulate incidence-angle effects, mixed-pixel returns, or ghost reflections. (iii) Parameters such as noise levels, PF settings, ICP tolerances, and EKF covariances were manually tuned for stable operation; automated tuning may yield different trade-offs.

**Threats to validity.** Although the evaluation was expanded beyond the original representative path to include a corridor-style trajectory, a repeated-turn trajectory, and a repeated-loop stress test, the results still represent a limited set of simulation scenarios. The baseline, corridor, and repeated-turn cases use the nominal sensor-error assumptions, whereas the repeated-loop stress test intentionally adds a persistent wheel-scale asymmetry to examine systematic long-duration odometric drift. The added cases improve confidence

that the observed qualitative trends are not tied to a single path geometry, but they do not constitute exhaustive validation across all warehouse layouts, operating speeds, controller settings, encoder-bias levels, or sensing conditions.

For stochastic components, a fixed pseudorandom seed was used for reproducibility. This does not reduce the underlying stochastic variance of the algorithms; rather, it ensures that the reported comparisons can be reproduced. Variation across multiple independent seeds was not quantified. PF behavior in the added-path cases remained more tuning-sensitive than the encoder, ICP, and EKF results. In addition, because the present results are simulation-based, broader claims about robustness in real robotic deployment should be made cautiously until supported by hardware experiments.

**Chapter summary.**

- Encoder odometry is computationally cheap but accumulates drift over time due to the integration of noisy motion estimates.
- ICP scan-to-map localization substantially reduces this drift by anchoring pose estimates to the warehouse geometry, although its performance depends on scan quality, geometry, and initialization.
- EKF fusion combines encoder prediction with ICP correction and uses adaptive measurement covariance to reduce sensitivity to unreliable ICP frames. Across the tested MATLAB scenarios, it provides a favorable balance between estimation accuracy, smooth state estimation, and computational efficiency.
- Particle-filter localization achieves low estimation error in the baseline simulation, but its behavior in the additional trajectory cases is more sensitive to tuning, particularly during sharper turning transitions.
- The added corridor and repeated-turn tests broaden the nominal evaluation beyond a single representative trajectory, while the repeated-loop experiment provides a separate long-duration stress test with an intentional 0.99/1.01 left–right wheel-scale asymmetry. Across these tested scenarios, encoder-only localization accumulated the largest error, while the LiDAR-assisted methods remained better bounded.
- These results should be interpreted as scenario-based simulation evidence rather than as exhaustive validation over all warehouse operating conditions.

**Next step.** The next chapter builds on these MATLAB results by examining reliability-aware localization in a more realistic implementation setting. While this chapter demonstrated the mathematical behavior of encoder odometry, ICP, PF, and adaptive EKF

fusion under controlled simulation conditions, the next chapter focuses more directly on robustness mechanisms for changing ICP reliability, including degraded scan matching, temporary map inconsistency, and the use of high-quality ICP updates to reduce encoder drift.

# Chapter 3

# Reliability-Aware Localization Enhancements

## 3.1 Motivation and Problem Statement

Chapter 2 compared multiple localization strategies, including encoder odometry, ICP-based LiDAR localization, EKF fusion, and particle filtering, under controlled warehouse simulation conditions. Those experiments provided a baseline comparison of the estimators, but they also motivated a practical question: how should the localization system respond when the reliability of scan-to-map registration changes over time?

Throughout this chapter, **ground truth (GT)** refers to the simulation-provided robot pose used as the reference for quantitative evaluation.

Two failure modes are particularly relevant in warehouse-like environments:

1. **Scan-to-map mismatch.** A temporary obstacle, such as a pallet, person, or blockage, may appear in the live LiDAR scan while being absent from the reference map. ICP may still return a finite transformation under this condition, but the returned pose can disagree strongly with the short-term motion prediction or, in a more severe case, converge to an incorrect local minimum. Directly accepting such a pose as reliable feedback can degrade closed-loop motion.

2. **Accumulated encoder drift.** Encoder odometry provides smooth local motion information, but small wheel-speed errors accumulate over time. If the encoder state

is allowed to drift indefinitely, it becomes a weaker fallback source during later periods in which LiDAR localization must be down-weighted.

The objective of this chapter is therefore to make the localization pipeline responsive to time-varying ICP reliability. The proposed architecture separates three related but distinct mechanisms:

1. **Adaptive EKF measurement weighting:** ICP RMSE and large innovation checks are used to scale the effective ICP measurement covariance within the EKF.
2. **Fuzzy feedback blending:** ICP RMSE, position innovation, and yaw innovation are combined by a Mamdani Fuzzy Inference System (FIS) to produce a continuous trust score used to blend encoder, EKF, and ICP feedback.
3. **Sustained-good encoder anchoring:** when ICP remains valid and consistent for a required number of consecutive frames, the encoder state used inside the FUZZY-BLEND method is gradually moved toward the current trusted ICP pose.

These mechanisms are intentionally separated. The fuzzy trust score does not directly control the EKF covariance scaling, and it does not by itself activate encoder anchoring. Each mechanism uses the relevant reliability indicators for a specific purpose.

Figure 3.1 summarizes the reliability-aware localization concept.

## 3.2 Experimental Configuration for the Reliability Tests

The experiments in this chapter retain the same basic robot, reference map, LiDAR model, and trajectory-following structure used in the MATLAB framework of Chapter 2. The LiDAR range-noise standard deviation used in the simulation code is 0.10 m, and the horizontal angular-noise standard deviation is 0.2°. These values should be distinguished from estimator tuning parameters such as the particle-filter likelihood width discussed in Chapter 2.

The control and estimation update period is

$$\Delta t = 0.1 \text{ s}.$$

A fixed pseudorandom seed is used for reproducibility of the stochastic simulation components.

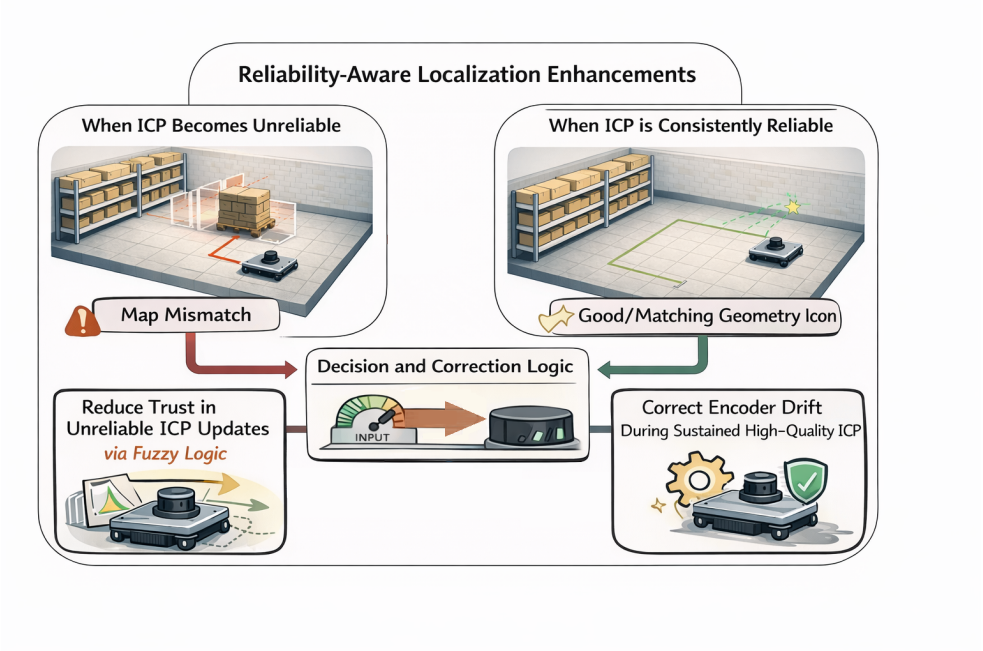



Figure 3.1: Reliability-aware localization concept used in this chapter. Finite but unreliable ICP measurements are down-weighted within the EKF, fuzzy trust controls controller-level feedback blending, and sustained high-quality ICP permits gradual encoder anchoring within FUZZY-BLEND.

To create a controlled scan-to-map mismatch, a temporary obstacle is inserted into the *live* occupancy map while the fixed reference map used by ICP remains unchanged. The obstacle is introduced approximately one-third of the way through the trajectory and remains active for approximately 3 s. Therefore, the mismatch experiment in this chapter differs from a nominal static-map run by the temporary live-map modification and by the reliability-aware estimation and feedback logic described below.

Four primary feedback configurations are evaluated:

- **ENC:** raw encoder odometry only. No ICP-based encoder anchoring is applied in this baseline.
- **ICP:** the current finite ICP pose is used as feedback. If ICP registration fails or returns a non-finite result, encoder feedback is used for that sample to maintain control continuity.
- **EKF:** encoder-driven prediction followed by ICP measurement correction using the reliability-dependent covariance scaling described in Section 3.3.4.
- **FUZZY-BLEND:** weighted feedback blending of encoder, EKF, and current valid ICP estimates using the filtered fuzzy trust score. Sustained-good encoder anchoring is enabled only in this configuration.

An additional FUZZY-BLEND run with encoder anchoring disabled is used only as an ablation case for the anchoring diagnostics. This separation is important: the reported ENC results represent a true encoder-only baseline, whereas encoder anchoring is part of the proposed FUZZY-BLEND architecture.

## 3.3 Reliability Metrics and Decision Logic

This section defines the signals used to evaluate ICP reliability at each time step. The same basic indicators support different parts of the architecture. RMSE and innovation checks are used to adjust the EKF measurement covariance, the three indicators are combined by the FIS for controller-level feedback blending, and explicit validity and consistency gates are used to determine whether encoder anchoring is permitted.

### 3.3.1 ICP Registration Error (RMSE)

At each time step, the live LiDAR scan point cloud is registered to the fixed reference-map point cloud using point-to-point ICP (`pcregistericp`). MATLAB reports an alignment residual in the form of a root-mean-square-error (RMSE). This quantity summarizes the residual distances between the transformed scan and corresponding map points after registration.

The reported value at time step $k$ is denoted

$$\mathrm{RMSE}_k.$$

Lower values generally indicate stronger scan-to-map consistency, whereas larger values can indicate reduced overlap, outliers, partial occlusion, or scan-to-map mismatch.

As shown in Fig. 3.2, the RMSE remains close to its nominal level for most of the run and increases during the interval in which the temporary obstacle is present only in the live environment. Because the obstacle is absent from the reference point cloud, scan-to-map consistency is reduced.

RMSE is useful but is not sufficient by itself. In repetitive or locally ambiguous geometry, ICP can occasionally converge to an incorrect pose while retaining a relatively modest residual. For this reason, RMSE is combined with a motion-consistency check based on the innovation relative to the encoder-driven prediction.

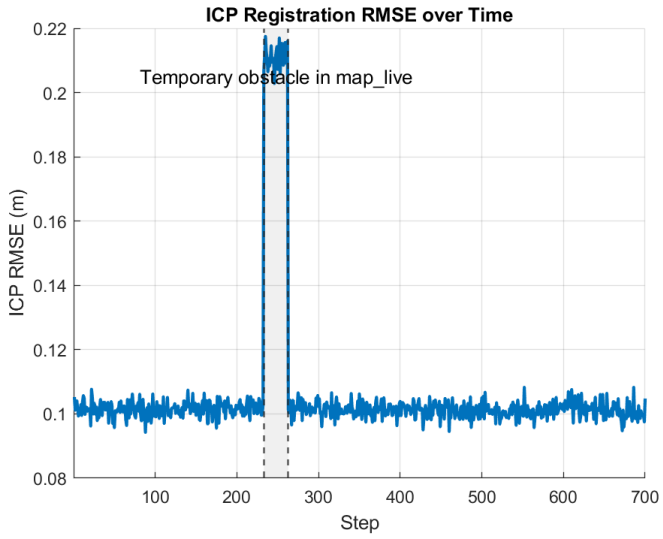



Figure 3.2: ICP registration RMSE over time. Dashed vertical lines mark the interval during which a temporary obstacle is present in the live map but absent from the ICP reference map. The mismatch increases registration inconsistency, while the comparison with the pose-error results shows why RMSE should not be interpreted as a complete correctness test by itself.

### 3.3.2 Innovation Relative to Encoder-Based Prediction

A second reliability signal compares the ICP measurement with the predicted pose obtained from the encoder-driven EKF propagation step. This comparison measures disagreement between the map-referenced measurement and the short-term motion prediction.

Let the encoder-driven predicted state at time step $k$ be

$$\mathbf{x}_k^- = \begin{bmatrix} x_k^- & y_k^- & \theta_k^- \end{bmatrix}^T,$$

and let the ICP-measured pose be

$$\mathbf{z}_k^{\mathrm{ICP}} = \begin{bmatrix} x_k^{\mathrm{ICP}} & y_k^{\mathrm{ICP}} & \theta_k^{\mathrm{ICP}} \end{bmatrix}^T.$$

The innovation is

$$\boldsymbol{\nu}_k = \begin{bmatrix} x_k^{\mathrm{ICP}} - x_k^- \\ y_k^{\mathrm{ICP}} - y_k^- \\ \mathrm{wrap}\left(\theta_k^{\mathrm{ICP}} - \theta_k^-\right) \end{bmatrix},$$

where the wrap operator keeps angular differences in $(-\pi, \pi]$.[1]

For gating and visualization, the innovation is separated into position and yaw magnitudes:

$$\nu_k^{\text{pos}} = \left\| \begin{bmatrix} \nu_{x,k} \\ \nu_{y,k} \end{bmatrix} \right\|_2, \tag{3.1}$$

$$\nu_k^{\theta} = |\nu_{\theta,k}|. \tag{3.2}$$

A large innovation indicates strong disagreement between the ICP measurement and the predicted state. It does *not*, by itself, prove that ICP is incorrect, because prediction error can also contribute to the disagreement. In the present framework, innovation is therefore treated as a consistency indicator and is interpreted together with ICP registration quality.

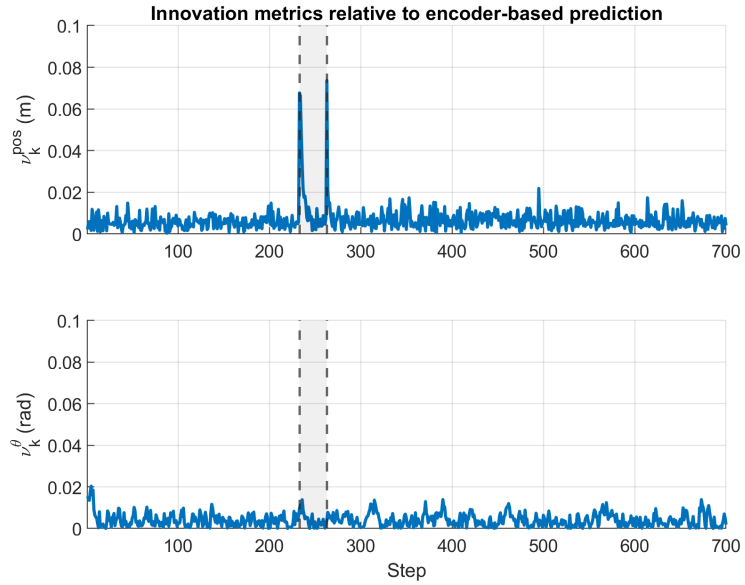



Figure 3.3: Innovation metrics relative to the encoder-based prediction: (top) position innovation $\nu_k^{\text{pos}}$, and (bottom) yaw innovation $\nu_k^{\theta}$. Increased innovation indicates stronger disagreement between the ICP measurement and the predicted state and is used as an indicator of reduced measurement consistency.

[1] In implementation, the wrap operator is evaluated as $\text{wrap}(\alpha) = \text{atan2}(\sin\alpha, \cos\alpha)$.

### 3.3.3 Trust Mapping via Fuzzy Inference System (FIS)

To avoid abrupt controller-level switching and to express the reliability decision in an interpretable rule-based form, a fuzzy mapping converts the three reliability indicators into a continuous trust value. A Mamdani FIS is used because it provides a direct way to combine nonlinear reliability cues without assuming that ICP failure can be represented by a single threshold.

**FIS inputs and output.** The three inputs are

$$u_{1,k} = \mathrm{RMSE}_k, \qquad u_{2,k} = \nu_k^{\mathrm{pos}}, \qquad u_{3,k} = \nu_k^{\theta},$$

and the instantaneous FIS output is

$$t_k \in [0, 1].$$

The output $t_k$ is an algorithmic trust score used for blending; it is not intended to represent a calibrated probability of ICP correctness.

**Membership functions and breakpoints.** Each input uses overlapping linguistic membership functions, *Small (S)*, *Medium (M)*, and *Large (L)*. Let $a$ and $b$ denote the corresponding low and high thresholds, and define $m = (a + b)/2$. The Small, Medium, and Large sets use the parameter sets $[0, 0, a, m]$ (trapezoidal), $[a, m, b]$ (triangular), and $[m, b, u_{\max}, u_{\max}]$ (trapezoidal), respectively.

The input domains used in the MATLAB implementation are $[0, 1]$ m for RMSE, $[0, 6]$ m for position innovation, and $[0, 3.5]$ rad for yaw innovation. The output membership functions are Low $[0, 0, 0.20, 0.45]$, Medium $[0.25, 0.55, 0.80]$, and High $[0.60, 0.85, 1, 1]$. The Mamdani implementation uses minimum for AND and implication, maximum for OR and aggregation, and centroid defuzzification.

Table 3.1: FIS breakpoint parameters used to define the input membership functions.

| Quantity | Low threshold | High threshold |
|---|---|---|
| RMSE (m) | 0.10 | 0.30 |
| $\nu^{\mathrm{pos}}$ (m) | 0.30 | 1.00 |
| $\nu^{\theta}$ (rad) | 0.30 | 1.00 |

Figure 3.4 shows the input and output membership functions used in the implemented FIS.

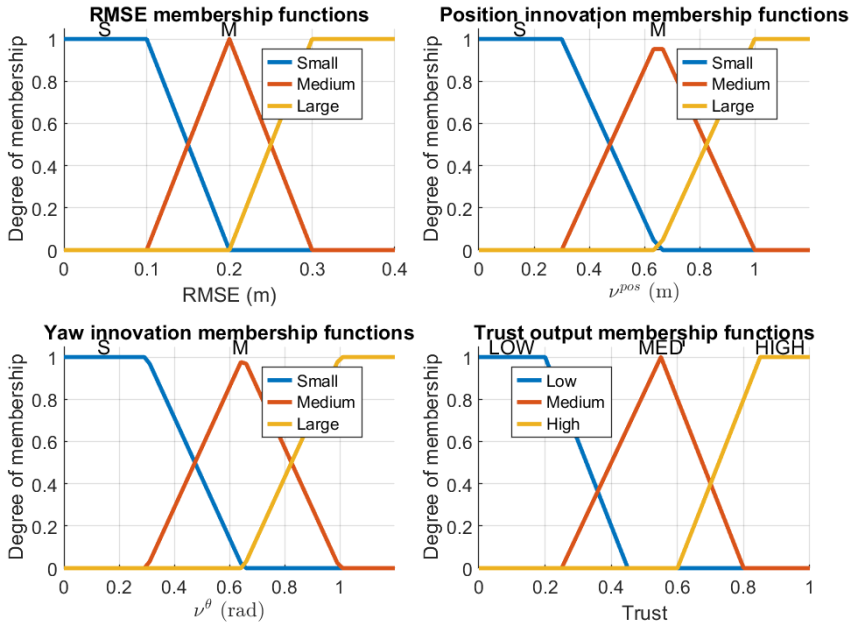



Figure 3.4: Input and output membership functions used in the Mamdani FIS. The three input variables use overlapping Small, Medium, and Large sets, while the output trust variable uses Low, Medium, and High sets.

**Rule base and mixed-case behavior.** The fuzzy rules implement the intended reliability behavior:

- Small RMSE together with small innovations produces high trust.
- A Large value in any one of the three reliability indicators produces low trust.
- Intermediate combinations produce high or medium trust according to the complete rule base below.

In the obstacle-mismatch experiment, the ICP residual alone does not fully describe the severity of every pose error. Position and yaw innovations therefore provide complementary consistency information.

For reproducibility, Table 3.2 gives the complete rule base implemented in MATLAB. Here S, M, and L denote Small, Medium, and Large, respectively, and "Any" represents a wildcard condition.

Table 3.2: Complete Mamdani FIS rule base implemented in MATLAB.

| Rule | RMSE | $\nu^{\text{pos}}$ | $\nu^{\theta}$ | Trust |
|---|---|---|---|---|
| R1 | S | S | S | High |
| R2 | S | S | M | High |
| R3 | S | M | S | High |
| R4 | M | S | S | Medium |
| R5 | M | M | S | Medium |
| R6 | M | S | M | Medium |
| R7 | S | M | M | Medium |
| R8 | M | M | M | Medium |
| R9 | L | Any | Any | Low |
| R10 | Any | L | Any | Low |
| R11 | Any | Any | L | Low |

**Filtered trust.** To reduce frame-to-frame variation, the instantaneous trust value is low-pass filtered:

$$\bar{t}_k = \bar{t}_{k-1} + \beta \left( t_k - \bar{t}_{k-1} \right),$$

with

$$\beta = 0.15.$$

This filtered score is the quantity used for feedback blending.

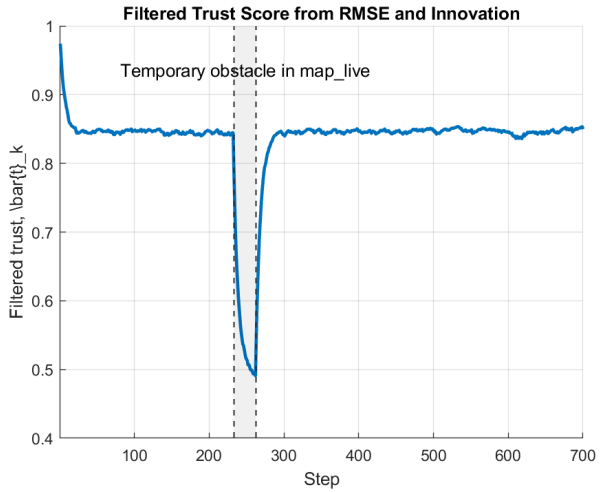



Figure 3.5: Filtered fuzzy trust score $\bar{t}_k$. The score decreases when the ICP reliability indicators become less consistent and recovers smoothly when scan-to-map consistency returns.

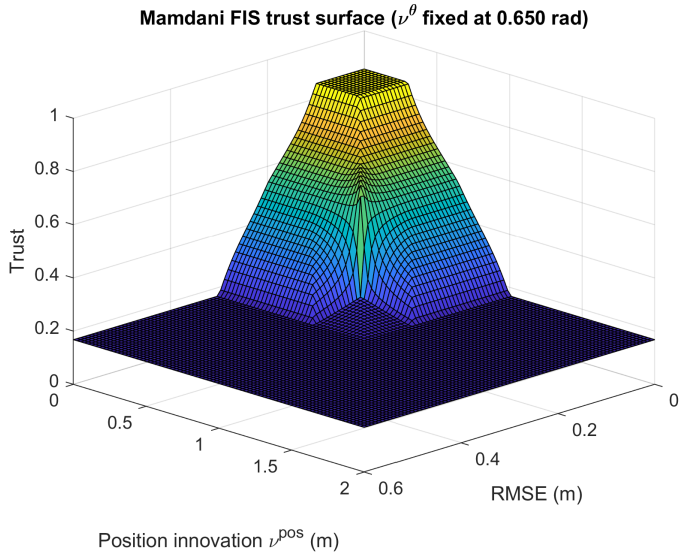



Figure 3.6: Example Mamdani FIS trust surface with yaw innovation fixed at approximately 0.650 rad. The maximum defuzzified value is approximately 0.85, rather than 1.0, because centroid defuzzification is applied to the High output membership function $[0.60, 0.85, 1, 1]$.

### 3.3.4 Decision Logic for Unreliable and Sustained-Good ICP

The reliability-aware architecture distinguishes between a finite but questionable ICP measurement and an unavailable or non-finite ICP measurement.

**Finite but unreliable ICP.** When ICP returns a finite pose and finite RMSE, the measurement can still be used, but its influence within the EKF is reduced as reliability decreases. The effective covariance is

$$\mathbf{R}_{\text{eff},k} = \alpha_k \mathbf{R}_{\text{base}}, \qquad \alpha_k \geq 1.$$

The RMSE-dependent scale is

$$\alpha_k = \min\left(\alpha_{\max}, \max\left[1, \left(\frac{\text{RMSE}_k}{\text{RMSE}_0}\right)^2\right]\right),$$

where

$$\text{RMSE}_0 = 0.10 \text{ m}, \qquad \alpha_{\max} = 50.$$

If

$$\nu_k^{\text{pos}} > 1.0 \text{ m} \quad \text{or} \quad \nu_k^{\theta} > 1.0 \text{ rad},$$

the implementation enforces

$$\alpha_k \geq 25.$$

Thus, a finite measurement with large RMSE or large prediction disagreement remains numerically usable but receives reduced weight in the EKF correction.

**Invalid or non-finite ICP.** A failed or non-finite ICP result is handled differently. Increasing the measurement covariance cannot make a non-finite pose usable. Therefore, when ICP registration fails or when the returned pose or RMSE is non-finite, the ICP measurement update is skipped for that sample:

$$\mathbf{x}_k = \mathbf{x}_k^-, \qquad \mathbf{P}_k = \mathbf{P}_k^-.$$

The invalid result is marked unavailable, it does not contribute to the sustained-good counter, and it is not used as the current ICP component of FUZZY-BLEND. In the ICP-feedback configuration, encoder feedback is used for that sample to maintain continuity. Normal ICP processing resumes when a finite measurement becomes available.

This distinction prevents non-finite data from entering the EKF while preserving the ability to down-weight finite but unreliable measurements.

**Controller-level fuzzy blending.** The filtered fuzzy trust score controls the controller-feedback weights:

$$w_{\mathrm{enc},k} = 1 - \bar{t}_k, \qquad w_{\mathrm{EKF},k} = \gamma_{\mathrm{EKF}}\bar{t}_k, \qquad w_{\mathrm{ICP},k} = (1 - \gamma_{\mathrm{EKF}})\bar{t}_k,$$

with

$$\gamma_{\mathrm{EKF}} = 0.70.$$

When all three estimates are valid, the weights sum to one. Position is blended by a weighted arithmetic mean and yaw is blended through weighted sine and cosine components to avoid angle-wrap discontinuities. If the current ICP pose is unavailable, it is excluded from the blend and the remaining valid weights are renormalized by the implementation.

The EKF covariance-scaling mechanism and fuzzy feedback weighting therefore operate in parallel: $\alpha_k$ modifies the estimator measurement update, whereas $\bar{t}_k$ modifies the controller-level feedback composition.

**Sustained-good ICP.** The opposite regime occurs when ICP remains finite, has sufficiently low RMSE, and remains consistent with the encoder-driven prediction over multiple consecutive samples. In this case, ICP can be used as a geometric reference for gradual encoder anchoring. This anchoring is enabled only in FUZZY-BLEND and only after the sustained-good criteria described in Section 3.4 have been satisfied.

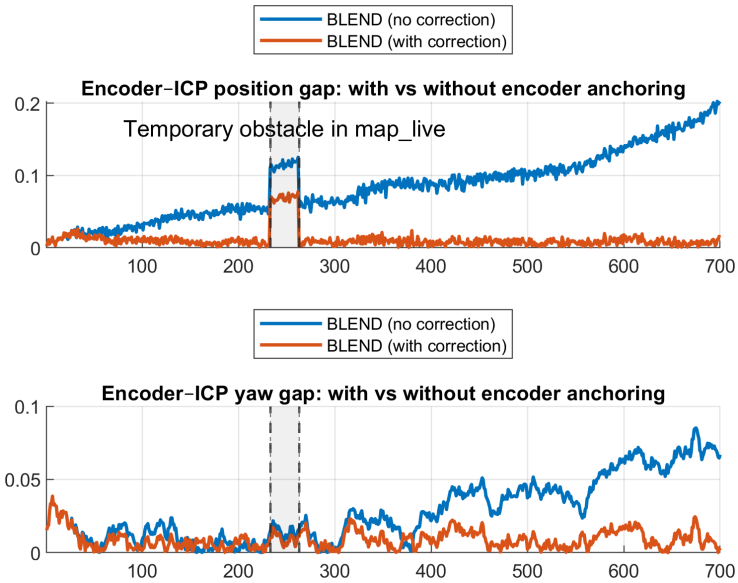



Figure 3.7: Encoder–ICP disagreement within FUZZY-BLEND with and without encoder anchoring. During unreliable ICP, anchoring is disabled. When reliable scan-to-map registration persists long enough to satisfy the sustained-good gate, the anchoring mechanism gradually reduces the encoder–ICP position and yaw disagreement.

## 3.4 Encoder Drift Correction Using Sustained High-Quality ICP

The reliability logic above protects the estimator and controller when ICP becomes unreliable. The opposite condition can also be useful. During intervals in which ICP repeatedly aligns to the fixed reference map with low RMSE and small innovations, the map-referenced pose provides information that can be used to limit long-term encoder drift.

The anchoring mechanism described in this section is part of FUZZY-BLEND only. It is deliberately disabled in the standalone ENC, ICP, and EKF runs so that those methods retain clear interpretations and, in particular, the ENC result remains a true encoder-only baseline.

### 3.4.1 Why Correct the Encoder Pose?

Encoder odometry provides smooth short-term propagation but has no absolute reference. As a result, its pose error can accumulate over long intervals. The corrected results in Section 3.6.5 show this behavior directly: the raw ENC position RMSE increases from 0.0353 m in the pre-mismatch interval to 0.1152 m in the post-mismatch interval.

Within FUZZY-BLEND, the encoder estimate becomes increasingly important when fuzzy trust decreases. If that encoder state has already accumulated a large offset, shifting feedback toward it can introduce additional tracking error. Gradual anchoring during sustained reliable ICP is therefore used to keep the internal encoder state closer to the map-referenced solution before a later low-trust period occurs.

### 3.4.2 Detection of Sustained High-Quality ICP

An ICP update is considered *good* for anchoring when all of the following conditions are satisfied:

1. ICP returns a finite pose and finite RMSE;
2. $$\mathrm{RMSE}_k \le \mathrm{RMSE}_{\mathrm{good}} = 0.12 \text{ m};$$
3. $$\nu_k^{\mathrm{pos}} \le \nu_{\mathrm{good}}^{\mathrm{pos}} = 0.30 \text{ m};$$
4. $$\nu_k^{\theta} \le \nu_{\mathrm{good}}^{\theta} = 0.30 \text{ rad}.$$

The innovation quantities are evaluated before the encoder anchoring correction for the current sample is applied. This prevents the current anchoring operation from artificially reducing the disagreement used to determine whether that same ICP sample is trustworthy.

A consecutive-good counter $c_k$ is then defined as

$$c_k = \begin{cases} c_{k-1} + 1, & \text{if the current ICP result satisfies all good conditions,} \\ 0, & \text{otherwise.} \end{cases}$$

Anchoring becomes eligible only when

$$c_k \geq N_{\text{good}},$$

with

$$N_{\text{good}} = 20.$$

At $\Delta t = 0.1$ s, this corresponds to approximately 2 s of consecutive high-quality ICP samples before anchoring is allowed.

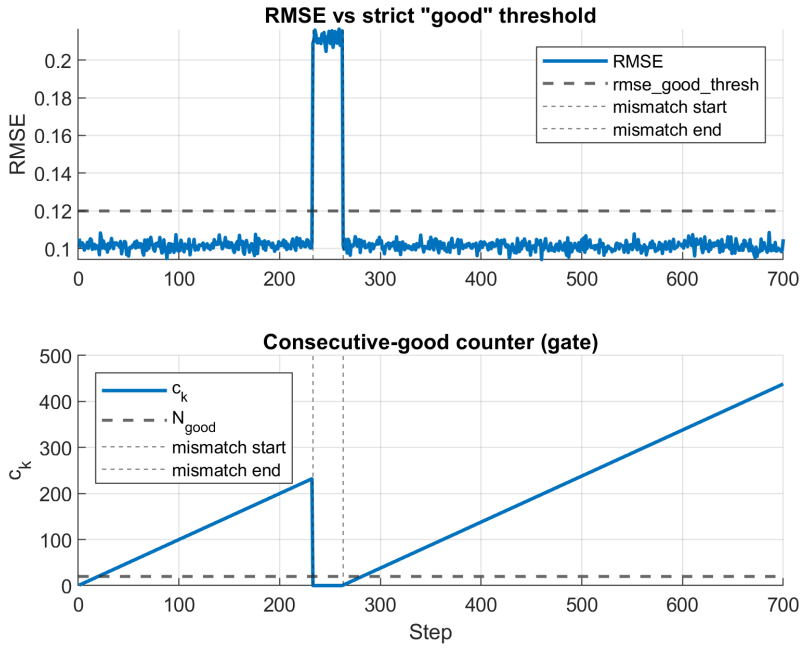



Figure 3.8: Diagnostics for the sustained-good ICP gate. (Top) ICP RMSE compared with the good-RMSE threshold. (Bottom) consecutive-good counter $c_k$ compared with $N_{\text{good}}$. A sample contributes to the counter only when the ICP result is finite and the RMSE, position innovation, and yaw innovation satisfy their respective limits.

### 3.4.3 Gradual Encoder Correction Toward ICP

Once the sustained-good gate is active, the encoder pose used inside FUZZY-BLEND is moved gradually toward the *current* valid ICP pose. Let

$$\mathbf{x}_k^{\text{enc}} = \begin{bmatrix} x_k^{\text{enc}} & y_k^{\text{enc}} & \theta_k^{\text{enc}} \end{bmatrix}^T,$$

and

$$\mathbf{x}_k^{\mathrm{ICP}} = \begin{bmatrix} x_k^{\mathrm{ICP}} & y_k^{\mathrm{ICP}} & \theta_k^{\mathrm{ICP}} \end{bmatrix}^T.$$

The encoder-to-ICP discrepancy is

$$\mathbf{e}_k = \begin{bmatrix} x_k^{\mathrm{ICP}} - x_k^{\mathrm{enc}} \\ y_k^{\mathrm{ICP}} - y_k^{\mathrm{enc}} \\ \mathrm{wrap}\big(\theta_k^{\mathrm{ICP}} - \theta_k^{\mathrm{enc}}\big) \end{bmatrix}.$$

A small fraction of this discrepancy is applied:

$$\Delta\mathbf{x}_k^{\mathrm{enc}} = \kappa_k \mathbf{e}_k,$$

followed by

$$\mathbf{x}_k^{\mathrm{enc}} \leftarrow \mathbf{x}_k^{\mathrm{enc}} + \Delta\mathbf{x}_k^{\mathrm{enc}}.$$

The effective gain is

$$\kappa_k = \kappa_0 \, w(\mathrm{RMSE}_k) \, r(c_k),$$

where

$$\kappa_0 = 0.02.$$

The RMSE-dependent weighting term is

$$w(\mathrm{RMSE}_k) = \begin{cases} 1, & \mathrm{RMSE}_k \le \mathrm{RMSE}_0, \\ 1 - s_k, & \mathrm{RMSE}_0 < \mathrm{RMSE}_k < \mathrm{RMSE}_{\mathrm{good}}, \\ 0, & \mathrm{RMSE}_k \ge \mathrm{RMSE}_{\mathrm{good}}, \end{cases}$$

where

$$s_k = u_k^2(3 - 2u_k), \qquad u_k = \frac{\mathrm{RMSE}_k - \mathrm{RMSE}_0}{\mathrm{RMSE}_{\mathrm{good}} - \mathrm{RMSE}_0},$$

with

$$\mathrm{RMSE}_0 = 0.10 \text{ m}, \qquad \mathrm{RMSE}_{\mathrm{good}} = 0.12 \text{ m}.$$

The consecutive-good ramp is

$$r(c_k) = \min\left(1, \frac{c_k - N_{\mathrm{good}} + 1}{N_{\mathrm{good}}}\right),$$

with $r(c_k) = 0$ whenever $c_k < N_{\mathrm{good}}$.

Finally, the component-wise correction increments are bounded:

$$\Delta x_k \in [-0.03, 0.03] \text{ m}, \qquad \Delta y_k \in [-0.03, 0.03] \text{ m},$$

and

$$\Delta \theta_k \in [-0.03, 0.03] \text{ rad}.$$

These limits make the operation a gradual re-centering rather than a hard reset.

### 3.4.4 Interaction With the Controller and EKF

Encoder anchoring does not overwrite the EKF state. It modifies only the encoder pose maintained inside FUZZY-BLEND after the current ICP measurement has passed the sustained-good checks. The EKF continues to operate through its own prediction and measurement-correction equations.

This separation has three practical consequences:

- The standalone ENC result remains an uncorrected encoder-only baseline and therefore continues to exhibit normal accumulated odometric drift.
- Within FUZZY-BLEND, the encoder state can be gradually re-centered during sustained reliable ICP.
- If fuzzy trust later decreases, the controller can place more weight on an encoder state that has previously been kept closer to the map-referenced solution.

## 3.5 Quantitative Evaluation Metrics

All reported metrics are computed from the trajectories logged by the MATLAB simulation. Because each estimator is used as feedback in its own closed-loop run, each estimate is compared with the ground-truth trajectory generated in that same run.

### 3.5.1 Pose Error Definitions

Let

$$\mathbf{x}_k^{\mathrm{GT}} = [x_k^{\mathrm{GT}}, y_k^{\mathrm{GT}}, \theta_k^{\mathrm{GT}}]^\top$$

denote the simulation ground-truth pose for a given run, and let

$$\hat{\mathbf{x}}_k = [\hat{x}_k, \hat{y}_k, \hat{\theta}_k]^\top$$

denote the corresponding localization estimate.

**Position error.**

$$e_k^{\text{pos}} = \sqrt{(\hat{x}_k - x_k^{\text{GT}})^2 + (\hat{y}_k - y_k^{\text{GT}})^2}. \tag{3.3}$$

**Yaw error.**

$$e_k^{\theta} = \left| \text{wrap}\left(\hat{\theta}_k - \theta_k^{\text{GT}}\right) \right|. \tag{3.4}$$

Because localization is used in the control loop, estimator errors also affect the actual robot trajectory. For visual closed-loop comparison, the ground-truth trajectories are therefore overlaid with the desired path in Section 3.6.1.

### 3.5.2 Aggregate Metrics

The full-trajectory position and yaw RMSE values are

$$\text{RMSE}_{\text{pos}} = \sqrt{\frac{1}{N}\sum_{k=1}^{N} (e_k^{\text{pos}})^2}, \qquad \text{RMSE}_{\theta} = \sqrt{\frac{1}{N}\sum_{k=1}^{N} \left(e_k^{\theta}\right)^2}. \tag{3.5}$$

The 95th-percentile position error is

$$e_{95}^{\text{pos}} = \text{percentile}_{95}\left(\{e_k^{\text{pos}}\}_{k=1}^{N}\right), \tag{3.6}$$

and the maximum position error is

$$e_{\max}^{\text{pos}} = \max_k e_k^{\text{pos}}. \tag{3.7}$$

The 95th-percentile and maximum metrics complement RMSE by distinguishing typical behavior from rare large failures.

### 3.5.3 Mismatch-Window Metrics

Because the temporary map mismatch occurs over a localized interval, position RMSE is also computed over three windows:

$$\mathcal{K}_{\text{pre}} = \{1, \ldots, k_{\text{start}} - 1\}, \tag{3.8}$$

$$\mathcal{K}_{\text{mid}} = \{k_{\text{start}}, \ldots, k_{\text{end}}\}, \tag{3.9}$$

and

$$\mathcal{K}_{\text{post}} = \{k_{\text{end}} + 1, \ldots, N\}. \tag{3.10}$$

This decomposition distinguishes nominal pre-mismatch behavior, estimation performance during the temporary obstacle interval, and post-mismatch recovery.

### 3.5.4 Encoder Anchoring Diagnostics

The encoder-anchoring mechanism is evaluated using an ablation comparison between FUZZY-BLEND with and without anchoring.

For samples with a current finite ICP pose, the encoder–ICP position and yaw gaps are

$$g_k^{\text{pos}} = \left\| \mathbf{p}_k^{\text{ICP}} - \mathbf{p}_k^{\text{enc}} \right\|_2, \tag{3.11}$$

$$g_k^{\theta} = \left| \text{wrap}\left(\theta_k^{\text{ICP}} - \theta_k^{\text{enc}}\right) \right|. \tag{3.12}$$

If the current ICP result is unavailable, the corresponding diagnostic sample is left undefined rather than being computed from a stale ICP pose.

The applied per-step correction magnitudes are

$$c_k^{\text{pos}} = \sqrt{(\Delta x_k)^2 + (\Delta y_k)^2}, \tag{3.13}$$

and

$$c_k^{\theta} = |\Delta \theta_k|. \tag{3.14}$$

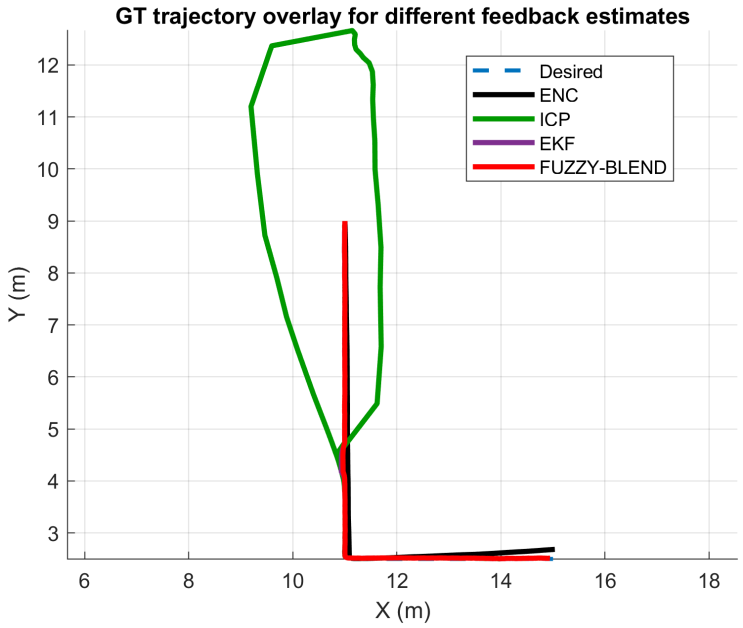
Position correction is therefore reported in metres and yaw correction in radians.

## 3.6 Results and Discussion

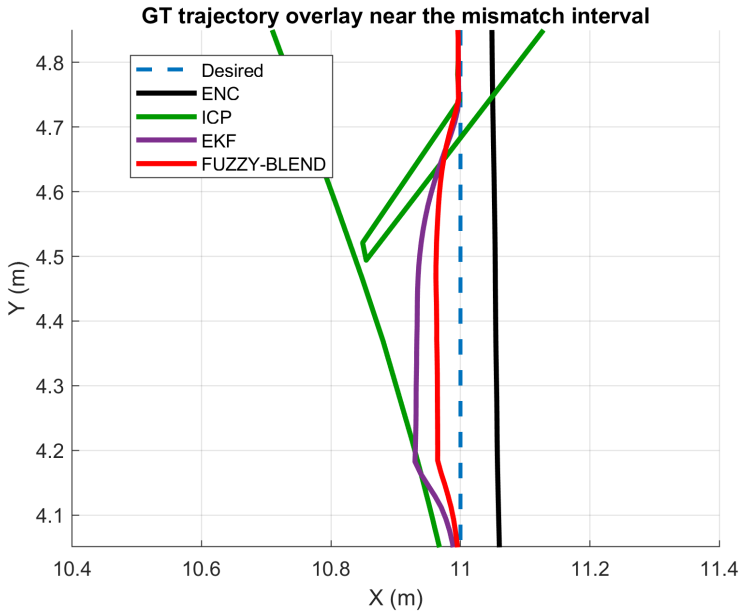
The results are interpreted in terms of three questions: how the methods behave in closed loop, how the reliability mechanisms respond during map mismatch, and whether sustained-good anchoring reduces encoder disagreement without contaminating the raw ENC baseline.

### 3.6.1 Trajectory Overlays and Closed-Loop Behavior

Figure 3.9 compares the ground-truth robot trajectories produced under the four feedback configurations. These curves are the actual simulated robot motions, not the estimator trajectories themselves. Because localization is used by the controller, different feedback estimates produce different closed-loop paths even though the same desired trajectory is supplied to each run.




(a) Full trajectory overlay.




(b) Zoomed view near the mismatch interval.

Figure 3.9: Ground-truth robot trajectories under ENC, ICP, EKF, and FUZZY-BLEND localization feedback. The standalone ENC run is a raw encoder-only baseline and is not ICP-anchored. The zoomed view highlights the region affected by the temporary scan-to-map mismatch.

The black ENC trajectory therefore continues to show accumulated drift, particularly later in the run. This is expected and is necessary for a meaningful encoder-only baseline. Encoder anchoring is evaluated separately inside FUZZY-BLEND and should not be interpreted as a correction applied to the ENC baseline.

### 3.6.2 ICP Quality and EKF Weighting During Map Mismatch

Figure 3.10 shows the ICP registration RMSE and the corresponding covariance-scaling factor $\alpha$ from the EKF run.

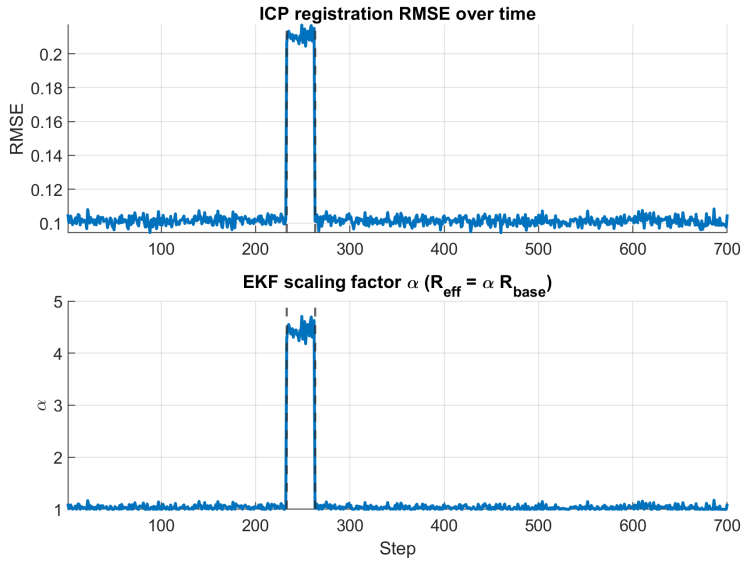



Figure 3.10: ICP reliability diagnostics from the EKF run. (Top) ICP registration RMSE. (Bottom) covariance-scaling factor $\alpha$, where $\mathbf{R}_{\text{eff}} = \alpha \mathbf{R}_{\text{base}}$. The vertical lines indicate the temporary mismatch interval.

During the mismatch interval, scan-to-map consistency degrades because the live LiDAR scan contains the temporary obstacle while the fixed reference point cloud does not. For finite measurements, an increase in RMSE increases $\alpha$, and sufficiently large innovations enforce an additional lower bound on $\alpha$. The resulting larger effective measurement covariance reduces the influence of the ICP measurement in the EKF correction. If ICP is invalid or non-finite, the correction is skipped altogether rather than attempting to use the invalid pose with a large covariance.

### 3.6.3 Impact of Fuzzy Trust on Estimation Error

Figure 3.11 compares estimator errors against GT and shows the filtered FUZZY-BLEND trust signal.

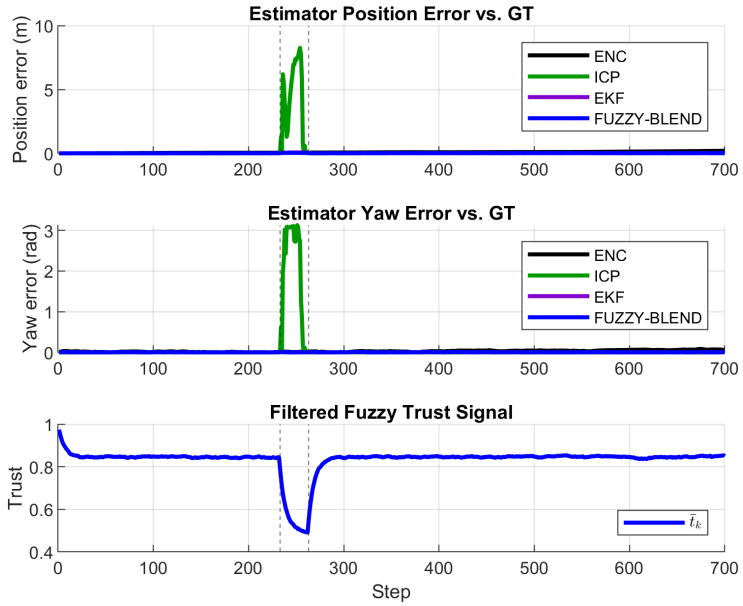



Figure 3.11: Estimator error versus GT over the full run. (Top) position error, (middle) yaw error, and (bottom) FUZZY-BLEND filtered trust. Reduced trust shifts controller-level weight away from ICP and toward encoder feedback while the EKF independently applies its covariance-scaling logic.

The ICP run is accurate through most nominal samples but exhibits severe finite scan-matching failures during the temporary mismatch interval. EKF and FUZZY-BLEND remain bounded in the same interval because the ICP measurement is prevented from dominating the state or controller feedback when the reliability indicators become unfavorable.

The mechanisms should be interpreted separately. The EKF response arises from covariance scaling and invalid-measurement rejection. The additional FUZZY-BLEND response arises from controller-level weighting based on the fuzzy trust signal.

### 3.6.4 Encoder Drift Correction During Sustained-Good ICP

Figure 3.7 compares the encoder–ICP disagreement inside FUZZY-BLEND with anchoring enabled and disabled. This ablation directly evaluates the effect of the encoder-anchoring mechanism without redefining the standalone ENC baseline.

Figure 3.12 further shows the position and yaw gaps together with the corresponding correction magnitudes.

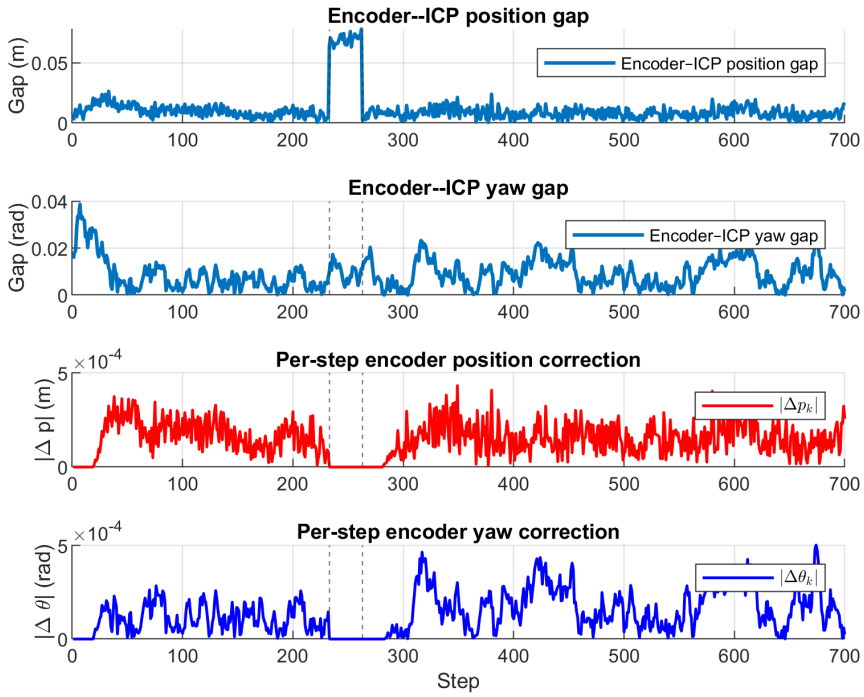



Figure 3.12: Encoder anchoring diagnostics within FUZZY-BLEND. From top to bottom: encoder–ICP position gap (m), encoder–ICP yaw gap (rad), applied encoder position correction per step (m), and applied encoder yaw correction per step (rad). Position and angular quantities are shown separately to avoid mixing units.

When the sustained-good gate is inactive, no encoder anchoring is applied. After the required number of consecutive reliable ICP samples has been reached, the correction activates gradually and remains bounded by the component-wise limits. The desired behavior is therefore not an instantaneous encoder reset but a slow reduction of accumulated disagreement.

### 3.6.5 Summary of Quantitative Results

Tables 3.3 and 3.4 report the results generated by the corrected implementation in which the standalone ENC run is not ICP-anchored.

Table 3.3: Overall estimator accuracy over the full trajectory. Each estimate is evaluated against the GT pose from its corresponding closed-loop run.

| Method | RMSE Pos (m) | RMSE Yaw (rad) | $e_{95}^{\mathrm{pos}}$ (m) | $e_{\max}^{\mathrm{pos}}$ (m) |
|---|---|---|---|---|
| ENC | 0.0939 | 0.0359 | 0.1737 | 0.2010 |
| ICP | 1.0427 | 0.4785 | 0.0273 | 8.2542 |
| EKF | 0.0192 | 0.0045 | 0.0230 | 0.0729 |
| FUZZY-BLEND | 0.0159 | 0.0042 | 0.0245 | 0.0406 |

Table 3.4: Position RMSE before, during, and after the temporary map-mismatch interval.

| Method | Pre $\mathcal{K}_{\mathrm{pre}}$ | Mid $\mathcal{K}_{\mathrm{mid}}$ | Post $\mathcal{K}_{\mathrm{post}}$ |
|---|---|---|---|
| ENC | 0.0353 | 0.0539 | 0.1152 |
| ICP | 0.0087 | 4.9542 | 0.0175 |
| EKF | 0.0075 | 0.0629 | 0.0167 |
| FUZZY-BLEND | 0.0069 | 0.0367 | 0.0169 |

Several observations follow from these results.

First, the raw encoder baseline now exhibits the expected accumulated drift. Its position RMSE rises from 0.0353 m before the mismatch window to 0.1152 m afterward, and its full-trajectory position RMSE is 0.0939 m. This behavior is consistent with the role of ENC as an uncorrected dead-reckoning baseline.

Second, ICP has low error through most nominal samples but fails severely during a short subset of the mismatch interval. Its 95th-percentile position error is only 0.0273 m, while its maximum position error reaches 8.2542 m and its mismatch-window position RMSE reaches 4.9542 m. The large difference between the percentile and worst-case metrics indicates that the dominant weakness in this experiment is a small number of very large finite registration failures rather than uniformly poor ICP accuracy.

Third, the reliability-aware EKF remains bounded during mismatch. Its mismatch-window position RMSE is 0.0629 m and its full-trajectory position RMSE is 0.0192 m, illustrating the benefit of reducing the influence of questionable ICP measurements rather than accepting them directly.

Fourth, FUZZY-BLEND achieves a mismatch-window position RMSE of 0.0367 m, lower than the corresponding values for the raw ENC baseline (0.0539 m) and EKF (0.0629 m) in this experiment. Its full-trajectory position and yaw RMSE values are 0.0159 m and 0.0042 rad, respectively, and its maximum position error is 0.0406 m.

The results should not, however, be summarized as FUZZY-BLEND being superior in every reported metric. For example, EKF has a slightly lower 95th-percentile position error (0.0230 m compared with 0.0245 m) and a slightly lower post-mismatch position RMSE (0.0167 m compared with 0.0169 m). The evidence instead supports the narrower conclusion that, for the tested mismatch scenario, FUZZY-BLEND reduces the mismatch-window error relative to both the raw encoder baseline and the adaptive EKF while maintaining low overall error.

### 3.6.6 Risks and Practical Considerations

Encoder anchoring introduces an important risk: ICP can occasionally be confident but wrong, particularly in repetitive or symmetric environments. If such an incorrect pose satisfies the anchoring conditions, moving the encoder state toward it can introduce bias.

The implementation therefore uses several safeguards:

- ICP must be finite and valid;
- RMSE, position innovation, and yaw innovation must all satisfy the sustained-good thresholds;
- the conditions must remain satisfied for 20 consecutive samples before anchoring is enabled;
- the anchoring gain ramps gradually after activation; and
- each position and yaw correction increment is bounded.

These safeguards reduce, but do not eliminate, the possibility of anchoring toward a confidently incorrect ICP solution. This limitation is addressed further in Section 3.7.

## 3.7 Limitations and Threats to Validity

The results provide simulation-based evidence for the proposed reliability mechanisms under the tested conditions, but several limitations should be considered.

- **Idealized map and LiDAR environment.** The reference map is a binary occupancy representation and ICP operates on a point cloud extracted from occupied cells. Real warehouse LiDAR data can be affected by reflective materials, glass, moving objects, partial returns, and other effects not reproduced here.
- **ICP confidence is not equivalent to correctness.** Low RMSE and small innovation do not guarantee globally correct registration. Repetitive aisle structures can still produce locally plausible but incorrect alignments.
- **Empirical tuning.** The FIS breakpoints, covariance-scaling thresholds, sustained-good thresholds, and anchoring gains were selected for the present simulation and may require retuning in other environments or on physical hardware.
- **Innovation definition.** Euclidean position innovation and wrapped yaw innovation are used for transparency and straightforward fuzzy-rule design. A covariance-normalized metric such as Mahalanobis distance could provide a more statistically grounded consistency test in future work.
- **Closed-loop coupling.** Each estimator is used as feedback in its own run. Consequently, localization errors alter robot motion, and the resulting GT trajectories are not identical across methods. The reported values therefore represent closed-loop localization-and-control behavior rather than a purely offline estimator comparison on one common trajectory.
- **Single seeded realization.** A fixed random seed is used for reproducibility. The reported numerical values characterize that reproducible realization and are not a statistical characterization across many independent noise seeds.
- **Simulation-only validation of the complete reliability-aware method.** The full fuzzy blending and encoder-anchoring logic is evaluated here in MATLAB simulation. Physical-robot validation and systematic sensitivity analysis remain future work.

## 3.8 Chapter Summary

This chapter developed and evaluated a reliability-aware localization architecture for two complementary problems: temporary scan-to-map mismatch and accumulated encoder drift.

Three mechanisms were kept distinct. First, finite but unreliable ICP measurements were down-weighted inside the EKF through reliability-dependent measurement covariance, while invalid or non-finite ICP results were skipped entirely. Second, a Mamdani FIS combined ICP RMSE, position innovation, and yaw innovation into a filtered trust score that controlled controller-level blending among encoder, EKF, and ICP estimates. Third, sustained-good validity, RMSE, and innovation conditions enabled gradual encoder anchoring within FUZZY-BLEND only.

The corrected raw ENC baseline exhibited the expected long-term drift, with post-mismatch position RMSE increasing to 0.1152 m. ICP remained accurate during most nominal samples but experienced severe finite failures during the temporary map mismatch, reaching 4.9542 m position RMSE within the mismatch window. The adaptive EKF reduced this effect to 0.0629 m, while FUZZY-BLEND produced 0.0367 m mismatch-window position RMSE. Over the full run, FUZZY-BLEND produced 0.0159 m position RMSE and 0.0042 rad yaw RMSE.

These results do not indicate that one method is lowest in every reported statistic. Rather, they show that, in the tested mismatch scenario, the proposed reliability-aware combination reduced the effect of severe ICP failures while maintaining low overall estimation error, and that sustained reliable ICP can be used conservatively to limit encoder drift inside the blended architecture. The next step is to transfer these reliability mechanisms to a real-time ROS 2 implementation and evaluate them under physical sensor, timing, and environmental uncertainties.

# Chapter 4

# ROS 2-Based Implementation in Gazebo and RViz

## 4.1 Introduction and Objectives

After developing and testing the localization methods in MATLAB in Chapter 2 and introducing the reliability-aware framework in Chapter 3, the next stage was to establish a robotic software implementation that could support future testing on a physical robot. This chapter presents the initial integration and simulation evaluation of the principal localization components using the **ROS 2** framework [46], **Gazebo** [34] for simulation, and **RViz** [30] for visualization.

MATLAB provided a controlled environment for developing the mathematical models and evaluating the reliability-aware method. Moving toward physical-robot experiments also requires a software architecture that handles sensor messages, coordinate transforms, different update rates, and communication between processing nodes. The ROS 2/Gazebo implementation was therefore developed as an intermediate step toward hardware testing, with emphasis on integrating the main components and evaluating their operation within a common robotic software framework.

The objectives of this chapter are:

1. to integrate wheel odometry, planar LiDAR processing, local ICP odometry, EKF-based estimation, and visualization in a Gazebo warehouse environment;
2. to generate a reusable occupancy-grid map with SLAM Toolbox and perform map-referenced localization with AMCL;

3. to implement a lightweight online ICP trust monitor using available scan-matching diagnostics; and

4. to record the simulated sensor and localization outputs for synchronization, error analysis, and plotting in MATLAB.

The implementation introduces system-level considerations that are less prominent in the synchronous MATLAB simulations, including asynchronous message passing, transform lookup, node scheduling, and different sensor-update rates. Addressing these considerations provides a practical foundation for later integration with physical sensors and robot hardware.

The complete reliability-aware framework developed in Chapter 3 was not implemented in ROS 2 during this stage. In the MATLAB framework, fuzzy trust determines the controller-feedback blending weights, RMSE and innovation checks separately scale the EKF measurement covariance, and sustained-quality conditions enable gradual encoder anchoring. These mechanisms remain part of the MATLAB evaluation. In ROS 2, the custom `icp_trust_monitor` instead produces a heuristic diagnostic score from RTAB-Map odometry information. This score is recorded and visualized but does not modify the EKF measurement weighting or apply encoder correction.

Accordingly, the results in this chapter assess the integration and localization performance of the supporting ROS 2 components under the tested Gazebo conditions. They establish an initial software foundation for future implementation of the complete reliability-aware method and subsequent testing on a physical robot. Hardware validation and full integration of the Chapter 3 mechanisms remain future work.

The remainder of this chapter describes the operational modes, ROS 2 architecture, workspace organization, robot and sensor models, map-generation procedure, local ICP odometry, diagnostic trust monitor, AMCL localization, EKF configuration, evaluation procedure, results, and limitations.

## 4.2 Operational Modes and Experimental Workflow

The ROS 2 workspace was organized as a reusable framework with three principal operating modes:

1. **Odometry and fusion mode:** Gazebo, the robot model, wheel odometry, planar LiDAR processing, local ICP odometry, EKF estimation, diagnostic nodes, and RViz were operated together through coordinated launch files.

2. **Mapping mode:** the robot was driven manually through the warehouse with keyboard teleoperation while SLAM Toolbox constructed the occupancy-grid map.

3. **Map-localization mode:** AMCL localized the robot against the saved occupancy-grid map.

All ROS 2 nodes used Gazebo simulation time. Keyboard teleoperation was used during map generation and preliminary system testing. During the final evaluation runs, robot motion was generated by the automated trajectory launch. Relevant pose, odometry, scan-matching, and Gazebo-state topics were recorded in ROS 2 bags. For the final scripted evaluations, the trajectory launch automatically published the known initial AMCL pose at the map-frame origin before sending motion commands. Consequently, manual initialization through RViz was not required. This experiment therefore evaluated localization from a known initial condition rather than global localization from an unknown pose. The recorded data were subsequently imported into MATLAB for synchronization, error calculation, and plotting so that the presentation of the ROS 2 results remained consistent with the figures used elsewhere in the thesis.

Table 4.1: Summary of ROS 2 operational modes used in the simulation.

| **Mode** | **Launch Entry** | **Main Packages/Nodes** | **Purpose** |
|---|---|---|---|
| Odometry / Fusion | `display.launch.py`, `icp.launch.py` | Gazebo, LiDAR conversion, RTAB-Map ICP odometry, EKF, RViz | Odometry, diagnostics, and fusion evaluation |
| Mapping | `online_async_launch.py` | SLAM Toolbox [44], keyboard teleoperation | Manual warehouse coverage, occupancy-grid construction, loop closure, and map saving |
| Map Localization | `amcl_only.launch.py` | AMCL [24], Nav2 map server | Localization on the saved occupancy-grid map |

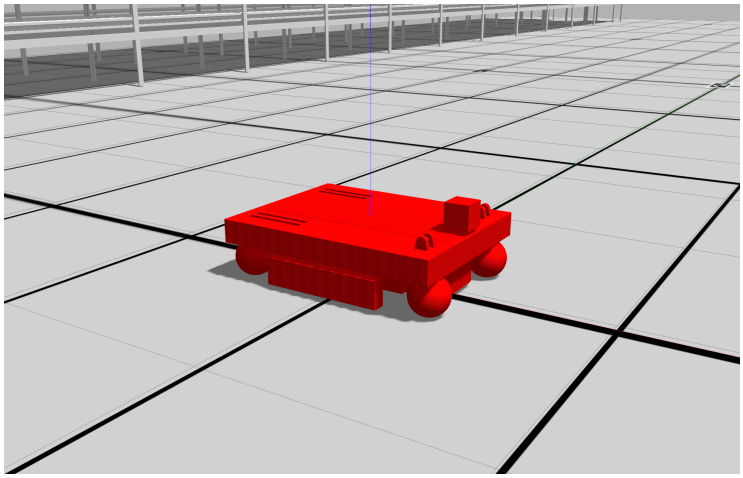


Figure 4.1: AGV5 robot model in Gazebo, showing the differential-drive base, passive caster wheels, and front-mounted planar LiDAR sensor.

## 4.3 System Architecture Overview

The localization system was implemented as a set of ROS 2 nodes connected through standard topics and TF transforms. Gazebo supplied the robot motion and simulated sensor data. ROS 2 nodes then generated local odometry, map-referenced localization, diagnostic signals, filtered estimates, and visualization outputs.

### 4.3.1 Overall Data Flow

The principal data flow was:

- **Gazebo simulation:** the differential-drive plugin published encoder-integrated wheel odometry on `/wheel_odom`;
- **LiDAR simulation:** a horizontal Gazebo ray sensor published planar range returns in a `sensor_msgs/PointCloud2` message on `/ray/lidar_points`;
- **LiDAR conversion:** `pointcloud_to_laserscan` converted these returns to a `sensor_msgs/LaserScan` message on `/scan`;
- **Local ICP odometry:** RTAB-Map ICP odometry estimated local motion and published `/icp_odom`;

- **ICP diagnostics:** RTAB-Map published registration information on `/odom_info`;
- **Local trust monitoring:** the custom monitor converted selected ICP diagnostics into a smoothed heuristic score on `/icp_trust`;
- **EKF estimation:** the `robot_localization` EKF produced `/odometry/filtered`;
- **Map localization:** in the separate AMCL mode, the saved map and live `/scan` data produced `/amcl_pose` and the `map` → `odom` transform; and
- **Visualization and logging:** RViz displayed the robot, scans, paths, map, and transforms, while ROS 2 bags retained the data for MATLAB post-processing.

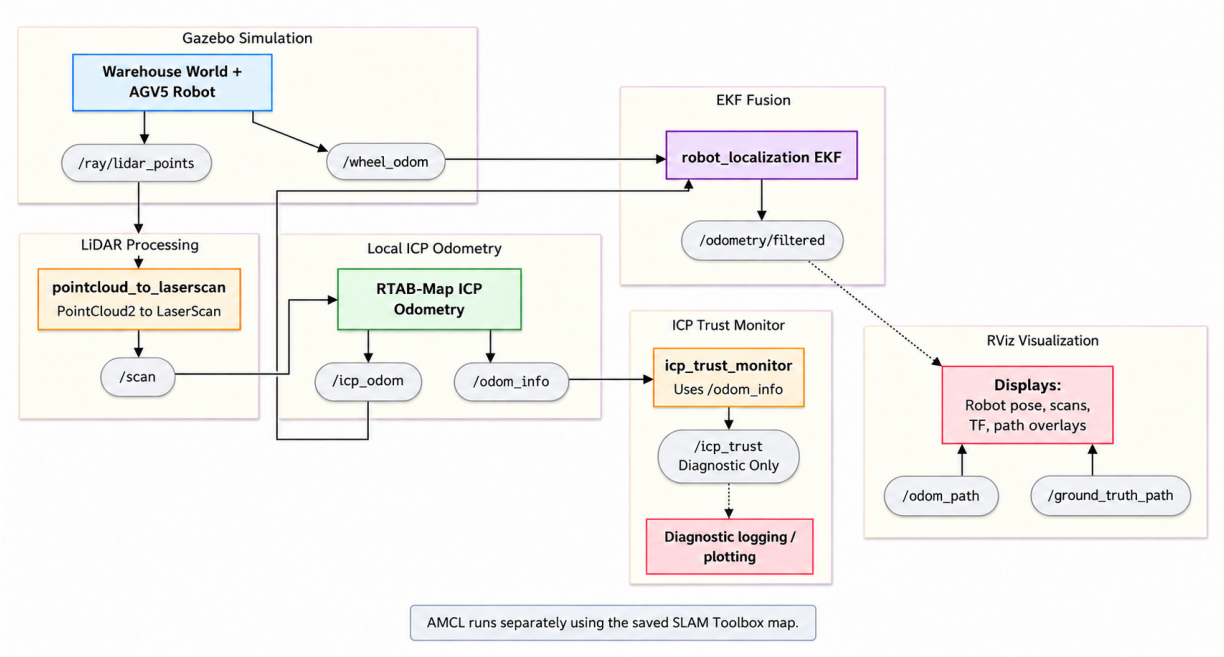



Figure 4.2: High-level ROS 2 localization architecture. Gazebo supplies wheel odometry and planar LiDAR returns; RTAB-Map provides local ICP odometry and diagnostics; the EKF provides a filtered estimate; and the custom monitor publishes a diagnostic ICP trust score. AMCL operates separately against the saved SLAM Toolbox map.

Table 4.2: Major ROS 2 topics and their roles in the Gazebo implementation.

| Topic | Message Type | Publisher | Principal Consumer |
|---|---|---|---|
| /wheel_odom | nav_msgs/Odometry | Gazebo differential-drive plugin | EKF, logging, path visualization |
| /ray/lidar_points | sensor_msgs/PointCloud2 | Gazebo ray-sensor plugin | PointCloud-to-LaserScan node |
| /scan | sensor_msgs/LaserScan | PointCloud-to-LaserScan node | ICP odometry, AMCL, SLAM Toolbox |
| /icp_odom | nav_msgs/Odometry | RTAB-Map ICP odometry | EKF, logging, RViz |
| /odom_info | rtabmap_msgs/OdomInfo | RTAB-Map ICP odometry | ICP diagnostic logger and trust monitor |
| /icp_trust | std_msgs/Float32 | ICP trust monitor | Diagnostic logging and visualization |
| /odometry/filtered | nav_msgs/Odometry | EKF node | Logging, RViz, path visualization |
| /amcl_pose | geometry_msgs/PoseWithCovarianceStamped | AMCL | Logging and RViz |
| /map | nav_msgs/OccupancyGrid | SLAM Toolbox or Nav2 map server | RViz and AMCL |
| /model_states_demo | gazebo_msgs/ModelStates | Gazebo state plugin | Ground-truth logging and visualization |

### 4.3.2 Key Topics and Data Interfaces

### 4.3.3 TF Frame Hierarchy

The frame hierarchy used in the cleaned implementation is

$$\texttt{map} \rightarrow \texttt{odom} \rightarrow \texttt{base_footprint} \rightarrow \texttt{base_link} \rightarrow \texttt{lidar_front}.$$

The URDF/Xacro model defines the fixed transforms from `base_footprint` to `base_link` and from `base_link` to `lidar_front`. The EKF owns the $\texttt{odom} \rightarrow \texttt{base_footprint}$ transform. During map localization, AMCL publishes $\texttt{map} \rightarrow \texttt{odom}$. The RTAB-Map ICP node publishes its odometry message on `/icp_odom`, but its TF publication is disabled so that two nodes do not compete for the same odometry transform.

## 4.4 Package Structure and Workspace Organization

The workspace follows the standard ROS 2 `colcon` organization and was developed for ROS 2 Humble.

Several small custom nodes supported visualization and data collection. These included odometry-to-path publishers, Gazebo model/link-state loggers, an odometry logger, an ICP

Table 4.3: Main packages and their roles in the localization framework.

| Package | Role |
|---|---|
| `robot_description` | URDF/Xacro model, Gazebo world, launch files, maps, EKF and localization parameters, and RViz configuration |
| `robot_path_publisher` | Path visualization, Gazebo-state logging, odometry logging, ICP diagnostic logging, and trust-monitor utilities |
| `rtabmap_odom` | Local ICP LiDAR odometry and `/odom_info` diagnostics |
| `slam_toolbox` | Occupancy-grid mapping and loop closure |
| `nav2_map_server` and `nav2_amcl` | Saved-map publication and AMCL localization |
| `robot_localization` | EKF-based state estimation |

information logger, and the `icp_trust_monitor`. Additional experimental utilities, such as a noisy-odometry publisher, were retained in the workspace but were not required for the principal localization comparison reported here.

Table 4.4: Principal launch and configuration files in the ROS 2 workspace.

| File | Purpose |
|---|---|
| `display.launch.py` | Starts Gazebo, spawns the robot, publishes the robot model, converts LiDAR data, optionally starts the EKF and path nodes, and opens RViz |
| `icp.launch.py` | Starts RTAB-Map ICP odometry using `icp_params_file.yaml` |
| `online_async_launch.py` | Starts SLAM Toolbox for occupancy-grid mapping |
| `amcl_only.launch.py` | Starts the Nav2 map server, AMCL, and their lifecycle manager |
| `ekf.yaml` | Configures the `robot_localization` EKF |
| `robot.urdf.xacro` | Defines robot geometry, frames, joints, simulated sensor, and Gazebo plugins |
| `catalyst_env.world` | Defines the Gazebo warehouse environment and Gazebo-state publisher |
| `config.rviz` | Stores the RViz visualization layout |

## 4.5 Robot Description and Simulation Environment

### 4.5.1 Robot Model

The mobile robot, referred to as **AGV5**, was represented in URDF/Xacro as a differential-drive platform with two active drive wheels and four passive caster wheels. The model also defines `base_footprint`, `base_link`, the wheel links and joints, and the front-mounted `lidar_front` link.

The Gazebo differential-drive plugin publishes `/wheel_odom`. For the encoder-odometry experiments, its odometry source is the wheel encoder integration rather than the Gazebo world pose. Consequently, `/wheel_odom` behaves as local dead reckoning and can accumulate error relative to Gazebo ground truth.

### 4.5.2 Planar LiDAR Model

The simulated sensor is a horizontal ray scanner with a full 360° field of view. Although the Gazebo ROS plugin transports its returns in a `PointCloud2` message, the configured ray model contains one horizontal scan plane and should therefore be described as a *planar LiDAR*. The point cloud is converted to `/scan` for SLAM Toolbox, AMCL, and RTAB-Map ICP odometry.

This distinction is important: the message type alone does not make the sensor three-dimensional, and the implementation does not use vertical LiDAR structure. The principal scan parameters, including range limits, angular sampling, and update rate, are defined in the robot and launch configurations.

### 4.5.3 Gazebo Warehouse Environment

The robot operates in the warehouse-like environment defined by `catalyst_env.world`. Shelves, walls, corners, and long aisles provide both well-constrained regions and repetitive corridor geometry. This makes the environment relevant to the localization problems considered in Chapters 2 and 3.

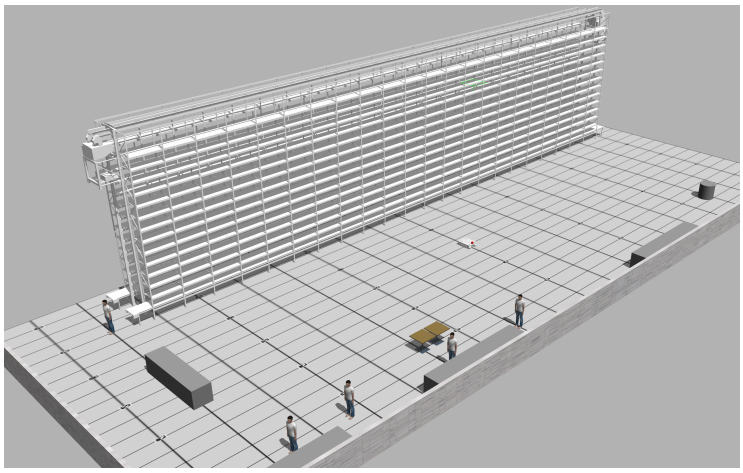


Figure 4.3: Gazebo warehouse environment containing shelves, open regions, and corridor-like structures used in the ROS 2 localization experiments.

## 4.6 Occupancy-Grid Mapping with SLAM Toolbox

The static occupancy-grid map used for AMCL was generated with SLAM Toolbox [44]. During the mapping run, the robot was driven manually through the Gazebo warehouse using keyboard teleoperation. The robot was moved through the principal aisles and open regions so that the LiDAR observed shelves, walls, corners, and corridor boundaries from multiple poses.

SLAM Toolbox received the planar scan on `/scan` together with the odometry transform and incrementally published the occupancy grid on `/map`. Mapping progress, scan alignment, and loop closure were monitored in RViz. Regions were revisited where necessary to improve coverage and internal consistency.

After mapping was complete, the occupancy grid was saved with

```
ros2 run nav2_map_server map_saver_cli -f final_map
```

which produced `final_map.pgm` and `final_map.yaml`. The YAML file stores the image reference, map resolution, origin, and occupancy thresholds. The map was generated once and then reused for the AMCL experiments; the warehouse was not remapped for every localization run.

SLAM Toolbox was responsible for generating the saved occupancy-grid map. RTAB-Map was used separately for local ICP odometry and did not generate the map used by AMCL.

## 4.7 Local LiDAR Odometry and RTAB-Map Integration

### 4.7.1 Local ICP Odometry

Local LiDAR odometry was implemented with the `rtabmap_odom/icp_odometry` node from RTAB-Map [36]. The node receives `/scan`, performs planar ICP registration against its recent local scan representation, and publishes odometry on `/icp_odom`. The robot frame is `base_link`; the message is expressed relative to a local odometry frame; and RTAB-Map TF publication is disabled because the EKF owns the odometry transform.

This process is local scan registration. It estimates motion from recent LiDAR observations and is distinct from AMCL, which estimates pose against the saved static occupancy grid.

### 4.7.2 Practical Configuration

The final launch remaps the LaserScan interface to `/scan`, the odometry output to `/icp_odom`, and the scan-cloud interface to an unused topic. This avoids ambiguity over which RTAB-Map input is active. The configured ICP registration is constrained to planar motion, point-to-point registration is used, scans are voxel-filtered before registration, and deskewing is disabled for the simulated scan stream. The configuration file records the iteration limit and maximum correspondence distance used for the evaluation.

### 4.7.3 ICP Diagnostic Information

RTAB-Map publishes an `OdomInfo` message on `/odom_info`. Fields used by the diagnostic layer include:

- `lost`, which indicates whether the odometry registration is currently reported as lost;
- `icp_inliers_ratio`, which reports the accepted-correspondence ratio;
- `icp_correspondences`, which reports the number of ICP correspondences;
- `icp_translation` and `icp_rotation`, which describe the magnitude of the ICP refinement relative to its registration guess; and
- the reported covariance and estimation time.

These quantities are useful indicators of local registration behavior, but they are not ground-truth localization errors. In particular, `icp_translation` is not translational RMSE, and the RTAB-Map correction fields are not the same as the EKF innovation signals used by the MATLAB fuzzy system in Chapter 3. Localization RMSE requires comparison with Gazebo ground truth, while an EKF innovation requires a measurement-minus-prediction calculation in a consistent frame and at a consistent timestamp.

### 4.7.4 Online Local ICP Trust Monitor

The custom `icp_trust_monitor` subscribes to `/odom_info` and calculates a heuristic trust score for the local scan-matching process. The score combines the ICP inlier ratio with the translation and rotation correction magnitudes reported by RTAB-Map. Increasing

correction magnitudes progressively reduce the trust score, while the penalties saturate at predefined translation and rotation thresholds. These corrections describe the ICP refinement relative to its registration guess and should not be interpreted as the robot's actual motion between successive scans.

First, a ratio-based baseline trust $q_r$ is determined from the inlier ratio $r$:

$$q_r = \text{clamp}\left(\frac{r - r_{\text{bad}}}{r_{\text{good}} - r_{\text{bad}}}, 0, 1\right). \tag{4.1}$$

Next, normalized correction-magnitude penalties are defined for the reported ICP translation correction ($\Delta d$) and rotation correction ($\Delta\psi$):

$$p_t = \text{clamp}\left(\frac{\Delta d}{d_{\text{bad}}}, 0, 1\right), \quad p_\psi = \text{clamp}\left(\frac{|\Delta\psi|}{\psi_{\text{bad}}}, 0, 1\right). \tag{4.2}$$

The raw trust score is then calculated by applying the penalties to the baseline ratio. If RTAB-Map explicitly reports tracking loss (`lost=true`), the raw trust evaluates to zero immediately:

$$\tau_{\text{raw}} = \begin{cases} 0, & \text{if ICP reports tracking loss,} \\ \text{clamp}(q_r - w_t p_t - w_\psi p_\psi, 0, 1), & \text{otherwise.} \end{cases} \tag{4.3}$$

Finally, to prevent high-frequency chattering in the diagnostic signal, the result is smoothed exponentially:

$$\tau_k = \tau_{k-1} + \beta(\tau_{\text{raw},k} - \tau_{k-1}), \tag{4.4}$$

where $\beta = 0.15$, $w_t = 0.5$, $w_\psi = 0.5$, $d_{\text{bad}} = 0.05$ m, and $\psi_{\text{bad}} = 0.10$ rad. The translation and rotation thresholds specify the correction magnitudes at which their corresponding normalized penalties saturate at one. These values were selected empirically to distinguish the small registration refinements observed during nominal tracking from unusually large ICP corrections.

During initial system testing, it was observed that the nominal inlier ratio in the simulated warehouse environment was approximately 0.24. An initial lower threshold of $r_{\text{bad}} = 0.30$ incorrectly produced very low trust scores even during periods of acceptable tracking. Therefore, an empirical recalibration procedure was performed. By recording deliberately degraded intervals alongside nominal operation and examining the resulting correspondence distributions, the thresholds were adjusted to $r_{\text{bad}} = 0.10$ and $r_{\text{good}} = 0.25$. These thresholds were then frozen before executing the final evaluation runs.

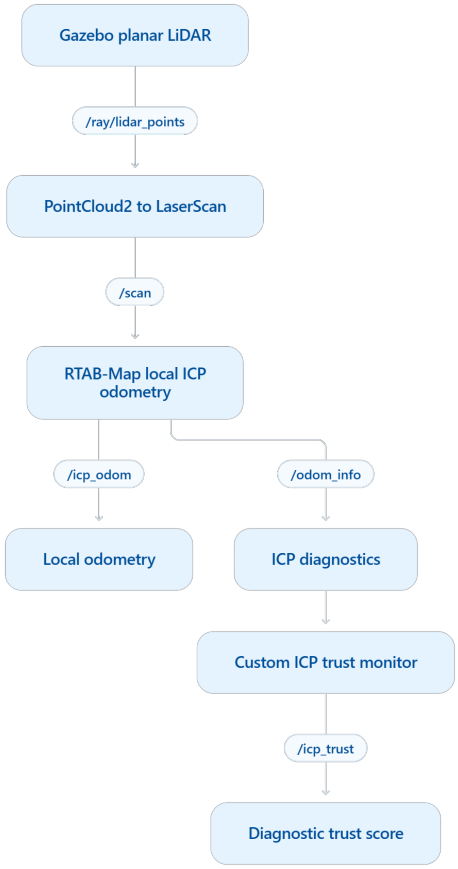



Figure 4.4: Local ICP odometry and diagnostic flow. RTAB-Map processes `/scan`, publishes `/icp_odom` and `/odom_info`, and the custom monitor converts selected diagnostic fields into `/icp_trust`.

## 4.8 AMCL Map Localization

Localization on the saved `final_map` occupancy grid was performed using **Adaptive Monte Carlo Localization (AMCL)** [24] from Navigation2 [45]. AMCL is a particle-filter method: it propagates pose hypotheses using odometric motion and updates their weights using the likelihood of the live LiDAR scan relative to the static map.

The Nav2 map server publishes the saved occupancy grid on `/map`. AMCL subscribes to the map and `/scan`, obtains odometric motion through TF, publishes `/amcl_pose` and

`/particle_cloud`, and supplies the `map` → `odom` transform. As detailed in Section 4.2, a manual *2D Pose Estimate* tool can initialize AMCL; however, the final evaluations utilized an automated trajectory launch script to publish the initial AMCL pose at the map-frame origin to ensure evaluation consistency.

AMCL and local ICP serve different roles. RTAB-Map ICP provides local scan-based motion estimation without a static global map, whereas AMCL provides drift-bounded map-referenced localization.

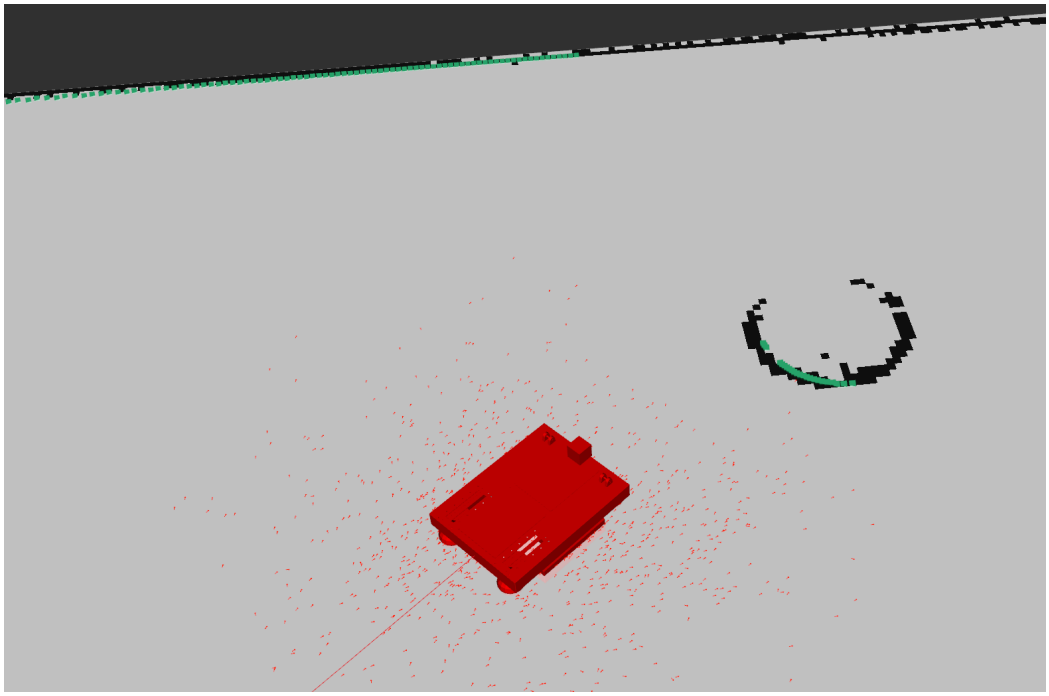


Figure 4.5: AMCL localization in RViz, showing the saved occupancy grid, live LiDAR scan, robot model, and particle-based pose estimate.

## 4.9 EKF-Based State Estimation

The ROS 2 `robot_localization` package [54] was used to implement the EKF estimation layer. The filter operates in two-dimensional mode and publishes `/odometry/filtered` together with the `odom` → `base_footprint` transform.

### 4.9.1 Configuration and State Variables

To prevent filter divergence caused by duplicate measurements, the state inputs were carefully selected:

- **Wheel Odometry (`/wheel_odom`):** Fuses forward velocity ($v_x$) and yaw rate ($\dot{\psi}$). Wheel-integrated pose $(x, y, \psi)$ was explicitly excluded because these are derived from the same underlying encoder measurements.
- **ICP Odometry (`/icp_odom`):** Fuses spatial pose $(x, y, \psi)$. Duplicate velocity quantities derived from the scan matching were omitted.
- **AMCL (`/amcl_pose`):** Maintained independently for map-based localization comparison and excluded from the local EKF fusion entirely.

Therefore, the filter continuously fused wheel velocity and yaw rate with the ICP pose to produce the `/odometry/filtered` estimate.

### 4.9.2 Covariance Calibration

The scaling of the measurement and process-noise covariances is important for EKF performance. Separate calibration runs were used to establish the wheel-odometry covariance and to assess the covariance reported by the ICP odometry source. The calibration procedure was:

1. Wheel odometry, ICP odometry, and Gazebo ground-truth streams were recorded during separate calibration runs.
2. The recorded streams were synchronized using ROS simulation timestamps.
3. Residuals for wheel forward velocity and yaw rate were calculated relative to the corresponding ground-truth quantities.
4. The resulting residual variances were used to configure the covariance fields published with the wheel-odometry measurements.
5. Residuals for the ICP pose components $(x, y, \psi)$ were calculated relative to the aligned Gazebo ground-truth trajectory.

6. These residual statistics were compared with the covariance values dynamically reported by RTAB-Map in the `/icp_odom` messages to verify that their magnitudes were reasonable for the observed registration performance.

7. The wheel-odometry covariance settings and the remaining EKF configuration parameters were then frozen before the final evaluation runs.

The `robot_localization` EKF obtained measurement covariances from the incoming odometry messages. The wheel-odometry covariance was configured using the calibration-derived residual variances, whereas the ICP covariance was generated dynamically by RTAB-Map and included in each `/icp_odom` message. The EKF process-noise covariance and initial-estimate covariance matrices were specified separately in the `ekf.yaml` configuration file and remained fixed throughout the final evaluations.

The trust score on `/icp_trust` was used only as a diagnostic output. It did not replace or modify the covariance values reported by RTAB-Map and therefore did not alter the EKF measurement weighting. Implementing the complete Chapter 3 strategy in ROS 2 would require an intermediary node that modifies the ICP covariance according to the trust score, or a custom EKF implementation. This extension was outside the scope of the present thesis.

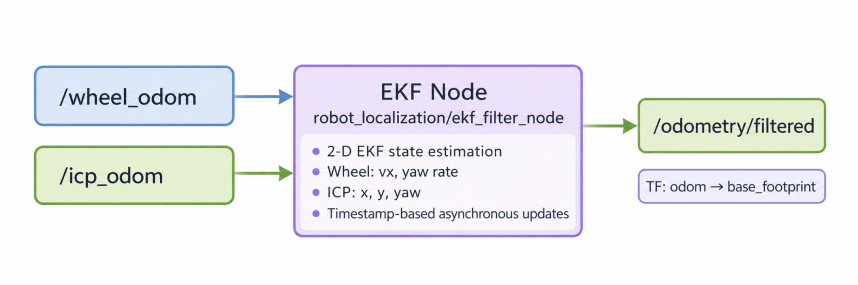



Figure 4.6: ROS 2 topic flow for wheel/ICP EKF estimation. The diagnostic trust output is logged separately and does not alter the EKF measurement weighting.

## 4.10 Results and Performance Analysis

### 4.10.1 Evaluation Procedure

The principal pose streams were wheel odometry (`/wheel_odom`), local ICP odometry (`/icp_odom`), AMCL localization (`/amcl_pose`), and EKF output (`/odometry/filtered`). Gazebo-model or link-state supplied the simulation ground-truth reference. Multiple runs were conducted to assess functional consistency.

Wheel odometry, local ICP odometry, and EKF estimates were recorded simultaneously during the odometry-and-fusion evaluation. AMCL was evaluated separately in the map-localization mode using the same nominal trajectory and initial condition. The results presented in this chapter were selected using predefined completeness criteria, including the availability of the required estimator and ground-truth streams, complete trajectory execution, and the absence of logging interruptions. The evaluations were not selected solely for having the lowest localization errors. Consequently, the AMCL result is included as a cross-mode reference rather than as a strictly simultaneous comparison.

### 4.10.2 MATLAB Post-Processing and Synchronization

The ROS 2 bag data were exported and processed in MATLAB solely for synchronization, evaluation, and plotting. MATLAB did not generate the ROS robot motion or localization estimates reported in this chapter.

Because ROS 2 relies on asynchronous node execution, data synchronization is required to perform accurate error analysis. The implementation relied on Gazebo simulation time via `/clock`, with `use_sim_time: true` enabled for all localization and logging nodes. The respective topic frequencies were approximately 10 Hz for the LiDAR `/scan`, 50 Hz for wheel odometry and the Gazebo state plugin, and 30 Hz for the EKF output.

Synchronization was achieved by first determining a common time interval available across all necessary streams, discarding samples outside this interval. The yaw was unwrapped prior to interpolation to prevent wrapping artifacts. A linear interpolation was applied to $x$ and $y$, while the unwrapped yaw was interpolated linearly and then wrapped back to $[-\pi, \pi]$. Within each evaluation run, the available estimator and ground-truth streams were evaluated on a common time grid, and data were interpolated strictly within each topic's available timestamp interval rather than extrapolated beyond it. Finally, a single planar rigid transform determined from the initial estimate and ground-truth poses

was applied to the complete estimated trajectory to remove arbitrary local frame origin differences without disguising subsequent drift.

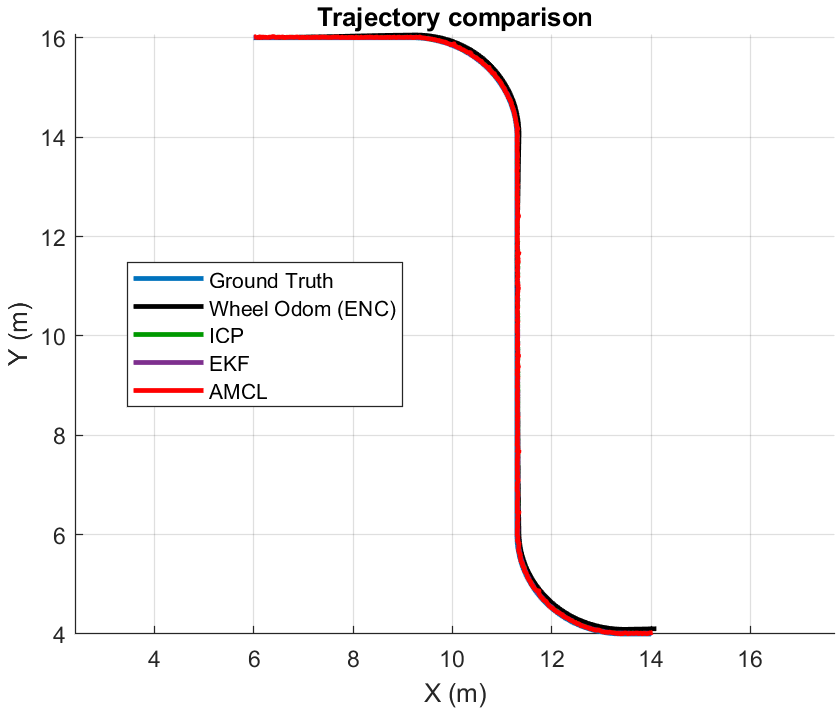



Figure 4.7: Representative ROS 2/Gazebo localization trajectories post-processed and plotted in MATLAB. When methods originate from different operating modes, the overlay is interpreted as a qualitative cross-mode comparison.

### 4.10.3 Interpretation of ICP Diagnostics

The custom `icp_trust_monitor` provides a heuristic diagnostic score based on the registration information published by RTAB-Map on `/odom_info`. A tracking-loss indication, a reduced inlier ratio, or large registration corrections can indicate degraded local scan matching. The monitor combines these indicators into the smoothed score published on `/icp_trust`.

This score should be interpreted as an indicator of local registration quality rather than a direct measure of localization accuracy. A low score flags an interval for further inspection, while a high score does not guarantee an accurate pose. Similarly, `lost=false`

indicates that RTAB-Map has not reported tracking loss; it does not establish that the estimated pose is close to ground truth. Localization accuracy must be assessed separately by comparing the estimated trajectory with the synchronized Gazebo ground-truth trajectory.

In this implementation, the trust score is used only for diagnostic logging and visualization. It does not reject ICP measurements, modify their covariance, or apply encoder correction. The quantitative results in the following subsection therefore evaluate the localization estimates and ICP tracking availability, rather than the effectiveness of trust-based adaptation. A quantitative assessment of the monitor's ability to identify inaccurate ICP estimates remains future work.

### 4.10.4 Quantitative Metrics and ICP Tracking Availability

Let $\mathbf{p}^{\mathrm{est}}(t_k) = [x^{\mathrm{est}}(t_k)\ y^{\mathrm{est}}(t_k)]^{\mathsf{T}}$ and $\mathbf{p}^{\mathrm{GT}}(t_k) = [x^{\mathrm{GT}}(t_k)\ y^{\mathrm{GT}}(t_k)]^{\mathsf{T}}$ denote the aligned, synchronized estimated and ground-truth positions. The translational absolute pose error (APE) is

$$e_{\mathrm{APE},k} = \left\| \mathbf{p}^{\mathrm{est}}(t_k) - \mathbf{p}^{\mathrm{GT}}(t_k) \right\|_2, \tag{4.5}$$

and the reported scalar is

$$\mathrm{APE}_{\mathrm{RMSE}} = \sqrt{\frac{1}{N}\sum_{k=1}^{N} e_{\mathrm{APE},k}^2}. \tag{4.6}$$

To quantify local translational drift, a **one-second translational displacement RMSE** was utilized. This fixed-interval displacement error ($\Delta t = 1.0$ s) ensures that different topic publication rates do not define different physical comparison intervals. It does not fully evaluate relative transformations including relative rotation, which separates it from conventional SE(2) Relative Pose Error (RPE) definitions:

$$\Delta\mathbf{p}_k^{\mathrm{est}} = \mathbf{p}^{\mathrm{est}}(t_k + \Delta t) - \mathbf{p}^{\mathrm{est}}(t_k), \tag{4.7}$$

$$\Delta\mathbf{p}_k^{\mathrm{GT}} = \mathbf{p}^{\mathrm{GT}}(t_k + \Delta t) - \mathbf{p}^{\mathrm{GT}}(t_k), \tag{4.8}$$

$$e_{\Delta p,k} = \left\| \Delta\mathbf{p}_k^{\mathrm{est}} - \Delta\mathbf{p}_k^{\mathrm{GT}} \right\|_2, \tag{4.9}$$

$$\mathrm{RMSE}_{\Delta p,1s} = \sqrt{\frac{1}{M}\sum_{k=1}^{M} e_{\Delta p,k}^2}. \tag{4.10}$$

The yaw error was calculated using the wrapped angular difference

$$e_{\psi,k} = \text{wrapToPi}\left(\psi^{\text{est}}(t_k) - \psi^{\text{GT}}(t_k)\right), \tag{4.11}$$

and the yaw RMSE was computed as

$$\text{RMSE}_{\psi} = \sqrt{\frac{1}{N}\sum_{k=1}^{N} e_{\psi,k}^2}. \tag{4.12}$$

Wrapping the angular difference prevents equivalent headings on opposite sides of the $-\pi$ and $\pi$ boundary from producing artificially large errors. Handling periods of ICP tracking loss is important for interpreting the reported errors. An ICP estimate was classified as invalid when RTAB-Map reported `lost: true`, when no ICP estimate was available, or when the published pose contained non-finite values. A scan containing infinite range measurements was not automatically classified as invalid. If RTAB-Map published a finite pose with `lost: false`, the estimate remained in the evaluation even when its ground-truth error was large. ICP tracking availability was quantified as

$$P_{\text{valid}} = 100\frac{T_{\text{valid,ICP}}}{T_{\text{evaluation}}}, \tag{4.13}$$

where $T_{\text{valid,ICP}}$ is the total duration for which a valid ICP estimate was available. For the representative evaluation run, ICP tracking availability was 99.8%. Therefore, tracking loss affected only a negligible portion of the recorded data. The ICP error metrics in Table 4.5 were calculated using valid ICP estimates, while the wheel-odometry, AMCL, and EKF metrics were calculated over their respective synchronized evaluation intervals.

Table 4.5: Localization metrics from the representative ROS 2/Gazebo evaluations. Wheel odometry, ICP, and EKF were evaluated simultaneously, while AMCL was evaluated separately under the same nominal trajectory and initial condition. The final column reports the one-second translational displacement-error RMSE.

| **Method** | **APE RMSE (m)** | **Yaw RMSE (rad)** | **1-s displacement RMSE (m)** |
|---|---|---|---|
| Wheel odometry | 0.284 | 0.114 | 0.092 |
| Local ICP odometry | 0.125 | 0.045 | 0.055 |
| AMCL | 0.052 | 0.019 | 0.022 |
| EKF estimate | 0.048 | 0.016 | 0.018 |

The results show that wheel odometry provides continuous local motion but accumulates drift, while local ICP reduces local drift when registration remains reliable. In the representative evaluation, the EKF produced lower APE RMSE, yaw RMSE, and one-second translational displacement-error RMSE than wheel odometry and local ICP odometry by combining wheel-derived motion information with ICP pose measurements. AMCL provides drift-bounded, map-referenced localization; however, because it was evaluated in a separate operating mode, its numerical result should be interpreted as a cross-mode reference rather than as a direct simultaneous ranking against the other estimators. The trust monitor provides a separate diagnostic output and does not alter any of these estimates.

### 4.10.5 Computational Behavior

The complete simulation was able to execute at approximately real-time speed on the development computer. The LiDAR scan and local ICP odometry updated at approximately 10 Hz, while the EKF operated at its configured higher update rate. ICP registration typically required only a small fraction of the scan period in the tested simulation. These timing observations describe the development platform and are not hardware-independent performance guarantees.

## 4.11 Limitations

The ROS 2 implementation has the following limitations:

- **No complete ROS 2 fuzzy implementation:** the Chapter 3 fuzzy rule base and trust-dependent correction logic were not implemented in ROS 2.
- **Diagnostic-only trust:** `/icp_trust` is logged and visualized but does not modify the EKF measurement covariance or reject ICP updates.
- **Local ICP diagnostics:** `/odom_info` describes local registration behavior and does not directly measure consistency against the saved map or Gazebo ground truth.
- **Heuristic thresholds:** the trust thresholds require calibration for the chosen environment, scan density, speed, and ICP configuration.
- **Planar sensing:** the simulated sensor and localization pipeline use one horizontal scan plane and therefore do not exploit vertical geometry.

- **Weak and repetitive geometry:** long aisles and repeated shelf structures can reduce the observability or uniqueness of ICP registration.
- **AMCL initialization:** AMCL requires an initial pose estimate and sufficient motion and scan information for convergence.
- **Cross-mode evaluation:** results collected in different operating modes are less controlled than estimates evaluated simultaneously from the same bag.
- **Simulation scope:** Gazebo adds realistic software timing and rigid-body dynamics, but it does not reproduce every sensor, surface, wheel-slip, communication, and calibration effect present on a physical warehouse robot.

## 4.12 Discussion and Reflection

This chapter extends the MATLAB investigation into a modular ROS 2/Gazebo implementation. The resulting workspace includes encoder-integrated wheel odometry, planar LiDAR processing, local RTAB-Map ICP odometry, EKF-based estimation, SLAM Toolbox mapping, AMCL localization, Gazebo ground-truth logging, MATLAB post-processing, and an online ICP diagnostic monitor.

The ROS 2 work also clarified the boundary between the thesis contributions. Chapter 3 contains the complete fuzzy reliability-aware localization method and demonstrates how reliability influences estimation and correction. Chapter 4 implements the supporting robotic software architecture and provides online ICP diagnostic information, but it does not claim that the complete Chapter 3 fuzzy controller or adaptive covariance logic runs in ROS 2.

The occupancy-grid map was created once with SLAM Toolbox by manually driving the robot through the warehouse and was then reused by AMCL. RTAB-Map served a separate role as the local ICP odometry source. This separation avoids conflating mapping, local scan-based odometry, and map-referenced localization.

The implementation required explicit handling of message timing, TF ownership, estimator initialization, topic quality of service, lifecycle nodes, coordinate alignment, and logging. These system-level concerns do not appear to the same degree in the synchronous MATLAB simulations, but they are necessary for later transfer to physical hardware.

The resulting ROS 2 framework therefore provides a practical basis for future work. The next research step is to introduce a validated intermediary or custom estimator to converts

the diagnostic trust signal into principled ICP acceptance or covariance adaptation and then to evaluates the complete reliability-aware method on a physical AGV.

# Chapter 5

# Conclusion and Future Work

## 5.1 Conclusion

This thesis investigated encoder- and LiDAR-based localization for a differential-drive mobile robot operating in warehouse-like indoor environments. The work focused on two practical localization challenges: the accumulation of wheel-odometry drift over time and the reduced reliability of LiDAR scan matching when the live environment differs from the reference map. The research progressed from algorithm-level evaluation in MATLAB to the development of reliability-aware localization mechanisms, and finally to implementation of a supporting localization architecture in ROS 2 and Gazebo.

Chapter 1 reviewed the localization concepts most relevant to this problem, including wheel odometry, probabilistic estimation, LiDAR scan matching, particle filtering, and multi-sensor fusion. The review showed that encoder-based localization provides continuous short-term motion information but accumulates drift, whereas map-referenced sensing can correct this drift but may become unreliable when environmental observations no longer agree with the map. The research problem was therefore formulated around time-varying sensor reliability rather than assuming that a single sensing source should be trusted equally under all operating conditions.

Chapter 2 developed a unified MATLAB simulation framework for evaluating encoder dead reckoning, ICP scan-to-map localization, EKF fusion, and particle-filter localization under controlled warehouse conditions. The results demonstrated the complementary behavior of the methods. Encoder odometry provided smooth and computationally inexpensive short-term propagation but accumulated error over distance. ICP provided map-referenced localization but was sensitive to ambiguous geometry and incorrect registration.

Particle filtering provided a probabilistic map-based alternative at greater computational cost, while EKF fusion combined encoder continuity with LiDAR-based pose correction. Additional corridor, repeated-turn, and repeated-loop tests were used to examine the estimators under different trajectory geometries and extended operation. The repeated-loop experiment also demonstrated how a small persistent wheel-scale mismatch can produce progressively increasing odometric drift over repeated laps.

Chapter 3 developed the main reliability-aware contribution of the thesis. Three complementary mechanisms were introduced and intentionally kept separate. First, ICP registration RMSE and innovation consistency were used to adapt the effective ICP measurement covariance within the EKF. Finite but unreliable ICP measurements were therefore down-weighted, while invalid or non-finite ICP measurements were excluded from the measurement update. Second, a Mamdani fuzzy inference system combined ICP RMSE, position innovation, and yaw innovation into a continuous trust score used for controller-level blending of encoder, EKF, and ICP estimates. Third, a separate sustained-good condition required ICP to remain valid and consistent for multiple consecutive samples before gradual encoder anchoring was enabled within the FUZZY-BLEND method. This anchoring mechanism reduced the accumulated disagreement between the internal encoder state and a reliable map-referenced ICP estimate without modifying the standalone encoder-only baseline.

The Chapter 3 mismatch experiment demonstrated the benefit of adapting localization behavior to changing ICP reliability. During the temporary map-mismatch interval, ICP-only localization experienced severe scan-matching failures, producing a position RMSE of 4.9542 m within the mismatch window. The adaptive EKF reduced this error to 0.0629 m, while FUZZY-BLEND produced a mismatch-window position RMSE of 0.0367 m. The raw encoder-only baseline produced 0.0539 m over the same interval and, as expected for uncorrected dead reckoning, its error continued to accumulate later in the trajectory, reaching a post-mismatch position RMSE of 0.1152 m. Over the complete run, FUZZY-BLEND produced a position RMSE of 0.0159 m and a yaw RMSE of 0.0042 rad. However, FUZZY-BLEND was not the lowest-error method for every individual metric; for example, the EKF produced a slightly lower 95th-percentile position error and slightly lower post-mismatch position RMSE. The appropriate conclusion is therefore that the reliability-aware blended method reduced estimation error during the tested mismatch interval while maintaining low overall estimation error, rather than that it dominated every method under every metric.

Chapter 4 transferred the localization workflow from the synchronous MATLAB environment to a modular ROS 2/Gazebo implementation. The resulting system included wheel odometry, planar LiDAR processing, RTAB-Map local ICP odometry, EKF-based

state estimation, SLAM Toolbox mapping, AMCL map-referenced localization, transform management, logging, visualization, and a custom online ICP trust monitor. This stage introduced practical robotics considerations that were largely absent from the MATLAB implementation, including asynchronous sensor timing, topic interfaces, TF-frame ownership, estimator initialization, lifecycle behavior, and real-time software integration.

The ROS 2/Gazebo evaluation used APE RMSE, yaw RMSE, and one-second translational displacement-error RMSE. In the representative evaluation, wheel odometry produced an APE RMSE of 0.284 m, local ICP odometry 0.125 m, and the EKF estimate 0.048 m. AMCL produced an APE RMSE of 0.052 m in a separate operating mode. The corresponding one-second translational displacement-error RMSE values were 0.092, 0.055, 0.018, and 0.022 m for wheel odometry, local ICP, EKF, and AMCL, respectively. Because AMCL was evaluated separately and because the ROS 2 experiments used different estimators, software components, and operating conditions from the MATLAB experiments, these results should not be interpreted as reproducing an identical numerical ranking between the two environments. Instead, they demonstrate the expected qualitative behavior of the implemented localization components and establish a practical software platform for further development.

An important boundary of the present work is that the complete reliability-aware algorithm from Chapter 3 was not integrated into the ROS 2 fusion loop. The ROS 2 ICP trust monitor provides diagnostic information about local scan-matching behavior, but this signal does not currently modify EKF measurement covariance, reject ICP measurements, or apply the Chapter 3 fuzzy feedback-blending and encoder-anchoring logic. The ROS 2 implementation should therefore be interpreted as a system-level implementation of the supporting localization architecture rather than as a direct ROS 2 reproduction of the Chapter 3 reliability-aware algorithm.

Overall, this thesis makes three main contributions. First, it provides a unified MATLAB framework for evaluating encoder-, ICP-, EKF-, and particle-filter-based localization in a representative warehouse environment. Second, it develops and evaluates a reliability-aware localization architecture that separates adaptive EKF measurement weighting, fuzzy controller-level feedback blending, and conservative encoder anchoring during sustained reliable ICP. Third, it establishes a modular ROS 2/Gazebo localization framework that integrates wheel odometry, local LiDAR odometry, EKF fusion, map-based localization, and online ICP diagnostics in a software structure suitable for further extension.

The results support the central premise of this thesis: localization reliability should be treated as time-varying rather than assumed constant. When scan matching becomes inconsistent with the available reliability indicators, its influence can be reduced; when

map-referenced localization remains consistently reliable, it can be used conservatively to limit accumulated odometric drift. At the same time, the presented results remain simulation-based and depend on the selected environment, trajectory, thresholds, and noise realization.

## 5.2 Future Work

Several extensions of this work would be valuable for further validating and developing the proposed localization approach.

First, the complete reliability-aware method developed in Chapter 3 should be integrated into the ROS 2 localization framework. In the current implementation, the ICP trust monitor is used only as a diagnostic signal. A natural next step is to use the reliability indicators directly within the ROS 2 fusion loop to adapt ICP measurement covariance, reject invalid measurements when necessary, and implement the fuzzy feedback-blending and encoder-anchoring mechanisms developed in MATLAB.

Second, the localization system should be evaluated on a physical mobile robot. The current results are based on MATLAB and Gazebo simulations, which cannot reproduce all effects encountered on real hardware. Physical experiments would allow the system to be evaluated under real wheel slip, encoder calibration errors, LiDAR measurement noise, communication delays, sensor timing differences, computational limitations, and environmental changes.

Third, more extensive multi-run experiments should be performed to evaluate the sensitivity of the reliability-aware method to parameters such as ICP RMSE thresholds, innovation thresholds, fuzzy membership functions, and the number of consecutive reliable ICP samples required before encoder anchoring. Testing multiple trajectories, obstacle configurations, map mismatch conditions, and noise realizations would provide a stronger statistical characterization of the method's robustness.

Future experiments could also investigate more challenging warehouse conditions, including dynamic obstacles, longer trajectories, repetitive aisles, partial map changes, and larger localization disturbances. These tests would help determine the operating conditions under which each reliability indicator is most informative and identify cases in which additional reliability measures may be required.

Finally, the ROS 2 implementation provides a foundation for investigating real-time deployment considerations. Future work could examine computational load, communication latency, sensor synchronization, update-rate selection, and estimator execution timing

when the complete reliability-aware localization method is executed on the onboard computing hardware of a physical AGV.

Together, these extensions would allow the proposed approach to progress from simulation-based validation toward a fully integrated and experimentally validated reliability-aware localization system for warehouse mobile robots.

# References


[1] S. Agarwal and K. Mierle. Ceres solver: Tutorial and reference. In *Google Research Report*, 2012.

[2] S. S. Ahmad and T. Namerikawa. Ekf-slam based on intermittent sensor data. In *Proceedings of the IEEE International Conference on Control Applications (CCA)*, pages 722–727, 2013.

[3] J. Andrade-Cetto and A. Sanfeliu. The effects of partial observability on the slam state estimation consistency. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 397–402, 2005.

[4] V. Ankisan, M. S. Kumar, and K. Subbarao. Srufastslam: A swarm robot unit approach to particle filter based slam. In *Proceedings of the IEEE International Conference on Mechatronics and Automation (ICMA)*, pages 1008–1013, 2013.

[5] B. Bescós, J. M. Facil, J. Civera, and J. Neira. Dynaslam: Tracking, mapping, and inpainting in dynamic scenes. *IEEE Robotics and Automation Letters*, 3(4):4076–4083, 2018.

[6] Paul J. Besl and Neil D. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992.

[7] M. Bijelic, T. Gruber, and W. Ritter. Benchmarking the robustness of lidar perception to adverse weather. *IEEE Transactions on Intelligent Transportation Systems*, 21(11):4882–4896, 2020.

[8] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart. Robust visual inertial odometry using a direct ekf-based approach. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 298–304, 2015.

[9] J. Borenstein and L. Feng. Measurement and correction of systematic odometry errors in mobile robots. *IEEE Transactions on Robotics and Automation*, 12(6):869–880, 1996.

[10] A. Borji and L. Itti. State-of-the-art in visual attention modeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):185–207, 2013.

[11] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, 32(6):1309–1332, 2016.

[12] C. Campos, R. Elvira, J. J. G. Rodrguez, J. M. Montiel, and J. D. Tards. Orb-slam3: An accurate open-source library for visual, visual–inertial, and multi-map slam. *IEEE Transactions on Robotics*, 37(6):1874–1890, 2021.

[13] L. Carlone and J. C. Dellaert. Attention and robustness in graph optimization for slam. *IEEE Transactions on Robotics*, 35(4):1020–1036, 2019.

[14] Central Museum of the Armed Forces (Moscow). Soviet tt-26 "teletank" (1930s1940s), 1941. Historical photograph; teleoperated tank used by the Red Army. Commonly cited in museum archives and robotics histories.

[15] F. A. Auat Cheein, N. Lopez, C. M. Soria, F. Rossomando, and R. Carelli. Slm for a robotic wheelchair navigating in unknown environments. *IEEE Latin America Transactions*, 7(4):438–444, 2009.

[16] S. Choi, T. Kim, and S. Oh. Adaptive unscented kalman filter-based slam for a mobile robot. In *Proceedings of the International Conference on Control, Automation and Systems*, pages 470–475, 2009.

[17] W. Churchill and P. Newman. Experience-based navigation for long-term localisation. *The International Journal of Robotics Research*, 32(14):1645–1661, 2013.

[18] Comau S.p.A. Agile1500 autonomous mobile robot (product image), 2024. Official product photography; used for illustrative purposes in academic review.

[19] F. Dellaert. Factor graphs and gtsam: A hands-on introduction. In *Technical Report GT-RIM-CP_2012_002, Georgia Institute of Technology*, 2012.

[20] F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte carlo localization for mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1322–1328, 1999.

[21] Deutsches Panzermuseum Munster. German goliath tracked mine (sd.kfz. 302/303), 1942. Historical photograph; wire/radio controlled demolition vehicle used by the Wehrmacht.

[22] G. Dissanayake, P. Newman, S. Clark, H. Durrant-Whyte, and M. Csorba. A solution to the simultaneous localization and map building (slam) problem. *IEEE Transactions on Robotics and Automation*, 17(3):229–241, 2001.

[23] D. Fox. Adapting the sample size in particle filters through kld-sampling. *The International Journal of Robotics Research*, 22(12):985–1003, 2003.

[24] Dieter Fox, Wolfram Burgard, Frank Dellaert, and Sebastian Thrun. Monte carlo localization: Efficient position estimation for mobile robots. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI)*, pages 343–349, 1999.

[25] Pranav Ganti and Steven L. Waslander. Network uncertainty informed semantic feature selection for visual slam. In *2019 16th Conference on Computer and Robot Vision (CRV)*, pages 121–128, 2019.

[26] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The LaTeX Companion*. Addison-Wesley, Reading, Massachusetts, 1994.

[27] G. Grisetti, R. Kummerle, C. Stachniss, and W. Burgard. A tutorial on graph-based slam. *IEEE Intelligent Transportation Systems Magazine*, 2(4):31–43, 2010.

[28] Xiaofeng Han, Shunpeng Chen, Zenghuang Fu, Zhe Feng, Lue Fan, Dong An, Changwei Wang, Li Guo, Weiliang Meng, Xiaopeng Zhang, Rongtao Xu, and Shibiao Xu. Multimodal fusion and visionlanguage models: A survey for robot vision, 2025.

[29] R. Havangi and M. A. Nekoui. A new fastslam using particle swarm optimization algorithm. *Robotics and Autonomous Systems*, 62(6):841–852, 2014.

[30] D. Hershberger et al. Rviz: Ros visualization tool. Robot Operating System (ROS) Package, 2010. https://wiki.ros.org/rviz.

[31] S. Indelman, Y. Placed, and E. Carrera. A survey on active simultaneous localization and mapping: State of the art and new frontiers. *IEEE Transactions on Robotics*, 39(4):3012–3035, 2023.

[32] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard, and F. Dellaert. isam2: Incremental smoothing and mapping using the bayes tree. *The International Journal of Robotics Research*, 31(2):216–235, 2012.

[33] A. Kendall and Y. Gal. What uncertainties do we need in bayesian deep learning for computer vision? *Advances in Neural Information Processing Systems (NeurIPS)*, 30, 2017.

[34] Nathan P. Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 3, pages 2149–2154, 2004.

[35] R. Kmmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g2o: A general framework for graph optimization. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3607–3613, 2011.

[36] Mathieu Labbé and François Michaud. Rtab-map as an open-source lidar and visual slam library for large-scale and long-term online operation. *Journal of Field Robotics*, 36(2):416–446, 2019.

[37] Q. V. Le, A. J. Smola, and S. Canu. Heteroscedastic gaussian process regression. In *Proceedings of the 22nd International Conference on Machine Learning (ICML)*, pages 489–496, 2005.

[38] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale. Keyframe-based visualinertial odometry using nonlinear optimization. *The International Journal of Robotics Research*, 34(3):314–334, 2015.

[39] Y. Li, S. Chen, and Z. Zhang. Dynamic-slam: Removing dynamic objects for reliable slam in urban environments. *IEEE Transactions on Intelligent Transportation Systems*, 22(5):2919–2930, 2021.

[40] Y. Li, M. Qiu, and K. Wang. Neuro-odometry: Learning-based drift compensation for mobile robot dead reckoning. *IEEE Robotics and Automation Letters*, 9(2):1842–1849, 2024.

[41] Y. Liu, H. Zhao, and J. Zhang. Deep-fastslam: Neural proposal distribution for particle filter slam. *IEEE Robotics and Automation Letters*, 9(3):2114–2121, 2024.

[42] S. Lowry, N. Snderhauf, P. Newman, J. Leonard, D. Cox, P. Corke, and M. Milford. Visual place recognition: A survey. *IEEE Transactions on Robotics*, 32(1):1–19, 2016.

[43] L. Ma, Z. Wang, C. Yang, and F. Gao. Lidar and high-definition map fusion for autonomous driving: A review. *Sensors*, 19(18):3899, 2019.

[44] Steve Macenski and Ivona Jambrecic. Slam toolbox: Slam for the dynamic world. *Journal of Open Source Software*, 6(61):2783, 2021.

[45] Steve Macenski, Francisco Martín, Ruffin White, and Jonatan Ginés Clavero. The marathon 2: A navigation system. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2718–2725, 2020.

[46] Steven Macenski, Tully Foote, Brian Gerkey, Chris Lalancette, and William Woodall. Robot operating system 2: Design, architecture, and uses in the wild. *Science Robotics*, 7(66):eabm6074, 2022.

[47] Ebrahim H. Mamdani and Sedrak Assilian. An experiment in linguistic synthesis with a fuzzy logic controller. *International Journal of Man-Machine Studies*, 7(1):1–13, 1975.

[48] MarketsandMarkets Research. Autonomous mobile robots (amr) market size, share and forecast to 2030. Industry Research Report, 2025.

[49] R. Martinez-Cantin, J. A. Castellanos, and J. Neira. A new approach for slam using the unscented kalman filter. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 206–211, 2005.

[50] L. Martnez, S. Indelman, and C. Cadena. A survey on active and semantic slam: State of the art and new frontiers. *IEEE Transactions on Robotics*, 39(6):4123–4148, 2023.

[51] Jiaqi Miao et al. Efficient end-to-end visual localization for autonomous driving with hd map association, 2025.

[52] Ehsan Mohammadbagher. *Self-Localization for Autonomous Driving Using Vector Maps and Multi-Modal Odometry.* PhD thesis, University of Waterloo, Waterloo, Ontario, Canada, 2023.

[53] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. Fastslam: A factored solution to the simultaneous localization and mapping problem. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, pages 593–598, 2002.

[54] Thomas Moore and Daniel Stouch. A generalized extended kalman filter implementation for the robot operating system. In Emanuele Menegatti, Nathan Michael, Karsten Berns, and Hiroaki Yamaguchi, editors, *Intelligent Autonomous Systems 13*, volume 302 of *Advances in Intelligent Systems and Computing*, pages 335–348. Springer, Cham, 2016.

[55] H. Moravec. Obstacle avoidance and navigation in the real world by a seeing robot rover. *Stanford University Technical Report*, 1980.

[56] Jad Mounsef et al. Vio-go: Optimizing event-based slam parameters for dynamic, low-light industrial environments. *Frontiers in Robotics and AI*, 12:1541017, 2025.

[57] A. I. Mourikis and S. I. Roumeliotis. A multi-state constraint kalman filter for vision-aided inertial navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3565–3572, 2007.

[58] R. Mur-Artal, J. M. M. Montiel, and J. D. Tards. Orb-slam: A versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.

[59] R. Mur-Artal and J. D. Tards. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017.

[60] Ryo Nakashima and Akihito Seki. Uncertainty-based adaptive sensor fusion for visual-inertial odometry under various motion characteristics. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3119–3125, 2020.

[61] N. Nilsson. A mobile automaton: An application of artificial intelligence techniques. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 509–520, 1969.

[62] N. Nilsson. *The Quest for Artificial Intelligence: A History of Ideas and Achievements*. Cambridge University Press, 2010.

[63] Prabin Kumar Panigrahi and Sukant Kishoro Bisoy. Localization strategies for autonomous mobile robots: A review. *Journal of King Saud University - Computer and Information Sciences*, 34(8):6019–6039, 2022.

[64] T. Qin, P. Li, and S. Shen. Vins-fusion: A generic framework for visual-inertial navigation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 6429–6436, 2019.

[65] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng. Ros: An open-source robot operating system. In *IEEE International Conference on Robotics and Automation Workshop on Open Source Software*, pages 1–6, 2009.

[66] M. Rahman, N. Ali, and K. Zhang. A review of autonomous mobile robots for the warehouse environment: Navigation, sensing and control. *arXiv preprint*, 2024.

[67] J. Rehder and R. Siegwart. Extending kalman filter-based sensor fusion for multimodal odometry with bias estimation. *IEEE Robotics and Automation Letters*, 1(1):393–400, 2016.

[68] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and D. Rus. Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5135–5142, 2020.

[69] SHARK Robotics. Colossus firefighting robot (press/photo), 2023. Official press/-gallery image from SHARK Robotics; used for illustrative purposes in academic review.

[70] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza. *Introduction to Autonomous Mobile Robots*. MIT Press, 2nd edition, 2011.

[71] SRI International. Shakey the robot (historical photo), 1969. Publicity/archival photograph credited to SRI International.

[72] E. Stumm, C. Cadena, and R. Siegwart. Self-supervised online learning for long-term autonomous navigation. *Journal of Field Robotics*, 33(3):254–273, 2016.

[73] L. Sun, Q. Zhang, and H. Zhao. Semantic fusion for object-level mapping. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 2683–2690, 2018.

[74] C. Thorpe, M. Hebert, T. Kanade, and S. Shafer. Vision and navigation for the carnegie mellon navlab. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(3):362–373, 1990.

[75] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2005.

[76] Inam Ullah, Deepak Adhikari, Habib Khan, M. Shahid Anwar, Shabir Ahmad, and Xiaoshan Bai. Mobile robot localization: Current challenges and future prospective. *Computer Science Review*, 53:100651, 2024.

[77] W. Grey Walter. An imitation of life. *Scientific American*, 182:42–45, 1950.

[78] F. Wang, Y. Liu, and X. Zhou. Semantic slam system for mobile robots based on large visual–lidar models. *Scientific Reports*, 15(90340):1–15, 2025.

[79] H. Wang, Y. Chen, and D. Zhao. Hdmapfusion: Hybrid lidarcameraimu localization for gnss-degraded urban driving. *IEEE Transactions on Intelligent Transportation Systems*, 25(7):6911–6923, 2024.

[80] H. Wang, G. Zhang, Y. Zhong, and G. H. Peng. Omnidirectional vision-based ekf-slam for a mobile robot. *International Journal of Advanced Robotic Systems*, 9(4):1–11, 2012.

[81] R. Wolcott and R. M. Eustice. Visual localization within lidar maps for automated urban driving. *IEEE Transactions on Intelligent Vehicles*, 2(4):1–16, 2017.

[82] Z. Wu, Q. Li, M. Zhao, and S. Yang. Minislam: Lightweight semantic slam for micro aerial vehicles with adaptive attention. *IEEE Robotics and Automation Letters*, 8(9):5748–5755, 2023.

[83] Y. Xie, F. Zhang, and L. Liu. Uncertainty-aware deep slam for outdoor robots in dynamic environments. *IEEE Robotics and Automation Letters*, 9(2):1468–1475, 2024.

[84] W. Xu and F. Zhang. Fast-lio2: Fast, direct lidar-inertial odometry. *IEEE Transactions on Robotics*, 38(4):2053–2073, 2022.

[85] Jiawei Yang, Chen Zhang, Bo Liu, and Yimu Wang. Rwkv-vio: An efficient and low-drift visualinertial odometry using recurrent weighted key-value models. *Sensors*, 25(18):5737, 2025.

[86] S. Yuan, A. Khajepour, and B. Litkouhi. Review of state estimation methods for vehicle motion and tire-road forces. *Vehicle System Dynamics*, 2019.

[87] J. Zhang and S. Singh. Loam++: Lidar odometry and mapping with high-definition map alignment for autonomous vehicles. *IEEE Robotics and Automation Letters*, 6(3):4632–4639, 2021.

[88] J. Zhao, L. Chen, and H. Lee. Ai-based approaches for improving autonomous mobile robot navigation and localization. *Engineering Applications of Artificial Intelligence*, 135:108634, 2025.

# APPENDICES

# Appendix A

# Jacobian Derivations for the EKF

This appendix summarizes the Jacobian matrices used in the MATLAB EKF implementation described in Chapter 2. The robot state is

$$\mathbf{x}_k = \begin{bmatrix} x_k & y_k & \theta_k \end{bmatrix}^T,$$

and the encoder-derived control input is

$$\mathbf{u}_k = \begin{bmatrix} v_k & \omega_k \end{bmatrix}^T,$$

where $v_k$ is the linear velocity, $\omega_k$ is the angular velocity, and $\Delta t$ is the sampling interval.

## A.1 First-Order Discrete-Time Motion Model

The MATLAB implementation uses first-order Euler integration:

$$x_{k+1} = x_k + v_k \Delta t \cos\theta_k, \tag{A.1}$$
$$y_{k+1} = y_k + v_k \Delta t \sin\theta_k, \tag{A.2}$$
$$\theta_{k+1} = \text{wrap}\left(\theta_k + \omega_k \Delta t\right). \tag{A.3}$$

The corresponding nonlinear state-transition function is

$$f(\mathbf{x}_k, \mathbf{u}_k) = \begin{bmatrix} x_k + v_k \Delta t \cos\theta_k \\ y_k + v_k \Delta t \sin\theta_k \\ \mathrm{wrap}(\theta_k + \omega_k \Delta t) \end{bmatrix}.$$

Linearizing this model with respect to the state gives the state-transition Jacobian

$$F_k = \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_k, \mathbf{u}_k} = \begin{bmatrix} 1 & 0 & -v_k \Delta t \sin\theta_k \\ 0 & 1 & v_k \Delta t \cos\theta_k \\ 0 & 0 & 1 \end{bmatrix}.$$

The predicted covariance is therefore computed as

$$P_{k+1}^{-} = F_k P_k F_k^T + Q.$$

where $Q$ is the state-space process-noise covariance. In the MATLAB implementation, $Q$ is added directly during covariance propagation; therefore, a separate input-noise Jacobian $G_k$ is not used.

## A.2 Measurement Jacobian

ICP provides a direct planar-pose measurement,

$$\mathbf{z}_k = \begin{bmatrix} x_k^{\mathrm{ICP}} & y_k^{\mathrm{ICP}} & \theta_k^{\mathrm{ICP}} \end{bmatrix}^T.$$

The measurement model is

$$\mathbf{z}_k = H\mathbf{x}_k + \boldsymbol{\eta}_k, \qquad \boldsymbol{\eta}_k \sim \mathcal{N}\left(\mathbf{0}, R_{\mathrm{eff},k}\right),$$

where the measurement Jacobian is

$$H = I_3.$$

The yaw component of the innovation is wrapped to the interval $(-\pi, \pi]$ before the EKF correction is applied.

# Appendix B

# ROS 2 Workspace Details

This appendix summarizes the main ROS 2 workspace structure used for the Gazebo-based localization implementation described in Chapter 4. The workspace was organized so that the robot model, launch files, localization nodes, and visualization utilities were kept modular and reusable.

Listing B.1 shows the main directory structure.

```
amr_localization_ws/
|-- src/
|   |-- robot_description/
|   |   |-- description/robot.urdf.xacro
|   |   |-- world/catalyst_env.world
|   |   |-- launch/display.launch.py
|   |   |-- launch/navigation.launch.py
|   |   |-- config/ekf.yaml
|   |   |-- rviz/config.rviz
|   |   `-- meshes/
|   |
|   |-- robot_path_publisher/
|   |   |-- odom_to_path.py
|   |   |-- icp_odom_to_path.py
|   |   |-- fusion_odom_to_path.py
|   |   `-- gazebo_ground_truth_path_publisher.py
|   |
|   |-- path_generator/
|   |   `-- path_generator.py
```

```
|   |
|   |-- nav2_bringup/
|   |-- slam_toolbox/
|   `-- rtabmap_ros/
|
|-- build/
|-- install/
`-- log/
```

Listing B.1: Directory structure of the ROS 2 workspace.

# Appendix C

# ROS 2 Configuration Excerpts

This appendix provides representative excerpts from the most relevant configuration files used in the ROS 2 localization framework. Only the most important blocks are shown here for readability.

## C.1 Robot Model (URDF/Xacro) Excerpt

Listing C.1 shows a representative excerpt from the AGV5 robot model. The excerpt includes example geometric parameters, the front-mounted LiDAR frame, the differential-drive odometry plugin, and the Gazebo LiDAR sensor plugin.

```
<robot xmlns:xacro="http://www.ros.org/wiki/xacro"
       name="AGV5">

  <!-- Drive-wheel geometry and placement -->
  <xacro:property name="wheel_x" value="0.0"/>
  <xacro:property name="wheel_y" value="-0.23"/>
  <xacro:property name="wheel_z" value="0.0"/>

  <xacro:macro name="wheel" params="prefix side">
    <link name="${prefix}_link">
      <collision>
        <origin xyz="0 0 0" rpy="${pi/2} 0 0"/>
        <geometry>
          <cylinder length="0.10" radius="0.10"/>
```

```
      </geometry>
    </collision>
  </link>

  <joint name="${prefix}_joint" type="continuous">
    <parent link="base_link"/>
    <child link="${prefix}_link"/>
    <origin xyz="${wheel_x} ${side * wheel_y} ${wheel_z}"
            rpy="0 0 0"/>
    <axis xyz="0 1 0"/>
  </joint>
</xacro:macro>

<xacro:wheel prefix="drive_wheel_left" side="1"/>
<xacro:wheel prefix="drive_wheel_right" side="-1"/>

<!-- LiDAR mounting transform -->
<joint name="lidar_front_to_base_link" type="fixed">
  <parent link="base_link"/>
  <child link="lidar_front"/>
  <origin xyz="0.4 0 0.22" rpy="0 0 0"/>
</joint>

<!-- Gazebo LiDAR sensor -->
<gazebo reference="lidar_front">
  <sensor name="front_lidar" type="gpu_ray">
    <update_rate>10</update_rate>
    <ray>
      <scan>
        <horizontal>
          <samples>1080</samples>
          <min_angle>-3.14159</min_angle>
          <max_angle>3.14159</max_angle>
        </horizontal>
      </scan>
      <range>
        <min>0.05</min>
        <max>30.0</max>
        <resolution>0.01</resolution>
      </range>
```

```
      </ray>

      <plugin name="gazebo_ros_ray_sensor"
              filename="libgazebo_ros_ray_sensor.so">
        <ros>
          <remapping>~/out:=ray/lidar_points</remapping>
        </ros>
        <frame_name>lidar_front</frame_name>
      </plugin>
    </sensor>
  </gazebo>

  <!-- Differential-drive plugin -->
  <gazebo>
    <plugin name="diff_drive"
            filename="libgazebo_ros_diff_drive.so">
      <left_joint>drive_wheel_left_joint</left_joint>
      <right_joint>drive_wheel_right_joint</right_joint>

      <wheel_separation>0.46</wheel_separation>
      <wheel_diameter>0.20</wheel_diameter>
      <odometry_source>encoder</odometry_source>
      <update_rate>50</update_rate>

      <robot_base_frame>base_link</robot_base_frame>
      <odometry_frame>odom</odometry_frame>
      <odometry_topic>wheel_odom</odometry_topic>

      <publish_odom>true</publish_odom>
      <publish_odom_tf>false</publish_odom_tf>
      <publish_wheel_tf>false</publish_wheel_tf>
    </plugin>
  </gazebo>

</robot>
```

Listing C.1: Key excerpts from the AGV5 robot model definition (`robot.urdf.xacro`).

## C.2 Gazebo World File Excerpt

Listing C.2 shows an excerpt from the Gazebo warehouse environment used for the ROS 2 experiments. The example includes the Gazebo state plugin and one representative shelf model.

```
<world name="default">
  <plugin name="gazebo_ros_state" filename="libgazebo_ros_state.so
   ">
    <ros>
      <argument>model_states:=model_states_demo</argument>
    </ros>
  </plugin>

  <light name="sun" type="directional">
    <direction>-0.5 0.1 -0.9</direction>
  </light>

  <model name="shelf_final_0">
    <pose>4.25 -7.5 0 0 0 0</pose>
    <link name="base_link">
      <visual name="visual">
        <geometry>
          <mesh>
            <uri>model://shelf_final/meshes/base_link.STL</uri>
          </mesh>
        </geometry>
      </visual>
    </link>
  </model>

  <!-- ... additional shelves, walls, and environment objects ...
   -->
</world>
```

Listing C.2: Excerpt from the Gazebo world file (`catalyst_env.world`).

## C.3 EKF Configuration Excerpt

Listing C.3 shows the principal input-selection, frame, and update-rate settings of the `robot_localization` EKF described in Chapter 4. The filter operates in two-dimensional mode, uses Gazebo simulation time, and publishes the `odom`→`base_footprint` transform.

Wheel odometry supplies forward velocity and yaw rate, while local ICP odometry supplies planar position and yaw. Wheel-integrated pose and ICP-derived velocity are excluded from the selected inputs. AMCL is evaluated separately and is not an input to this local EKF.

Measurement covariances are obtained from the incoming odometry messages. The process-noise and initial-estimate covariance settings are omitted from this excerpt for brevity.

```
ekf_filter_node:
  ros__parameters:
    use_sim_time: true
    frequency: 30.0
    two_d_mode: true

    map_frame: map
    odom_frame: odom
    base_link_frame: base_footprint
    world_frame: odom
    publish_tf: true

    # Input-selection order:
    # x, y, z, roll, pitch, yaw,
    # vx, vy, vz, vroll, vpitch, vyaw,
    # ax, ay, az

    # Wheel odometry: forward velocity and yaw rate
    odom0: /wheel_odom
    odom0_config: [false, false, false,
                   false, false, false,
                   true,  false, false,
                   false, false, true,
                   false, false, false]

    # Local ICP odometry: planar position and yaw
```

```
27      odom1: /icp_odom
28      odom1_config: [true,  true,  false,
29                     false, false, true,
30                     false, false, false,
31                     false, false, false,
32                     false, false, false]
33
34      # Other settings omitted for brevity.
```

Listing C.3: Principal settings from the EKF configuration file (`ekf.yaml`).