

UNIVERSITÀ DEGLI STUDI DELL'AQUILA
DEPARTMENT OF INDUSTRIAL AND INFORMATION
ENGINEERING AND ECONOMICS

DOTTORATO DI RICERCA IN
Industrial and Information Engineering and Economics

CURRICULUM
Mechanical, Energy and Management Engineering
xxxviii ciclo

TESI DI DOTTORATO IN

Autonomous Navigation and
Intelligent Visual Inspection in
Industrial Environments

SSD: 09/IIND-02

Coordinatore del Corso

Prof. Katia Gallucci

Dottorando

Muhammad Tayyab

Tutor

Prof. Pierluigi Beomonte Zobel

Co-Supervisor

Prof. Francesco Durante

A.Y. 2024/2025

Abstract

This dissertation examines two independent research directions: autonomous mobile navigation and industrial visual inspection. Each direction is developed as a complete and self-contained system, from problem formulation to implementation and controlled experimental evaluation. The two domains respond to different industrial needs, employ different technical philosophies, and are assessed against different operational criteria. The navigation study pursues a minimalist sensing and control strategy for dependable operation in structured indoor environments, whereas, in a separate research domain, the visual inspection study adopts a high-fidelity deep-learning pipeline for complex component-verification tasks in automotive assembly.

The first direction focuses on a minimalist navigation framework intended for robots performing sterilisation-related tasks in corridor-like indoor spaces representative of the pharmaceutical industry. The system uses a single two-dimensional LiDAR together with wheel odometry, avoiding complex sensing configurations that are difficult to maintain in environments with strict operational constraints. A ROS-based navigation pipeline is configured around AMCL localisation, static-map global planning, a rolling local costmap, and a tuned Dynamic Window Approach planner. Parameter refinement covers footprint geometry, obstacle-layer behaviour, inflation radii, update frequencies, and velocity limits. Navigation behaviour is assessed through repeated clockwise and counter-clockwise waypoint cycles in a controlled indoor layout shaped to reflect corridor constraints. The results demonstrate reliable localisation in most scenarios, though systematic lateral drift (up to 290 mm) was observed during clockwise right-hand turns. This indicates that route completion and localisation recovery were achieved, but that waypoint repeatability remained direction-dependent and requires cautious interpretation in the absence of loaded-platform sterilisation trials.

Complementary to the navigation task, the second research direction develops a multi-class inspection system for identifying missing components on fuel-tank assemblies. A dedicated dataset of real components is constructed and annotated to capture typical visual variability in such assemblies. YOLOv8 and Faster R-CNN models are trained under matched conditions to allow direct comparison. YOLOv8 achieves stronger overall performance in precision, recall, and mAP, and cross-dataset evaluation with the NPU-BOLT benchmark demonstrates that the detector generalises beyond the training setup. Real-time testing in a controlled

inspection environment, including staged removal of components, confirms that the detector can identify missing parts at frame rates compatible with in-line quality control.

Taken together, the two research directions illustrate how carefully matched engineering choices, parameter strategies, and experimental validation can yield practical solutions in different industrial contexts. The work provides empirical evidence that deployable performance depends not on enforcing a single design doctrine, but on aligning system architecture with the operational objective of each task.

Acknowledgments

First and foremost, I offer my deepest praise and gratitude to **ALLAH Almighty**, the Most Gracious and the Most Merciful, whose wisdom, protection, and countless blessings have sustained me throughout this journey. I extend my profound respect to the **Holy Prophet Muhammad (PBUH)**, whose teachings of perseverance, humility, and the pursuit of knowledge continue to serve as my guiding light.

I am deeply indebted to my supervisors, **Prof. Pierluigi Beomonte Zobel** and **Prof. Francesco Durante**. Their expert guidance, profound academic insight, and unwavering support were instrumental in shaping the direction and quality of this research. Their constructive feedback and patient mentorship challenged me to refine my critical thinking and strengthened my work at every stage. It has been a true privilege to conduct this research under their supervision.

My sincere appreciation also extends to **Francesco Vacca**, whose industrial expertise and practical perspective added significant value to the applied components of this thesis. His collaboration enriched the real-world relevance of my research, for which I am very grateful.

I gratefully acknowledge the financial support provided for this work. This research has been funded by the **European Union – NextGenerationEU** under the Italian Ministry of University and Research (MUR) National Innovation Ecosystem grant **ECS00000041 – VITALITY – CUP E13C22001060006**. This support was essential for the development and successful completion of this project.

On a personal note, I owe everything to my family. My father, **Muhammad Khalid**, and my mother, **Najma Sadiq**, have been my greatest pillars of strength. Their unconditional love, prayers, and sacrifices have been the foundation of my success. I am equally grateful to my wife, **Dr. Hurria Waheed**, whose patience, resilience, and unwavering belief in me sustained me through the most challenging phases of this PhD. Her support, even from afar, has been invaluable; this achievement is as much hers as it is mine.

Finally, I extend my thanks to my colleagues, friends, and academic peers. Your discussions, collaboration, and encouragement made this journey memorable. To all of you, I offer my sincere and enduring gratitude.

Contents

Abstract	i
Acknowledgments	iii
List of Acronyms	xvii
1 Introduction	1
1.1 Background and Motivation	1
1.1.1 Autonomous Navigation in Sterilisation-Oriented Indoor Spaces	2
1.1.2 Multi-Class Visual Inspection on Industrial Assembly Lines . .	4
1.1.3 Integrative Motivation: Simplicity as a Design Strategy	5
1.2 Industrial Context and Historical Evolution	6
1.2.1 Autonomous Navigation in Structured Environments	6
1.2.2 Machine Vision for Industrial Inspection	7
1.3 Fundamental Technical Challenges	8
1.3.1 Challenges in LiDAR-Only Indoor Navigation	8
1.3.2 Challenges in Multi-Class Missing-Component Detection . . .	10
1.4 Why Existing Approaches Are Insufficient	11
1.4.1 Limitations of Sensor-Rich Navigation Pipelines	11
1.4.2 Limitations of Conventional Industrial Vision Systems	12
1.5 Theoretical Rationale for Minimalist System Design	12
1.6 Research Questions	14
1.7 Research Objectives	15
1.7.1 Objectives for Autonomous Sterilisation Navigation	15
1.7.2 Objectives for Multi-Class Missing-Component Detection . . .	16
1.7.3 Integrative Objectives	16
1.8 Scope, Assumptions, and Limitations	17
1.8.1 Scope and Assumptions for Autonomous Navigation	17
1.8.2 Scope and Assumptions for Visual Inspection	18
1.9 Research Methodology Overview	18
1.9.1 Methodology for Navigation Experiments	18
1.9.2 Methodology for Inspection Experiments	19
1.10 Scientific Significance and Real-World Impact	20

1.11	Contributions of the Thesis	21
1.11.1	Contributions to Autonomous Navigation	21
1.11.2	Contributions to Visual Inspection	22
1.11.3	Integrative Contributions	22
1.12	Structure of the Thesis	22
2	Literature Review	24
2.1	Introduction	24
2.2	Foundations of Autonomous Mobile Robotics	25
2.2.1	Industrial Deployment Considerations	27
2.3	Minimalist Sensor Architectures	27
2.3.1	2D vs. 3D LiDAR in Structured Indoor Spaces	28
2.4	LiDAR and IMU Sensing Principles	28
2.4.1	Motivation for Excluding IMU in This Thesis	29
2.5	Multi-Sensor Fusion Strategies	29
2.5.1	Loosely-Coupled Fusion	30
2.5.2	Tightly-Coupled Fusion	30
2.5.3	Hybrid Approaches	31
2.6	SLAM Systems: From Filters to Graphs	31
2.6.1	Filter-Based SLAM	31
2.6.2	Graph-Based SLAM and Gmapping	31
2.7	Point Cloud Processing and Registration	32
2.7.1	Classical ICP and Variants	32
2.7.2	NDT and Probabilistic Approaches	33
2.7.3	Degeneracy and Corridor Geometry	33
2.8	Loop Closure Detection	33
2.9	Sensor Calibration	34
2.9.1	Extrinsic Calibration	34
2.9.2	Intrinsic Calibration	34
2.10	Path Planning and Navigation	34
2.10.1	Global Planning in Occupancy Grids	34
2.10.2	Local Planning and the Dynamic Window Approach	35
2.10.3	Costmaps, Inflation, and Dynamic Obstacles	35
2.11	Visual and Learning-Based Perception	36
2.11.1	Limitations of Classical Computer Vision	36
2.11.2	Visual-Inertial Odometry and Visual SLAM (Contextual)	36

2.11.3	Deep Learning for Object Detection: Two-Stage vs. One-Stage	36
2.11.4	Industrial Datasets and Missing-Component Detection	37
2.12	Research Trends and Gaps	38
3	Mechanical Design	40
3.1	Concept and Functional Design	40
3.2	System Architecture and Specifications	41
3.3	Mechanical Layout and Dimensions	42
3.4	Actuation System	43
3.5	Control and Electronic Drive System	44
3.6	Power Supply and Battery Selection	45
3.7	Structural Design and Prototyping	48
3.8	Physical Prototype and Structural Realization	50
3.8.1	Mechanical Framework and Assembly	50
3.8.2	Internal Layout and Stability	50
3.9	Experimental Platform and Operational Use	50
4	System Architecture and Software Framework	53
4.1	Introduction	53
4.1.1	Predictability and Real-Time Constraints	54
4.1.2	Scope and System Limitations	54
4.2	Software Decomposition and Node Graph	55
4.3	Computational Hardware and Environment	56
4.3.1	Why Edge Computing?	56
4.4	System Initialization and Boot Process	57
4.4.1	The Master Launch Sequence	57
4.5	Hardware–Software Interface (HAL)	58
4.5.1	Serial Communication Protocol	58
4.5.2	Differential Drive Kinematics	58
4.5.3	Empirical Payload Correction (Heuristic)	59
4.6	Low-Level Control Dynamics	59
4.6.1	Inertial Damping and Ramp Functions	59
4.7	Odometry and State Estimation	59
4.7.1	Runge-Kutta Integration and IMU Exclusion	60
4.8	Coordinate Frames and TF Tree	60
4.8.1	Live TF Tree Validation	60

4.8.2	Homogeneous Transformation Matrices	61
4.8.3	Frame Hierarchy	62
4.9	Perception and Sensor Processing	62
4.9.1	LiDAR Data Pipeline	62
4.10	Environment Representation (Costmaps)	63
4.10.1	Global Costmap	63
4.10.2	Local Costmap	63
4.10.3	Cost Inflation Model	63
4.11	Localization Architecture (AMCL)	63
4.11.1	Bayes Filter Formulation	63
4.11.2	KLD-Sampling for Resource Management	64
4.12	Navigation Planning Stack	64
4.12.1	Global Planner (Dijkstra)	64
4.12.2	Local Planner (Dynamic Window Approach)	64
4.13	Operational Workflow and Logic Integration	65
4.13.1	Phase 1: Mapping Sequence	65
4.13.2	Phase 2: Sterilization Mission Logic	66
4.13.3	Evaporator Integration and Timing	66
4.14	Safety Systems and Recovery Behaviors	67
4.14.1	Oscillation Suppression	67
4.14.2	Recovery Behaviors	67
4.14.3	Hierarchical Safety Protocols	68
4.15	Summary and System Specification	68
5	Mapping, Localization, and SLAM Implementation	70
5.1	Introduction	70
5.1.1	Problem Statement: The Minimalist Sensing Constraint . . .	70
5.2	Environment Mapping Strategy	70
5.2.1	Algorithm Selection: Why Gmapping?	71
5.2.2	Rao-Blackwellized Particle Filter (RBPF) Theory	71
5.2.3	Mapping Workflow and Tuning	72
5.3	Map Server and Post-Processing	73
5.3.1	Occupancy Grid Representation	75
5.3.2	Manual Post-Processing	75
5.4	Localization Architecture (AMCL)	75
5.5	Motion Model Configuration (No IMU)	76

5.5.1	The Alpha Parameters	77
5.5.2	Tuning for High Inertia and No Gyro	77
5.6	Sensor Model: Likelihood Field for A1 LiDAR	77
5.6.1	The Likelihood Field Model	78
5.6.2	Parameter Optimization	79
5.7	Handling Dynamic Obstacles in Localization	79
5.8	Experimental Validation and Performance Analysis	80
5.8.1	Localization Drift Analysis	80
5.8.2	Kidnapping and Recovery Test	81
5.9	Computational Complexity Analysis	81
5.10	Summary	81
6	Navigation, Planning, and Full ROS Stack Deployment	83
6.1	Introduction	83
6.2	Overview of the ROS Navigation Stack	84
6.2.1	Data flow and topic structure	84
6.2.2	<code>move_base</code> state machine	85
6.3	Global Costmap Design	86
6.3.1	Layer structure	86
6.3.2	Parameter configuration	87
6.4	Local Costmap (Rolling Window)	88
6.4.1	Window size and frame	88
6.4.2	Sensor integration and ray-tracing	88
6.5	Inflation Layer and Robot Footprint	90
6.5.1	Robot footprint model	90
6.5.2	Cost function and parameter relationship	91
6.5.3	Sensitivity to inflation parameters	91
6.6	Global Planner (NavFn)	92
6.6.1	Choice between Dijkstra and A*	92
6.6.2	Representative global path	92
6.7	Local Planner (Dynamic Window Approach)	93
6.7.1	Objective function	93
6.7.2	Dynamic window	94
6.8	Tuning of DWA Parameters	94
6.8.1	Core parameters	94
6.8.2	Observed effects of tuning	96

6.9	Path Tracking Behaviour	96
6.10	System Deployment and Configuration Strategy	98
6.10.1	Launch sequence	98
6.10.2	Controller frequency and internal consistency	99
6.11	Straight Line Tracking Analysis	99
6.11.1	Experimental Results and Temporal Convergence	100
6.11.2	Technical Discussion: Determinants of Asymptotic Stability	101
6.11.3	Functional Validation for Sanitization	102
6.12	Multi goal Navigation Tests	102
6.12.1	Experimental Setup and Test Geometry	103
6.12.2	Physical Measurement Methodology	104
6.12.3	Experimental Setup and Test Geometry	106
6.12.4	Localization accuracy: multi-loop navigation	107
6.12.5	Positional Accuracy and Error Distribution	113
6.12.6	Visual Documentation of Terminal Accuracy	113
6.12.7	Behaviour in presence of dynamic obstacles	116
6.13	Operational Performance and Safety Metrics	118
6.13.1	Temporal Consistency and Cycle Times	118
6.13.2	Error Distribution by Vertex (0–3)	119
6.13.3	Computational Resource Utilization and System Health	120
6.14	Limitations and Future Improvements	124
6.14.1	Kinematic Asymmetry and Lateral Slip Detection	124
6.14.2	Perceptual Horizons in 2D Space	125
6.14.3	Static Map Assumption in Dynamic Facilities	125
6.14.4	Glass and Specular Reflections	125
6.14.5	Optimization of Navigation Parameters	126
6.14.6	Conclusion of the Chapter	126
7	Advancing Automobile Manufacturing Inspections: YOLO-Based	
	Missing Part Detection	128
7.1	Introduction	128
7.2	Related Work	130
7.2.1	Two-Stage Detectors	130
7.2.2	Single-Stage Detectors	131
7.3	Methodology	131
7.3.1	Phase 1: Dataset Construction	132

7.3.2	Phase 2: Model Training and Optimization	133
7.3.3	Phase 3: Real-Time Deployment	134
7.4	YOLOv8 Architecture	134
7.4.1	Backbone: CSPDarknet with C2f Modules	135
7.4.2	Neck: Path Aggregation Network (PAN)	136
7.4.3	Head: Decoupled and Anchor-Free	136
7.4.4	Loss Function Formulation	136
7.5	Experimental Study	137
7.5.1	Training Configuration	137
7.6	Results and Discussion	137
7.6.1	Training Performance Analysis	137
7.6.2	Overall Performance Comparison	138
7.6.3	Class-wise Performance with Augmentation	139
7.6.4	Confusion Matrix Analysis	139
7.6.5	Generalization (Comparison with the NPU-BOLT Dataset) . .	140
7.6.6	Visual Validation	141
7.7	Real-Time Deployment and Validation	142
7.7.1	Real-time Implementation Results	143
7.7.2	Deployment Process	144
7.8	Conclusion	145
8	Conclusion and Future Work	146
8.1	Overview and Research Motivation	146
8.2	Summary of the Research	146
8.3	Thesis Contribution Statement	147
8.4	Discussion of Key Findings	148
8.5	Limitations	148
8.5.1	Navigation Limitations	148
8.5.2	Inspection Limitations	149
8.6	Future Work	149
8.6.1	Enhancing Navigation Capability	149
8.6.2	Machine Learning-Driven Extensions	149
8.7	Concluding Remarks	150
	References	150

List of Figures

2.1	Taxonomy of SLAM algorithms illustrating the evolution from filtering methods (EKF, particle filters) to modern graph-based optimization techniques.	31
2.2	Structural difference between single-stage detectors (YOLO) and two-stage detectors (R-CNN). YOLO processes the entire image in a single forward pass, enabling real-time performance suited to assembly lines.	37
3.1	Overall mechanical dimensions of the designed device.	43
3.2	BLDC hub motor selected for platform propulsion.	44
3.3	Motor control board with STM32 microcontroller and integrated drivers.	45
3.4	Bioreset Smart hydrogen peroxide generator.	47
3.5	LiFePO ₄ battery selected for the platform.	48
3.6	CAD design of the robotic platform.	49
3.7	Manufactured prototype of the robotic platform.	49
3.8	Physical realization of the robot’s mechanical framework. The assembly shows the differential-drive mobile base (Magni platform, welded rectangular steel profiles) topped by the custom-built payload superstructure constructed from 20×20 mm T-slot aluminum extrusions. Note that the lower structural chassis visible in Figure 3.6 is the welded steel skeleton of the base, while the vertical pillars shown here are the aluminum extrusion uprights that support the sterilization generator. These views represent the completed skeleton prior to the installation of sensors and the sterilization payload.	51
4.1	System Data Flow. Feedback (Blue) flows upward: Hardware → Localization → Navigation. Control (Red) flows downward: Navigation → Hardware. This closed-loop architecture ensures the planner always acts on the most recent pose estimate.	56
4.2	Live Transform (TF) tree captured during operation. The graph confirms the chain <code>map</code> → <code>odom</code> → <code>base_footprint</code> → <code>base_link</code> → <code>laser</code> , consistent with the architectural model.	61

4.3	Sterilization mission logic. At each waypoint, the robot navigates, stops, activates the inverter and nebulizer for a dwell period, then proceeds to the next waypoint or returns to dock.	66
5.1	The experimental corridor environment. The uniform surfaces and lack of distinct geometric landmarks require precise scan-matching to prevent odometric drift.	73
5.2	The complete static map generated using the Gmapping SLAM stack. Black pixels represent occupied space (walls, obstacles), while white pixels represent free traversable space. This high-fidelity map provides the geometric constraints required for the "Likelihood Field" sensor model.	74
5.3	The recursive cycle of the AMCL Particle Filter. Without an IMU, the "Motion Model" step introduces high uncertainty, which must be aggressively corrected by the "Sensor Model" step.	76
5.4	Visualization of Motion Model Noise. Blue represents tight tuning (with IMU). Red represents the "Banana Distribution" (non-Gaussian curved posterior) required for our system to capture the true pose despite encoder slip.	78
6.1	High-level data flow in the navigation stack. The global planner uses the static map and LiDAR information to compute a global path, while the local planner uses the pose estimate, local costmap, and global path to generate velocity commands.	85
6.2	Example costmap visualisation during navigation. The global costmap covers the entire environment and includes static obstacles and inflation. The local costmap forms a rolling window around the robot, focusing on nearby obstacles and free space.	90
6.3	Example global path on the static map. Dijkstra search finds a minimum-cost route that stays near the corridor centreline and respects inflated obstacle margins (visualized as orange contours). . . .	93

6.4	Screenshot during navigation. The global path (green line) runs along the corridor centreline. The robot pose is represented by the blue triangle. The local planner samples short forward candidate trajectories (red arcs) that follow the path while preserving clearance from walls. Note: if viewed in greyscale, the global path appears as the lightest continuous line and the candidate trajectories as darker short arcs.	97
6.5	13.0 m Trajectory analysis demonstrating recursive pose refinement. As the robot encounters static features, the AMCL particle filter distribution collapses, narrowing the uncertainty envelope.	101
6.6	Schematic of the 2.0 m $\times$ 2.0 m experimental square path. Vertex v_0 (bottom right) serves as the starting point. The green arrows indicate the counter clockwise (CCW) flow and the orange arrows indicate the clockwise (CW) flow.	104
6.7	Photographic documentation of representative terminal stopping poses and the associated manual measurement dataset. The printed grid and black rectangular target marker identify the vertex center and allow the manual recording of physical deflection (e_x, e_y) once the robot reaches its stopping pose.	105
6.8	Experimental setup for the Square Path Test. The trajectory is defined by four target vertices (V_0, V_1, V_2, V_3) forming a 2.0 m $\times$ 2.0 m square path. The robot executes a counter-clockwise traversal ($V_0 \rightarrow V_1 \rightarrow V_2 \rightarrow V_3 \rightarrow V_0$) and a clockwise traversal to isolate kinematic asymmetries.	106
6.9	Visual trajectory log of the counter-clockwise loops on the 2.0 $\times$ 2.0 m square path. Each coloured scatter cluster corresponds to one of the four goal vertices (V_0-V_3) , with the vertex label annotated adjacent to its cluster. Blue markers indicate the estimated robot stopping pose; the dashed grey circle around each cluster represents a 150 mm reference radius. Arrow annotations indicate the direction of CCW travel ($V_0 \rightarrow V_1 \rightarrow V_2 \rightarrow V_3 \rightarrow V_0$). An isolated outlier in loop 4 at V_0 (annotated) deviates from the cluster but is corrected by the subsequent AMCL update.	109

6.10	Schematic of the 2.0m $\times$ 2.0m experimental square path. v_0 serves as the starting and ending point. The green interior arrows indicate the counter-clockwise (CCW) flow ($v_0 \rightarrow v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow v_0$), while the orange arrows indicate the clockwise (CW) flow.	112
6.11	Visual trajectory log of the clockwise loops on the 2.0 $\times$ 2.0 m square path. Coloured scatter clusters represent the four goal vertices (V_0 – V_3), with each vertex label annotated at its cluster centroid. Orange markers at V_1 indicate CW-direction stopping poses; the dashed grey circle indicates a 150 mm reference radius at each vertex. Arrow annotations indicate the CW travel direction ($V_0 \rightarrow V_3 \rightarrow V_2 \rightarrow V_1 \rightarrow V_0$). The clustering of goal markers around each vertex illustrates the repeatability, though Vertex 1 (orange) shows systematic offsets outside the 150 mm reference radius during CW loops—a kinematic asymmetry discussed in Section 6.14.	112
6.12	Decomposition of navigation errors. (Left) Longitudinal error (e_{xc}) represents stopping accuracy. (Center) Lateral error (e_{yc}) reveals the systematic mechanical drift, particularly at CW Vertex 1. (Right) Heading error (e_θ) remains predominantly negligible, with a single outlier at 5.3° (CCW Vertex 2).	115
6.13	Sequential visualization of the dynamic obstacle avoidance test. (a) The robot identifies the blockage. (b) The DWA planner computes a valid curve around the inflation zone. (c) The robot executes the evasive maneuver. (d) The robot realigns with the global path to reach the destination.	117
6.14	Navigation duration analysis for all four vertices (0, 1, 2, 3). Annotations indicate the exact time in seconds. The asymmetry is evident: CW loops (right) show higher volatility, particularly at Vertex 1, confirming the impact of the mechanical drift identified in the positional analysis.	119
6.15	Total deflection error distribution for all four vertices (0, 1, 2, 3). The dashed red line indicates the 150 mm reference threshold. Vertex 1 in the CW phase exhibits both high variance and a median above this line, whereas Vertices 2 and 3 remain comparatively compact.	120

6.16	System Health Monitoring. The stable CPU load (Red) confirms the efficiency of the edge-computing architecture, while the clearance (Blue) confirms the safety inflation layer was never breached.	122
6.17	Heading error (e_θ) distribution by vertex and direction. Individual data points are plotted (diamonds) to reveal the density. The explicit annotation highlights the maximum observed deflection of 5.3° at Vertex 2 (CCW). Despite this outlier, the median error across all other waypoints remains tightly constrained within $\pm 2.0^\circ$	123
7.1	The comprehensive research methodology comprising three distinct phases. Phase 1 focuses on data acquisition and annotation from the physical assembly line. Phase 2 involves training the YOLOv8 model with extensive data augmentation. Phase 3 represents the deployment of the trained weights into a real-time inference loop for defect rejection.	132
7.2	The YOLOv8 Architecture. Key innovations include the C2f module in the backbone for richer gradient flow and the decoupled head structure for separate classification and regression tasks.	135
7.3	Training and validation metrics, including losses (box, classification, DFL) and performance metrics (precision, recall, mAP50) over 80 epochs. The convergence indicates effective feature learning.	138
7.4	Normalized Confusion Matrix for YOLOv8 Model Performance across Different Classes.	140
7.5	Comparison of Input and Output Images: (a) and (c) show the input images captured during the assembly process, while (b) and (d) show the corresponding output images with detected components highlighted.	142

List of Tables

2.1	Scope of Literature Review and Key Research Domains	25
2.2	Comparison of Sensor Fusion Architectures for Mobile Robotics . . .	30
3.1	Architectural Layers of the Autonomous System	41
3.2	Technical Specifications of the Mobile Robot	42
3.3	Technical Specifications of the Bioreset Smart Evaporizer	46
4.1	Hierarchical safety protocols implemented in the navigation stack. . .	68
4.2	System specification matrix: key configurations for the hardware–software interface.	69
5.1	Comparative analysis of 2D SLAM algorithms for the Magni platform	71
5.2	Sensor Specification Comparison justifying the Probabilistic Tuning .	78
5.3	AMCL Parameter Tuning for RPLiDAR A1 without IMU	79
6.1	Overview of main navigation stack components	84
6.2	Global costmap parameters	87
6.3	Local costmap parameters and configuration rationale	89
6.4	Key DWA parameters	95
6.5	Launch sequence for navigation stack	99
6.6	Detailed Localization Error Metrics - Counter-clockwise (CCW) Loops	108
6.7	Detailed Localization Error Metrics-Clockwise (CW) Loops	111
6.8	Summary of mean absolute lateral error $ e_y $ (mm) by vertex and direction, across all 5 loops. Values exceeding the 150 mm reference threshold are shown in bold.	113
7.1	Chapter 7 Overview: From Dataset Creation to Real-Time Deployment	130
7.2	Overall performance metrics comparing YOLOv8 and Faster R-CNN. (Bold indicates best).	138
7.3	Class-wise Performance Metrics with YOLOv8: Comparison With and Without Data Augmentation.	139
7.4	Performance comparison of YOLO using the MP dataset and the external NPU-BOLT dataset.	141
7.5	Real-time Implementation Results. QR (Quantity Removed), DA (Detection Accuracy averaged over 5 repeated trials per case). . . .	144

List of Acronyms

Acronym	Definition
AMCL	Adaptive Monte Carlo Localisation
AI	Artificial Intelligence
BOM	Bill of Materials
CCW	Counter-Clockwise
CNN	Convolutional Neural Network
COCO	Common Objects in Context (dataset)
CoG	Centre of Gravity
CPU	Central Processing Unit
CSP	Cross Stage Partial
CTE	Cross-Track Error
CW	Clockwise
DFL	Distribution Focal Loss
DWA	Dynamic Window Approach
EKF	Extended Kalman Filter
FPS	Frames Per Second
FSM	Finite State Machine
GPU	Graphics Processing Unit
HAL	Hardware Abstraction Layer
ICP	Iterative Closest Point
IESKF	Iterated Error-State Kalman Filter
IMU	Inertial Measurement Unit
IoU	Intersection over Union
LAN	Local Area Network
LiDAR	Light Detection and Ranging
MCB	Motor Control Board

Acronym	Definition
MEMS	Micro-Electro-Mechanical Systems
MP	Missing Parts (dataset)
mAP	Mean Average Precision
NAV	Navigation
NDT	Normal Distributions Transform
PAN	Path Aggregation Network
PDF	Probability Density Function
PID	Proportional–Integral–Derivative
ROS	Robot Operating System
RPN	Region Proposal Network
SSD	Single Shot MultiBox Detector
SLAM	Simultaneous Localisation and Mapping
ToF	Time of Flight
VFL	Varifocal Loss
VHP	Vaporised Hydrogen Peroxide
VIO	Visual-Inertial Odometry
YOLO	You Only Look Once

Chapter 1

Introduction

1.1 Background and Motivation

Industrial automation has advanced substantially over the past two decades; however, a persistent gap remains between academic robotic systems and solutions that can be reliably deployed, operated, and maintained in real industrial facilities. Many research prototypes assume access to rich sensor suites, high-performance embedded computation, and carefully controlled calibration procedures. While such assumptions are acceptable in laboratory environments, they rarely align with industrial practice, where systems must remain stable, predictable, and serviceable by personnel without specialised robotics or machine-learning expertise [82].

In industrial settings, operational reliability often outweighs algorithmic sophistication. Maintenance simplicity, ease of fault diagnosis, and repeatable behaviour across long operational periods are frequently more valuable than marginal performance gains achieved through complex sensing or tightly coupled multi-modal pipelines. This divergence between research priorities and deployment realities motivates a reconsideration of how autonomous systems are designed and evaluated when real-world operation, rather than methodological novelty, is the primary concern.

In this thesis, minimalist design refers to the deliberate restriction of sensing, modeling, and algorithmic complexity to the minimum required for reliable operation, with system performance evaluated in terms of robustness, repeatability, and deployability rather than optimality.

This thesis investigates two autonomous system problems that commonly arise in industrial environments:

1. autonomous navigation for a mobile platform operating in structured indoor spaces representative of pharmaceutical or controlled manufacturing facilities; and
2. automatic visual inspection for missing components on industrial assembly lines, with specific emphasis on automotive fuel-tank assemblies.

The two problems are addressed independently. They belong to different application domains, employ distinct sensing modalities, and are evaluated using separate

experimental setups. No attempt is made to integrate them into a single system or to propose a unified algorithmic framework. The first direction adopts a minimalist engineering philosophy in which sensing and algorithmic complexity are deliberately restricted to maximise reliability, interpretability, and maintainability in structured indoor navigation. The second direction addresses a separate verification problem whose objective is fundamentally different: high-confidence visual discrimination of small missing components in dense assemblies. For that reason, the inspection study is not evaluated by minimalist-navigation standards, but by the accuracy, robustness, and deployment feasibility of a high-fidelity learning-based vision system.

Rather than pursuing novelty for its own sake, the work focuses on extracting dependable performance through disciplined system design and empirical validation. In the navigation study, this takes the form of a deliberately minimal configuration built from well-understood components. In the inspection study, by contrast, dependable performance depends on a data-centric, high-fidelity deep-learning workflow in which detector capability, dataset quality, and deployment evaluation are treated as the central design variables.

In summary, the thesis brings together two independent research directions. On one side, a mobile robot navigates corridor-like indoor environments motivated by sterilisation and routine logistics tasks using a minimalist sensing philosophy. On the other hand, a vision system performs multi-class inspection on densely populated industrial assemblies using a high-fidelity learning-based approach. The continuity between them is therefore not a shared performance standard, but a shared commitment to engineering systems whose design choices are explicitly matched to their operational objective.

1.1.1 Autonomous Navigation in Sterilisation-Oriented Indoor Spaces

Pharmaceutical production areas and other controlled indoor environments require frequent and systematic environmental disinfection in order to maintain hygiene standards and ensure product safety. These procedures are traditionally carried out manually, which makes them labour-intensive, repetitive, and potentially hazardous, as personnel may be exposed to chemical disinfectants or biological contaminants. Mobile robotic platforms offer a natural means to automate portions of this workload; however, the physical and operational constraints of such facilities impose stringent requirements on navigation performance [83].

Typical layouts consist of narrow corridors and compact rooms, with nominal

corridor widths commonly ranging between 1.2 and 1.8 m. Fixed equipment, cabinets, or piping often protrude into the nominal free space, reducing effective clearance. The geometric structure of these environments is highly repetitive, dominated by long straight segments, right-angle intersections, and limited visual distinctiveness. From a perception standpoint, surfaces are frequently glossy or reflective, which degrades the reliability of vision-based depth estimation and visual SLAM pipelines. In addition, cleanliness requirements discourage exposed sensor assemblies, open cabling, or components that are difficult to sanitise. Integrating cameras or three-dimensional LiDAR sensors often necessitates protective housings, increasing mechanical complexity, platform size, and cost.

Operational demands further complicate the navigation problem. Sterilisation tasks typically require robots to execute the same routes repeatedly, often on a daily basis, traversing identical corridors and stopping at predefined locations where disinfection actions are applied. These stopping points impose implicit accuracy requirements: not only must the robot reach the correct location, but it must do so with consistent pose over extended periods of operation. As a result, the navigation subsystem must prioritise localisation stability, safety, and repeatability rather than aggressive motion, high speed, or short-term optimality.

Within this context, a single two-dimensional LiDAR sensor represents an attractive sensing modality. Two-dimensional LiDAR provides geometrically stable range measurements that are largely independent of ambient lighting conditions and surface texture. Calibration requirements are modest compared to multi-sensor systems, and the sensor can be integrated into a compact, sealed housing that aligns well with hygiene constraints. Two-dimensional LiDAR provides geometrically stable measurements that are generally more robust to ambient lighting than cameras. However, performance remains susceptible to specular reflections from industrial materials like stainless steel, which can introduce range-finding errors.

This leads to the central question addressed by the first research direction:

Can a carefully engineered, LiDAR-only navigation pipeline provide stable localisation and repeatable path tracking in structured indoor environments that approximate pharmaceutical layouts, without relying on inertial measurement units or complex sensor fusion?

The navigation work presented in this thesis does not claim deployment within certified cleanroom environments. Instead, it aims to demonstrate, in laboratory settings that replicate the spatial and operational characteristics of such facilities, that

a minimal sensing configuration—comprising a two-dimensional LiDAR and wheel odometry—can deliver the localisation consistency, obstacle avoidance capability, and smooth corridor-centred trajectories required by sterilisation-oriented robotic platforms [84]. The emphasis is placed on long-term stability, conservative safety margins, and repeatable behaviour across multiple navigation cycles.

1.1.2 Multi-Class Visual Inspection on Industrial Assembly Lines

In automotive and general manufacturing, seemingly minor assembly defects can lead to disproportionate downstream consequences. A single missing bolt, absent electrical connector, or improperly seated cap in a fuel-tank assembly may propagate through subsequent production stages, resulting in rework, production delays, costly recalls, or safety-critical failures in the field. Manual inspection remains widely used, but it is expensive, time-consuming, and inherently susceptible to human fatigue, particularly when components are small, visually similar, and densely arranged [85].

Conventional industrial machine-vision systems are typically designed around controlled lighting conditions, fixed camera viewpoints, and limited variability in part placement or orientation. Under such assumptions, techniques based on template matching, edge detection, or rule-based blob analysis can achieve high accuracy once carefully tuned. However, real assembly lines rarely maintain these idealised conditions. Practical deployments must contend with several complicating factors:

- multiple small components appearing in close proximity on a single assembly fixture;
- visually and geometrically similar parts, such as different bolt or screw types, that must be distinguished reliably;
- lighting variations across stations and over time due to ageing fixtures or environmental changes; and
- partial occlusion caused by cables, hoses, or adjacent structural elements.

These factors significantly reduce the robustness of traditional rule-based inspection pipelines and increase the effort required for reconfiguration when product variants or assembly conditions change. Learning-based object detectors, particularly those based on convolutional neural networks, provide a more flexible alternative. Architectures from the YOLO family offer single-stage detection with favourable

speed-accuracy trade-offs, making them attractive for inline inspection tasks where real-time operation is required.

Nevertheless, achieving deployment-ready performance requires more than selecting a modern detector architecture. The training dataset must capture the visual variability encountered on the target production line, including realistic lighting, pose variation, and clutter. In addition, inference speed must be compatible with existing hardware constraints, as reliance on high-end accelerators can limit practical adoption.

These considerations motivate the second research direction of this thesis:

Can a YOLO-based multi-class detector, trained on a dedicated industrial dataset, reliably detect missing fuel-tank components at real-time speeds, and how robust is its performance under realistic variations in lighting, pose, and assembly configuration?

The focus is not solely on peak detection accuracy under ideal conditions, but on robustness, failure modes, and performance stability when deployed under conditions that reflect real manufacturing environments.

1.1.3 Integrative Motivation: Simplicity as a Design Strategy

Although the navigation and inspection problems addressed in this thesis belong to different domains, they are united by a common observation drawn from industrial practice: systems that are complex to configure, calibrate, or diagnose are difficult to deploy and sustain, even when they demonstrate strong performance in controlled experiments. Multi-sensor robotic platforms that combine three-dimensional LiDAR, multiple RGB-D cameras, and tightly coupled visual-inertial pipelines may achieve impressive results, but they also introduce a dense web of dependencies that can become fragile outside research settings.

This motivates a design philosophy centred on *simplicity, transparency, and robustness*. Rather than maximising sensor diversity or algorithmic novelty, the work in this thesis deliberately adopts minimal sensor configurations that are easy to mount, protect, and maintain. Established algorithmic components—such as Adaptive Monte Carlo Localization (AMCL) and standard Robot Operating System (ROS) planners for navigation, and YOLO-based detectors for vision—are favoured over experimental or opaque frameworks.

The central claim is not that minimal systems universally outperform more complex alternatives. Instead, the thesis argues that, in many industrial contexts, *well-engineered simple systems can offer a more favourable balance between performance, robustness, and operational cost*. By treating complexity as an explicit design trade-off rather than an implicit goal, the work seeks to clarify what levels of performance are achievable when deployment constraints are taken seriously.

1.2 Industrial Context and Historical Evolution

1.2.1 Autonomous Navigation in Structured Environments

The development of autonomous navigation in industrial environments has progressed through several distinct phases, each shaped by prevailing technological constraints and operational requirements. Early mobile robotic systems relied on simple navigation strategies such as line-following, beacon-based localisation, or motion constrained by discrete fiducial markers embedded in the environment. These approaches proved effective in highly structured and purpose-built settings, including early automated warehouses and factory floors, but they offered limited flexibility and required significant infrastructure modification to support navigation.

The emergence of probabilistic robotics and simultaneous localisation and mapping (SLAM) techniques marked a fundamental shift in both research and industrial practice. By modelling uncertainty explicitly, robots became capable of mapping and localising within previously unknown environments using onboard sensors. This transition encouraged the adoption of sensor-rich platforms equipped with laser scanners, cameras, inertial sensors, and, later, three-dimensional LiDAR systems. While these approaches significantly improved autonomy and adaptability, they also increased system complexity and dependence on careful calibration and environmental conditions.

In parallel with academic developments, industrial automated guided vehicles (AGVs) evolved into automated mobile robots (AMRs) that rely primarily on LiDAR-based navigation. Modern AMRs commonly use two-dimensional LiDAR sensors for localisation, obstacle detection, and safety monitoring in warehouses and production facilities. These environments are well suited to LiDAR sensing due to their regular geometry, long straight aisles, and static structural features such as shelving. In such contexts, 2D LiDAR-based localisation has proven to be both reliable and cost-effective, leading to widespread industrial adoption.

Pharmaceutical and healthcare facilities occupy a distinct position within this

evolutionary landscape. While their geometric structure resembles that of warehouses, with corridors, intersections, and repetitive layouts, they impose substantially stricter cleanliness, safety, and operational constraints. Corridor widths are often narrower, available clearance is reduced by fixed installations, and access to multi-sensor payloads may be restricted by hygiene requirements or enclosure limitations. Off-the-shelf navigation solutions frequently assume wider passages, richer geometric features, or the integration of cameras and inertial sensors that are difficult to deploy and maintain in sterile or semi-sterile environments.

As a result, navigation strategies that perform well in warehouses may exhibit degraded stability or overly conservative behaviour when transferred directly to pharmaceutical layouts. Bridging this gap does not necessarily require additional sensing modalities; instead, it calls for a careful re-examination of two-dimensional LiDAR-based navigation under conditions of limited observability, long straight corridors, and stringent repeatability requirements. This thesis situates its navigation work within this context, focusing on extracting robust and predictable behaviour from minimal sensing configurations through systematic parameter engineering and empirical validation.

1.2.2 Machine Vision for Industrial Inspection

The evolution of industrial machine vision has followed a trajectory similar to that of mobile robotics. Early inspection systems were dominated by hand-crafted features, binary thresholding, and geometric rules applied to tightly controlled visual scenes. With fixed lighting, static camera viewpoints, and limited product variability, these approaches delivered reliable performance. Their primary limitation lay in brittleness: even modest changes in illumination, part appearance, or camera alignment often required extensive re-tuning by domain experts.

The rise of convolutional neural networks introduced a shift from manually engineered features to data-driven visual representations. Large-scale datasets such as ImageNet demonstrated the effectiveness of deep learning for visual recognition tasks, and this success translated into object detection architectures including Faster R-CNN, SSD, and the YOLO family. These models enabled real-time detection while accommodating greater variability in appearance, pose, and background context.

Despite these advances, most publicly available datasets and benchmarks remain focused on everyday objects, outdoor scenes, or natural imagery. Industrial inspection tasks present a different set of challenges: components are often small, visually

similar, densely packed, and observed under non-uniform lighting. The scarcity of open datasets representing such conditions has limited systematic comparison of detection methods and slowed the transfer of research results into industrial practice.

Industry adoption of deep learning for inspection has therefore proceeded in a fragmented manner. Solutions are frequently developed for specific production lines, with proprietary datasets and tightly customised models that are rarely published or reused. While effective in isolation, this approach hinders reproducibility and makes it difficult to assess generalisation across different assembly configurations or facilities.

This thesis contributes to this landscape by addressing missing-component detection within a concrete industrial domain: automotive fuel-tank assemblies. By constructing a dedicated dataset and evaluating a multi-class detector under realistic conditions, the work aims to clarify the practical capabilities and limitations of learning-based inspection systems in dense industrial assemblies, rather than proposing a general-purpose vision framework.

1.3 Fundamental Technical Challenges

The two research directions addressed in this thesis are driven by distinct application domains, yet they are united by a common set of technical challenges that extend beyond routine implementation details. In both cases, the core difficulty lies not in the availability of algorithms, but in ensuring stability, robustness, and repeatability when those algorithms are deployed under realistic industrial constraints.

1.3.1 Challenges in LiDAR-Only Indoor Navigation

A navigation system based exclusively on two-dimensional LiDAR sensing and wheel odometry must operate within a constrained perceptual regime. In structured indoor environments dominated by corridors and right-angle intersections, several fundamental challenges arise.

- **Limited observability in feature-poor geometries:** Long, straight corridors with parallel walls generate range measurements that change slowly as the robot moves along the corridor axis. In such settings, translational and rotational errors can remain weakly observable for extended periods. As a result, probabilistic localisation methods such as AMCL may struggle to correct accumulated drift, particularly when motion and sensor noise are comparable

in magnitude to the geometric variations observed by the LiDAR.

- **Planar sensing constraints:** A two-dimensional LiDAR provides information only within a single horizontal scan plane. Obstacles above or below this plane are invisible, and vertical structure is not captured. This limitation places strict assumptions on the environment, including obstacle placement, floor flatness, and the absence of overhanging structures within the robot’s operating height. Navigation safety therefore, depends not only on algorithmic configuration but also on mechanical design choices such as sensor height and robot footprint.
- **Heading drift without inertial sensing:** In the absence of an inertial measurement unit, rotational motion is inferred from wheel encoders and corrected indirectly through LiDAR-based localisation updates. Wheel slip, uneven flooring, or minor contacts with obstacles can introduce heading errors that propagate along long corridor segments. If these errors are not corrected promptly, they lead to systematic lateral drift, degraded path tracking, and increased interaction with costmap boundaries.
- **Interaction between costmap inflation and narrow passages:** Safety in human-accessible environments requires conservative obstacle inflation; however, in narrow corridors this directly conflicts with navigability. Inflation radii and cost scaling factors must be tuned to preserve sufficient clearance while avoiding the artificial closure of otherwise traversable paths. Small parameter changes can shift the system from smooth corridor-centred motion to oscillatory behaviour or repeated recovery events.
- **Stability under repeated multi-loop operation:** Unlike short demonstration runs, industrial navigation tasks often involve repeated execution of identical routes over extended periods. The navigation system must therefore exhibit long-term stability, with no systematic accumulation of localisation error and no escalation in recovery behaviors across successive loops. This requirement exposes weaknesses that may remain hidden during single-pass experiments.

Addressing these challenges requires a coordinated treatment of mapping, localisation, costmap construction, and local planning. In particular, the interaction between AMCL parameterisation, costmap inflation policies, and planner dynamics

must be understood as a coupled system rather than as independent components. The methodology and experimental validation of these interactions are developed in Chapters 5 and 6.

1.3.2 Challenges in Multi-Class Missing-Component Detection

The visual inspection task considered in this thesis introduces a different, yet equally fundamental, set of challenges rooted in the characteristics of industrial assemblies and production constraints.

- **Fine-grained inter-class similarity:** Many component classes differ only subtly in geometry, size, or surface texture, such as variations among screws, bolts, or fasteners. Reliable discrimination requires the detector to learn fine-scale visual cues while remaining robust to noise, reflections, and manufacturing tolerances.
- **Small object representation:** Components of interest often occupy a small fraction of the image area, particularly when captured by fixed overhead cameras covering an entire assembly fixture. This places pressure on spatial resolution, feature pyramid design, and anchor or prediction strategies, as small localisation errors can translate into missed detections.
- **Clutter and partial occlusion:** Industrial assemblies are inherently dense. Components are embedded among hoses, cables, brackets, and structural elements, leading to frequent partial occlusion. The detector must therefore rely on incomplete visual evidence and contextual cues rather than isolated object appearance.
- **Domain shift between training and deployment:** Variations in lighting conditions, camera calibration, component batches, and surface wear introduce distribution shifts between training data and deployment scenarios. Models that perform well under controlled validation conditions may degrade when exposed to these variations, making robustness and generalisation central concerns.
- **Real-time operational constraints:** Inspection systems must operate at frame rates compatible with production line speeds. Excessive inference latency can necessitate hardware upgrades or reduce throughput, limiting prac-

tical adoption. Performance must therefore be evaluated jointly in terms of accuracy and computational efficiency.

Together, these factors make missing-component detection in real industrial assemblies substantially more demanding than conventional object detection benchmarks. They motivate the construction of a dedicated dataset and a systematic evaluation of detector performance under realistic conditions, as reported in Chapter 7.

1.4 Why Existing Approaches Are Insufficient

The preceding sections have introduced the industrial contexts and the specific technical challenges they impose. This section synthesises those observations into a coherent argument for why the prevailing approaches in the literature are not well-suited to the scenarios studied here—providing the explicit motivation for the minimalist design strategy adopted throughout this thesis.

1.4.1 Limitations of Sensor-Rich Navigation Pipelines

State-of-the-art autonomous navigation systems frequently integrate three-dimensional LiDAR, RGB-D cameras, inertial measurement units, and tightly coupled SLAM frameworks. Such pipelines can deliver high localisation accuracy in visually rich or unstructured environments; however, they are not always appropriate for the class of industrial scenarios considered in this thesis.

First, sensor-rich platforms are typically more expensive and physically larger than systems based on a single two-dimensional LiDAR. In narrow indoor corridors, additional sensors and protective housings reduce available clearance and complicate mechanical integration. Second, these pipelines rely on precise multi-sensor calibration. Calibration parameters can drift over time due to temperature variation, mechanical vibration, or minor impacts, leading to degraded localisation performance and increased maintenance requirements.

Camera-based components introduce further sensitivity to environmental conditions. Variations in illumination, reflective surfaces, and partial occlusions can adversely affect depth estimation and visual feature extraction, particularly in corridor-dominated environments with limited texture. From a deployment perspective, retrofitting existing industrial corridors to accommodate such sensor suites may be infeasible due to space limitations, hygiene requirements, or regulatory constraints.

Even within the ROS ecosystem, many reference implementations and default parameter configurations implicitly assume wide operating spaces and feature-rich environments. When applied directly to corridors with widths on the order of 1.2 m, these configurations often produce oscillatory motion, excessive proximity to walls, or frequent activation of recovery behaviours. These issues reflect a mismatch between assumed operating conditions and actual deployment constraints rather than inherent deficiencies of the underlying algorithms.

1.4.2 Limitations of Conventional Industrial Vision Systems

Commercial industrial inspection systems commonly rely on rule-based pipelines that are tuned to specific stations and product variants. Under stable and controlled conditions, such systems can achieve high precision; however, their limitations become pronounced in more variable inspection scenarios.

Changes in lighting conditions, camera placement, or component batches often necessitate extensive re-tuning by experienced personnel. As assembly complexity increases, manually engineering rules for multi-class detection of small, visually similar components becomes increasingly difficult and error-prone. Moreover, traditional inspection systems provide limited support for reuse or generalisation across different assembly lines, constraining scalability.

Although deep-learning-based inspection methods address many of these shortcomings, research prototypes frequently overlook deployment constraints. Assumptions of access to high-performance GPUs and disregard for inference latency limit practical applicability. For adoption on existing production lines, detectors must operate in real time on modest hardware and be trainable using datasets that can be realistically collected and annotated within an industrial setting.

1.5 Theoretical Rationale for Minimalist System Design

Building on the motivation established in Sections 1.1 and 1.4, this section formalises the theoretical basis for the minimalist design paradigm adopted in this thesis. The argument proceeds from system-theoretic principles and establishes why simplicity—when applied with rigour—is not a compromise but a deliberate engineering strategy.

The emphasis on simple hardware configurations and well-understood algorithms in this thesis is not merely pragmatic; it is grounded in fundamental considerations of system reliability, observability, and controllability. In autonomous systems intended for long-term operation, the ability to reason about system behaviour and failure

modes is as important as nominal performance.

As system complexity increases, the dimensionality of the underlying state space grows with each additional sensor, processing layer, and calibration dependency. This expansion increases the number of possible failure modes, including sensor misalignment, calibration drift, synchronisation errors, and unexpected interactions between subsystems. Diagnosing such failures becomes progressively more difficult as dependencies multiply, and the mean time to repair correspondingly increases. From a systems perspective, increased complexity often trades short-term performance gains for reduced transparency and maintainability.

By contrast, minimalist system designs exhibit several advantageous properties:

- fewer hidden states and fewer interdependent calibration parameters;
- greater transparency in the relationship between sensor measurements and system behaviour;
- more predictable and interpretable failure modes; and
- a reduced parameter space that enables focused and systematic optimisation.

In the navigation domain, restricting the sensing configuration to a two-dimensional LiDAR and wheel encoders necessarily limits the available perceptual information. At the same time, it yields a localisation and planning pipeline whose behaviour can be analysed in detail. The interactions between AMCL parameterisation, costmap inflation, and local planner dynamics can be examined explicitly, allowing stability and repeatability to be addressed through principled parameter tuning rather than through additional sensing layers. This analytical transparency underpins the experimental investigations reported later in the thesis.

In the visual inspection domain, the rationale is different. The task is not to minimise sensing or model capability, but to achieve reliable, high-accuracy verification of small components under realistic imaging variability. Accordingly, the inspection study adopts a high-fidelity deep-learning workflow centred on detector selection, dataset construction, generalisation analysis, and deployment testing. Rather than treating architectural capacity as an avoidable burden, this part of the thesis treats model capability as a necessary resource whose adequacy must be demonstrated empirically for the verification task at hand.

The thesis therefore treats simplicity not as a constraint imposed by budget or hardware availability, but as an explicit design choice motivated by the goal of

constructing systems that are understandable, diagnosable, and maintainable under realistic industrial conditions.

1.6 Research Questions

The research questions of this thesis follow directly from the technical challenges and design rationale outlined above. They are organised into two independent groups corresponding to the two research directions, and each group is evaluated against the requirements of its own domain rather than against a single common technical standard.

Navigation-Related Research Questions

RQ1: LiDAR-Only Localisation Feasibility. Can a navigation pipeline based on two-dimensional LiDAR and wheel odometry achieve stable localisation and path tracking in long, corridor-like environments without inertial sensing?

RQ2: Parameter Regimes for Stability. Which combinations of AMCL parameters, costmap configurations, and Dynamic Window Approach (DWA) settings yield stable and repeatable navigation in narrow corridors, and how sensitive is system performance to these parameters?

RQ3: Loop-Level Robustness. How does the navigation system behave under repeated multi-loop operation, and does performance degrade due to cumulative drift or intermittent disturbances such as wheel slip?

RQ4: Dynamic Obstacle Handling. To what extent can the configured navigation stack respond safely to typical dynamic obstacles, such as personnel crossing corridors, while preserving mission completion?

Inspection-Related Research Questions

RQ5: Multi-Class Detection Performance. Can a YOLO-based detector trained on a dedicated industrial dataset achieve reliable precision and recall across multiple small, visually similar component classes in fuel-tank assemblies?

RQ6: Real-Time Operation. Is it possible to deploy such a detector at frame rates compatible with inline inspection on modest hardware?

RQ7: Robustness and Generalisation. How robust is the detector to realistic variations in lighting, pose, and assembly configuration, and to what extent does it generalise across different datasets?

RQ8: Failure Modes. What failure modes dominate in practical deployment, and what implications do they have for future system refinement and deployment strategy?

1.7 Research Objectives

The research objectives translate the research questions into concrete and verifiable goals. They define the scope of system design, experimental validation, and empirical analysis carried out in this thesis.

1.7.1 Objectives for Autonomous Sterilisation Navigation

1. **Minimalist System Design and Implementation:** Design and implement a complete autonomous navigation system for corridor-like environments using a differential-drive platform equipped solely with a two-dimensional LiDAR and wheel encoders, and deploy the system on a resource-constrained embedded platform.
2. **Localisation Stability Under Repetition:** Achieve localisation stability sufficient for repeated navigation of multi-vertex routes, ensuring that pose errors at designated stopping points remain within corridor-scale tolerances across multiple navigation cycles.
3. **Safe Obstacle Avoidance in Human-Accessible Spaces:** Ensure reliable real-time obstacle avoidance for both static and dynamic obstacles, maintaining conservative safety margins appropriate for environments shared with human personnel.
4. **Smooth Navigation in Narrow Corridors:** Demonstrate stable and smooth motion in corridors with widths between 1.2 and 1.8 m, avoiding oscillatory behaviour, wall hugging, or excessive reliance on recovery behaviours.
5. **Long-Term Operational Robustness:** Validate navigation performance under extended multi-loop operation, quantifying localisation drift, trajectory consistency, and behavioural stability across both clockwise and counter-clockwise routes.

6. **Mechanical-Algorithmic Interaction Analysis:** Analyse the influence of mechanical design factors, including robot footprint, wheel characteristics, and sensor placement, on algorithmic parameter selection and overall navigation performance.
7. **Reusable Configuration and Deployment Guidelines:** Provide documented and reusable parameter configurations for AMCL, costmap layers, and local planners that can be adapted to similar corridor-based industrial environments.

1.7.2 Objectives for Multi-Class Missing-Component Detection

1. **Industrial Dataset Construction:** Construct a realistic, annotated dataset of fuel-tank assemblies containing multiple small component classes, captured under representative lighting conditions and geometric configurations.
2. **Multi-Class Detector Development:** Train a YOLO-based multi-class object detector on the constructed dataset, targeting reliable precision and recall across all component classes.
3. **Benchmark Evaluation:** Benchmark the YOLO-based detector against a strong baseline model, such as Faster R-CNN, using standard detection metrics under matched training and evaluation protocols.
4. **Real-Time Deployment Feasibility:** Demonstrate real-time inference performance on hardware representative of practical industrial deployments, without reliance on high-end accelerators.
5. **Robustness and Generalisation Assessment:** Evaluate detector robustness to variations in lighting, pose, and assembly configuration, including cross-dataset testing on external assemblies to assess generalisation capability.
6. **Failure Mode Characterisation:** Identify and analyse dominant failure modes, providing guidance on data requirements, model refinement, and system-level safeguards needed for deployment.

1.7.3 Integrative Objectives

1. **Demonstration of Pragmatic Engineering Principles:** Demonstrate, through two independent case studies, that robust autonomous systems can

be realised using simple hardware configurations and established algorithms when supported by careful parameter engineering, integration, and empirical validation.

2. **Provision of Reproducible Artefacts:** Deliver datasets, configuration files, and implementation details that enable reproducibility and facilitate extension of the presented work.
3. **Establishment of Transferable Methodological Patterns:** Formulate methodological patterns encompassing requirement analysis, parameter tuning, and empirical evaluation that can inform the development of autonomous systems in other industrial automation contexts.

1.8 Scope, Assumptions, and Limitations

1.8.1 Scope and Assumptions for Autonomous Navigation

The autonomous navigation study is conducted under a set of explicit assumptions that define the intended operating regime and delimit the scope of applicability of the results.

- The environment is a structured indoor space characterised by corridor-like geometry and a static layout over the duration of the experiments.
- A reliable static map of the environment is available; online SLAM, map adaptation, or long-term map maintenance during navigation are not considered.
- The mobile platform employs a differential-drive configuration and operates on reasonably flat indoor flooring with adequate, though not ideal, traction.
- Perception is restricted to a two-dimensional LiDAR and wheel encoders; inertial sensors, cameras, and three-dimensional LiDARs are deliberately excluded in order to maintain a minimal sensing configuration.
- Dynamic obstacles are limited to occasional human traffic and movable objects such as trolleys; dense crowds or highly dynamic environments fall outside the scope of this study.

These assumptions define a class of environments representative of many corridor-based industrial and institutional facilities. At the same time, they exclude outdoor

navigation, environments with rapidly changing layouts, and scenarios requiring explicit reasoning about complex three-dimensional structure. The conclusions drawn from the navigation experiments should therefore be interpreted within this clearly defined operational context.

1.8.2 Scope and Assumptions for Visual Inspection

The visual inspection study is similarly constrained by design choices that reflect practical industrial considerations.

- Input data consist exclusively of RGB images acquired from a fixed overhead camera; alternative sensing modalities such as infrared imaging or depth sensing are not employed.
- The inspection task is limited to a single assembly type, namely automotive fuel-tank assemblies, and a predefined set of component classes relevant to that assembly.
- Data acquisition is performed in a controlled yet realistic laboratory setup that emulates an industrial inspection station, rather than a fully operational production line with all peripheral equipment and process variability.

The resulting dataset captures the visual characteristics of one facility configuration over a specific time period. While cross-dataset evaluation is performed to assess generalisation, the work does not attempt to encompass the full diversity of industrial assembly lines or inspection conditions. The findings are intended to inform similar inspection scenarios rather than to serve as a universal solution for all industrial vision tasks.

1.9 Research Methodology Overview

1.9.1 Methodology for Navigation Experiments

The navigation experiments follow an iterative and evidence-driven methodology designed to expose both nominal performance and failure modes under realistic operating conditions. Rather than treating navigation as a monolithic capability, the methodology isolates and evaluates individual subsystems before assessing integrated behaviour.

1. **Requirement Analysis:** Translate physical and operational constraints, including corridor widths, turning radii, and safety expectations, into design limits on robot footprint, sensor placement, velocity bounds, and acceleration profiles.
2. **Baseline Configuration:** Deploy a baseline ROS navigation stack comprising AMCL, a global planner, and the Dynamic Window Approach (DWA) local planner, using conservative default parameters to establish a stable reference configuration.
3. **Targeted Subsystem Experiments:** Conduct structured experiments that isolate individual components of the navigation stack, such as localisation performance, costmap behaviour, and local planner response, in order to identify dominant sensitivities and interactions.
4. **Multi-Loop Route Trials:** Execute repeated navigation loops across multi-vertex routes in both clockwise and counter-clockwise directions, logging pose estimates, goal errors, path deviations, and obstacle clearance.
5. **Analysis and Parameter Refinement:** Analyse recorded data to quantify performance metrics, including error distributions, recovery behaviour frequency, and computational load, and iteratively refine parameters based on empirical evidence.
6. **Dynamic Obstacle Evaluation:** Introduce controlled dynamic obstacles, such as human crossings, to assess obstacle avoidance behaviour, recovery stability, and the impact of disturbances on localisation and path tracking.

This methodology allows performance limitations to be traced to specific configuration choices rather than to aggregate system behaviour. Detailed implementation and results are presented in Chapters 5 and 6.

1.9.2 Methodology for Inspection Experiments

The inspection experiments combine dataset engineering with systematic detector evaluation, with emphasis on robustness and deployment feasibility rather than peak benchmark performance.

1. **Data Acquisition and Annotation:** Capture RGB images of fuel-tank assemblies under representative lighting and geometric conditions, followed by manual annotation of component instances for all target classes.
2. **Dataset Partitioning:** Divide the dataset into training, validation, and test subsets in a manner that preserves realistic variability in component appearance, pose, and scene composition.
3. **Model Training:** Train YOLO-based detectors and baseline models under controlled and comparable hyperparameter settings to ensure fair evaluation.
4. **Quantitative Performance Evaluation:** Evaluate trained models using standard metrics, including precision, recall, mean average precision (mAP), and inference time.
5. **Cross-Dataset Generalisation Testing:** Assess detector robustness by evaluating performance on external datasets representing different assembly configurations or imaging conditions.
6. **Real-Time Deployment Trials:** Execute the detector within a real-time processing loop on representative hardware, simulating inline inspection scenarios and monitoring detection outcomes in cases of missing components.

The inspection methodology and experimental results are described in detail in Chapter 7.

1.10 Scientific Significance and Real-World Impact

The scientific contribution of this thesis lies primarily in demonstrating how established methods can be engineered into reliable and deployable systems when subjected to realistic constraints. Rather than proposing new algorithms, the work clarifies the performance boundaries and trade-offs associated with task-matched engineering choices: minimalist sensing and explicit parameter control for navigation, and high-fidelity learning-based perception for visual inspection.

- The navigation study demonstrates that two-dimensional LiDAR-based localisation and planning, when carefully configured, can satisfy the stability and repeatability requirements of corridor-like indoor environments without

reliance on inertial sensing or complex sensor fusion. This provides a concrete reference for practitioners designing navigation systems under similar constraints.

- The inspection study contributes a dedicated dataset, a systematic evaluation of a multi-class detector, and an analysis of robustness and failure modes that is directly relevant to industrial deployment of learning-based inspection systems.

From an application perspective, the findings suggest two principal impacts:

- reduced cost and integration complexity for autonomous sterilisation or logistics robots operating in narrow indoor environments; and
- improved feasibility of data-driven inspection for complex assemblies without reliance on highly customised or vendor-locked solutions.

1.11 Contributions of the Thesis

The thesis makes a set of concrete contributions to autonomous navigation and industrial visual inspection. These contributions are organised as two independent technical directions: a minimalist navigation study for industrial reliability, and a separate high-fidelity inspection study for complex visual verification.

1.11.1 Contributions to Autonomous Navigation

- A validated minimalist ROS navigation configuration for corridor-like indoor environments based exclusively on two-dimensional LiDAR and wheel odometry, including fully documented and reproducible parameter sets for AMCL, costmap layers, and local planners.
- A detailed empirical characterisation of navigation behaviour under repeated multi-loop operation, covering clockwise and counter-clockwise trajectories, with quantitative analysis of localisation error distributions, loop-to-loop stability, and response to dynamic obstacles.
- An explicit analysis of the coupling between mechanical design choices, such as robot footprint, wheel characteristics, and sensor placement, and algorithmic configuration, providing practical guidance for the design and deployment of similar mobile robotic platforms.

1.11.2 Contributions to Visual Inspection

- A dedicated and publicly reproducible annotated dataset of automotive fuel-tank assemblies containing multiple small component classes, captured under realistic industrial imaging conditions.
- A systematically evaluated YOLO-based multi-class detector tailored to missing-component detection, including benchmark comparison against a strong baseline detector under matched training and evaluation conditions.
- A robustness and failure-mode analysis, incorporating cross-dataset evaluation, that clarifies the operational limits, dominant error sources, and generalisation behaviour of learning-based inspection systems in dense industrial assemblies.

1.11.3 Integrative Contributions

- A demonstration, through two independent case studies, that robust industrial systems can emerge from different technical philosophies when each is matched to its operational objective: minimalist sensing and explicit parameter engineering for navigation, and high-fidelity learning-based perception for visual inspection.
- The formulation of transferable methodological patterns for requirement analysis, parameter tuning, and empirical evaluation that can inform the development of autonomous systems in other industrial automation contexts.

1.12 Structure of the Thesis

The remainder of the thesis is organised as follows.

- **Chapter 2** reviews the relevant literature on autonomous mobile robotics, minimalist sensing strategies, LiDAR-based localisation and planning, and deep-learning methods for industrial inspection, with emphasis on multi-class object detection.
- **Chapter 3** describes the mechanical and hardware design of the mobile platform, including robot footprint, sensor placement, and physical factors influencing navigation performance.

- **Chapter 4** presents the overall software architecture and ROS-based implementation, detailing node interactions, coordinate frames, and safety mechanisms.
- **Chapter 5** addresses environment representation and localisation using AMCL on pre-mapped indoor layouts, including mapping procedures and localisation parameter tuning.
- **Chapter 6** details the design, implementation, and experimental validation of the navigation stack, including costmap configuration, local planning behaviour, and multi-loop endurance experiments.
- **Chapter 7** introduces the Missing Parts dataset and presents the training, evaluation, and robustness analysis of the YOLO-based multi-class missing-component detector.
- **Chapter 8** summarises the main findings, discusses limitations, and outlines directions for future work in both navigation and visual inspection.

Chapter 2

Literature Review

2.1 Introduction

The pursuit of autonomous mobile robotics is fundamentally grounded in the ability of a system to perceive its environment, estimate its state, and actuate to achieve specific goals without human intervention. This chapter surveys the foundational and contemporary literature relevant to the two research pillars of this thesis: (i) autonomous navigation in structured, sterilization-oriented indoor environments and (ii) visual inspection for missing-component detection in manufacturing.

The literature spans several interconnected domains: sensing and sensor selection, multi-sensor fusion, Simultaneous Localization and Mapping (SLAM), point cloud registration, loop closure, calibration, path planning and navigation, and deep-learning-based perception for industrial inspection. Rather than attempting an exhaustive survey, the focus is on works that shape the design decisions and constraints of the systems developed in this thesis. In particular, priority is given to approaches that have demonstrable relevance to real-world deployments, operate under hardware or environmental constraints similar to the target use cases, or embody a “minimalist but robust” philosophy.

Table 2.1 categorizes the literature reviewed in this chapter and links each domain to its specific relevance for the research contributions.

Table 2.1: Scope of Literature Review and Key Research Domains

Domain	Key Topics Reviewed	Relevance to Thesis
Sensing	Minimalist Architectures, 2D vs. 3D LiDAR, Vision Sensors, IMU Drift	Justifies the selection of a single 2D LiDAR setup and the deliberate exclusion of vision for navigation.
Fusion	Loosely vs. Tightly Coupled Architectures, Kalman Filters, Factor Graphs	Contextualizes the decision to avoid LiDAR-IMU fusion and to rely on LiDAR + odometry only.
SLAM	EKF-SLAM, FastSLAM, Gmapping, Graph-SLAM, Loop Closure	Bases the mapping strategy on RBPF (Gmapping) and clarifies why full 3D graph-SLAM is not pursued.
Planning	A*, Dijkstra, Dynamic Window Approach (DWA), Potential Fields, Costmaps	Informs the tuning of the ROS local planner and costmap inflation for narrow corridors.
Perception	YOLO vs. R-CNN, CNN Architectures, Real-time Inference, Industrial Datasets	Supports the choice of YOLOv8 for multi-class assembly inspection and highlights dataset gaps.

The following sections first revisit the core concepts and algorithms that underpin autonomous mobile robotics, then narrow down to minimalist sensing and LiDAR-based navigation in structured indoor environments. The second half of the chapter turns to learning-based visual perception for industrial inspection, emphasising object detection architectures and their suitability for missing-component detection.

2.2 Foundations of Autonomous Mobile Robotics

The capability to operate without human intervention is especially critical in industrial domains where robots must operate alongside human personnel, within physical constraints imposed by existing infrastructure, or in sensitive environments such as pharmaceutical facilities. Foundational work in mobile robotics established the for-

mal structure of the perception-action loop and introduced probabilistic reasoning as a first-class citizen in robot decision-making.

Early research in the late 20th century established the probabilistic foundations that underpin modern navigation systems. Thrun et al. formalized the problem of robot perception through probabilistic robotics, treating state estimation as a recursive Bayesian process [1]. Estimation is framed as computing a belief distribution over the robot state, updated by motion and measurement models. This framework represented a paradigm shift from deterministic approaches, allowing robots to explicitly represent and manage uncertainty in sensor measurements and control actions rather than hiding uncertainty behind heuristic filters.

Siegwart et al. classified mobile robot locomotion and kinematics, providing the mechanical basis for differential drive systems commonly used in indoor environments [2]. Non-holonomic constraints prevent instantaneous sideways motion and thus shape feasible motion primitives. In narrow corridors and turning spaces—a central concern for this thesis—these constraints directly limit feasible trajectories and influence the design of local planners and costmaps.

Early robot architectures such as Brooks’ subsumption architecture emphasised reactive behaviours, where simple behaviours (e.g., obstacle avoidance) inhibited or subsumed higher-level ones (e.g., goal-directed navigation) [3]. While subsumption architectures provided robustness to unexpected events through layering, they offered limited support for global reasoning about maps or longer-term objectives. As sensors and computation improved, hybrid and deliberative architectures became dominant, with explicit mapping, localisation, and planning components.

Murphy and others highlighted that in practical deployments, failures rarely stem from theoretical shortcomings of algorithms; instead, they typically result from sensor degradation, calibration drift, poor environmental modeling, or unanticipated corner cases [4]. In other words, the gap between theoretical robotics and real deployments is often dominated by engineering details rather than algorithmic novelty. This observation underlies the approach of this thesis: to take well-understood probabilistic tools (e.g., Monte Carlo localisation, grid-based planners) and invest effort in configuration, integration, and empirical validation, instead of proposing new but untested algorithms.

2.2.1 Industrial Deployment Considerations

Several authors have emphasized the difference between laboratory demonstrations and industrial deployment. Robustness to changing temperatures, floor reflectivity, mechanical wear, and imperfect maintenance regimes is rarely captured in academic benchmarks. In industrial contexts, robots must tolerate sensor degradation, occasional wheel slip, or imperfectly updated maps, and they must do so without frequent human intervention. As such, robustness, explainability, and ease of maintenance become equally important as raw localization or planning accuracy.

For pharmaceutical-like environments in particular, additional constraints arise: strict hygiene regimes, controlled access, regulated maintenance procedures, and aversion to complex external sensor rigs that are difficult to clean or shield. Approaches that depend on delicate multi-sensor calibrations or complex external rigs are often not acceptable, regardless of their nominal performance. These considerations strongly shape the choice of a single 2D LiDAR and wheel odometry as the primary navigation sensing configuration in this work.

2.3 Minimalist Sensor Architectures

In contrast to the prevailing trend of equipping robots with expensive, heavy multi-sensor suites (e.g., 3D LiDAR, stereo cameras, depth sensors, radar, and ultrasonic arrays), a growing body of literature advocates for minimalist sensor architectures designed around robustness and simplicity. The minimalist paradigm argues that robust navigation can be achieved with limited sensing if the underlying algorithms and configurations are carefully engineered, thereby reducing power consumption, computational overhead, hardware costs, and failure modes [5].

McGuire et al. demonstrated that micro-UAVs could navigate complex obstacle fields using only sparse optical flow and range data, challenging the assumption that dense 3D reconstruction is strictly necessary for obstacle avoidance [6]. Similarly, research in “blind” or degraded-visual navigation has shown that proprioceptive sensing (encoders and IMU) combined with sparse probing can resolve localisation ambiguities even in perception-degraded environments such as smoke-filled rooms or glass-walled corridors [7]. These results collectively suggest that careful exploitation of environment structure and task constraints can offset the lack of rich sensory modalities.

For industrial sterilization robots, where cost and sanitisation constraints are paramount, minimalist 2D LiDAR configurations reduce the mechanical complexity

and calibration overhead associated with multi-modal systems [8]. External cameras require protective housings, lenses must be kept clean, and cabling routes introduce potential contamination traps. In contrast, a single LiDAR installed within or behind a sealed cover can be integrated neatly into the robot chassis.

Jeong et al. demonstrated a linear four-point LiDAR SLAM approach tailored to Manhattan-world environments, showing that even extremely sparse geometric data can yield stable odometry in structured indoor settings provided the environment contains sufficient orthogonality and straight lines [68]. This is directly relevant for corridor-based pharmaceutical layouts, which typically approximate Manhattan geometry in plan-view. Such findings support the thesis hypothesis that 3D LiDAR and dense multi-sensor fusion, while powerful, may be overkill for the task of navigating long, regular corridors at modest speeds.

2.3.1 2D vs. 3D LiDAR in Structured Indoor Spaces

While 3D LiDAR sensors provide volumetric information and are now widely used in autonomous driving and large-scale mapping, their advantages are less pronounced in strongly planar indoor environments. For typical corridor geometries, most relevant obstacle information lies near the floor plane: walls, doors, cabinets, and human legs. A 2D LiDAR scanning parallel to the floor captures sufficient information for collision avoidance and localisation in such settings, assuming the environment does not exhibit significant overhanging obstacles or multi-level structure.

3D LiDAR introduces higher data rates, more complex processing pipelines, and increased cost and power consumption. Moreover, integrating 3D data into occupancy maps suitable for classical 2D planners requires additional down-projection and filtering. For a use case where vertical structure is limited and the primary safety concern is lateral clearance from walls and equipment, the marginal benefit of 3D LiDAR does not obviously justify the additional system complexity. This trade-off underpins the design choice in this thesis to remain strictly within a 2D LiDAR paradigm for navigation.

2.4 LiDAR and IMU Sensing Principles

Light Detection and Ranging (LiDAR) remains a primary enabling technology for indoor navigation due to its active illumination, high angular resolution, and geometric interpretability. A 2D LiDAR produces a planar scan of range measurements, which can be readily transformed into an occupancy grid or costmap.

However, the reliance on planar slices introduces limitations. In long, straight corridors with few distinctive features along the direction of travel, consecutive scans can appear nearly identical regardless of how far the robot has moved longitudinally. This leads to the “corridor effect,” where longitudinal localization becomes ambiguous [9]. Scan-matching algorithms may align scans by sliding them along the corridor axis with minimal cost difference, leading to drift. For slow-moving robots with frequent loop closures, this can be acceptable; for long corridors without distinctive features, the drift can accumulate significantly.

IMUs provide high-frequency measurements of angular velocity and linear acceleration. Although low-cost MEMS IMUs suffer from bias instability, scale factor errors, and noise that accumulates into drift [10], they provide reliable short-time priors and can be used to deskew LiDAR scans acquired while the sensor is moving [11]. Without IMU-based motion correction, rotating LiDARs can produce warped scans in dynamic motion, especially at higher speeds or during combined rotation and translation.

2.4.1 Motivation for Excluding IMU in This Thesis

Many state-of-the-art LiDAR SLAM systems assume tight integration with IMU data to achieve high accuracy and robustness [14, 15]. Nevertheless, integrating an IMU adds more calibration steps (extrinsic and temporal), more potential failure modes, and more complexity in the estimation stack. For a slow-moving sterilization robot operating at conservative speeds, some of the benefits of IMU-based deskewing are less critical, while the additional system complexity remains.

The approach in this thesis therefore deliberately excludes IMU fusion from the navigation loop and relies purely on LiDAR and wheel odometry. This enforces a stricter constraint: the system must be configured such that mapping, localisation, costmaps, and planning remain robust in the absence of high-frequency inertial priors. As a result, the evaluation directly reveals whether a LiDAR-only configuration, with disciplined parameter tuning, can maintain acceptable performance under the corridor conditions of interest.

2.5 Multi-Sensor Fusion Strategies

Where multiple sensors are used, their integration profoundly shapes system behaviour. The literature distinguishes between loosely-coupled and tightly-coupled fusion architectures, with hybrid variants attempting to combine their advantages.

Table 2.2: Comparison of Sensor Fusion Architectures for Mobile Robotics

Feature	Loosely-Coupled Fusion	Tightly-Coupled Fusion
Data Input	Pre-processed poses from each sensor	
(e.g., LiDAR odom + IMU odom)	Raw sensor measurements (e.g., points + accelerations)	
Computational Cost	Low (typically EKF-based)	High (non-linear factor graph optimization)
Robustness	Sensitive to failure of any input module	More resilient; resolves conflicts via joint optimization
Typical Algorithms	EKF, UKF, loosely-coupled filters	Factor Graphs (GTSAM, Ceres), iSAM

2.5.1 Loosely-Coupled Fusion

Loosely-coupled systems treat sensor-specific estimators as black boxes: each sensor provides its own pose estimate, which is then fused using a Kalman filter or related estimator [12]. This modularity simplifies development and debugging and facilitates incremental integration. However, if one sensor’s estimator fails silently (e.g. LiDAR odometry in a featureless corridor), the fusion filter may continue to trust it, degrading the global estimate. Loosely-coupled fusion is also less able to exploit partial information from a degraded sensor, since it operates only at the pose level.

2.5.2 Tightly-Coupled Fusion

Tightly-coupled systems define a joint state vector that includes robot pose, IMU biases, and sometimes landmark parameters, then construct a factor graph or optimization problem over raw measurements [14, 15]. This allows partial information from each sensor to influence the joint state, often leading to superior performance in challenging environments. Frameworks like LIO-SAM [15] and subsequent variants exploit this paradigm for high-performance LiDAR–IMU odometry. However, the computational and implementation cost is considerable, and the number of calibration and configuration parameters increases.

2.5.3 Hybrid Approaches

Hybrid architectures combine loosely- and tightly-coupled ideas, for example by using a loosely-coupled estimator as an initial guess for a sliding-window tightly-coupled optimizer, or by adapting the level of coupling depending on sensor quality [16, 66]. These approaches attempt to balance robustness and computational efficiency, especially on embedded platforms.

In the context of this thesis, such complexity is intentionally avoided. The navigation system is deliberately constrained to a single primary exteroceptive sensor, removing most fusion design questions and forcing the navigation performance to emerge from parameter engineering and structural choices within a simple ROS stack.

2.6 SLAM Systems: From Filters to Graphs

Simultaneous Localization and Mapping (SLAM) is central to autonomous navigation. The evolution from early filter-based solutions to modern graph-based approaches reflects a broader trend: from recursive online estimation toward batch-like optimization with the ability to retrospectively correct trajectories once new constraints (e.g. loop closures) are discovered [60].

Figure 2.1: Taxonomy of SLAM algorithms illustrating the evolution from filtering methods (EKF, particle filters) to modern graph-based optimization techniques.

2.6.1 Filter-Based SLAM

EKF-SLAM [17] maintains a joint Gaussian distribution over robot pose and landmark positions. It is mathematically elegant but scales poorly with the number of landmarks and can suffer from linearization errors that accumulate downstream [18]. FastSLAM [19] addresses the computational scaling by factorizing the problem, but introduces particle depletion issues, especially in long, repetitive environments where resampling tends to discard critical hypotheses [20]. For large indoor facilities or long-term operation, both approaches are limited.

2.6.2 Graph-Based SLAM and Gmapping

Graph-SLAM formulations represent each robot pose as a node and each relative constraint (from odometry or scan matching) as an edge [21]. By minimizing the

residuals on all edges using non-linear least-squares, graph-based SLAM can relinearize and adjust past poses when new loop closures are discovered. Libraries such as g2o and GTSAM provide efficient solvers that exploit problem sparsity [22, 23].

Gmapping, which is widely used in ROS, is an instance of Rao-Blackwellized particle filtering where each particle carries a map and pose hypothesis while the map itself is updated with occupancy-grid techniques. Although not a pure pose-graph SLAM, Gmapping effectively blends particle filtering with grid mapping and has proven robust for 2D indoor maps when loop closure opportunities exist and sensor noise is moderate. For structured, planar environments, its performance is often sufficient, especially when mapping is conducted at moderate speeds and with conservative motion.

Given the modest size of the mapped environment in this thesis and the constraint to 2D LiDAR data, a full-blown 3D graph-SLAM backend is unnecessary. Gmapping offers a practical compromise: well-supported in ROS, relatively easy to configure, and sufficiently robust for the target corridor layouts.

2.7 Point Cloud Processing and Registration

Point cloud registration underlies LiDAR odometry, scan matching, and map building.

2.7.1 Classical ICP and Variants

The Iterative Closest Point (ICP) algorithm remains the canonical method for rigid alignment of point clouds. Point-to-point ICP minimizes squared distances between corresponding points [27], whereas point-to-plane ICP minimizes distances to tangent planes, leveraging local surface geometry [28]. While simple and conceptually transparent, ICP is sensitive to initial alignment, can converge to local minima, and struggles in degenerate geometries where multiple relative transformations produce similar error.

Several works have sought to improve robustness by incorporating robust error norms, outlier rejection strategies, or constraints on search regions [61]. Despite these advances, in long, feature-poor corridors, ICP-based registration still struggles to constrain motion along the corridor axis, making it less suitable as a standalone odometry solution in such environments.

2.7.2 NDT and Probabilistic Approaches

The Normal Distributions Transform (NDT) [29] discretizes the environment into cells, modelling each cell’s point distribution as a Gaussian. Registration is formulated as aligning the moving cloud so that its points land in high-likelihood regions of the fixed Gaussians. NDT provides smoother cost surfaces, often enabling faster convergence and more reliable optimization than ICP in cluttered environments. It also naturally provides a measure of registration confidence.

More recent probabilistic frameworks, such as FAST-LIO2’s direct raw-point registration via IESKF [26], bypass explicit feature extraction and operate directly on dense point sets, leveraging continuous-time models and efficient linearization. However, such methods generally assume tight IMU integration and are more complex than what is needed for the planar navigation scenarios considered here.

2.7.3 Degeneracy and Corridor Geometry

Long, straight corridors induce degeneracy in longitudinal motion estimation. The eigenvalues of the information matrix reveal directions along which geometry provides little constraint [30]. Techniques like LP-ICP [31] adaptively adjust the contribution of measurements in ill-conditioned directions, relying more heavily on motion priors (e.g. wheel odometry) when geometric constraints are weak. For the system in this thesis, this degeneracy is handled indirectly by restricting speed, enforcing conservative planning, and leveraging AMCL’s ability to use global map information rather than relying solely on local scan matching.

2.8 Loop Closure Detection

Loop closure is essential to correct drift accumulated during odometry and scan matching. In large-scale or long-term operation, a SLAM system without loop closure eventually diverges from reality.

In 2D LiDAR-based indoor mapping, loop closure is often implemented implicitly through scan-matching against a global map: when the robot revisits an already-mapped region, the scan matching aligns new scans with the existing map, effectively closing the loop. Explicit loop-closure detection via global descriptors (e.g. Scan Context, M2DP) [33, 34, 62] has become popular for large outdoor or multi-floor mapping, but for modest single-floor indoor layouts, simpler approaches can suffice.

The critical insight for the navigation context of this thesis is that once a reliable

static map is built offline (or during a dedicated mapping run), online navigation relies more on Monte Carlo localisation than on continuous SLAM. As such, loop closure becomes primarily a mapping problem rather than a day-to-day navigation concern.

2.9 Sensor Calibration

Robust navigation presupposes well-calibrated sensors. Calibration errors propagate into all downstream components, often manifesting as the “double walls”, skewed maps, or misaligned coordinate frames familiar to practitioners.

2.9.1 Extrinsic Calibration

Extrinsic calibration of LiDAR to robot base frame is typically performed using manufacturer specifications, verified through controlled experiments. When multiple sensors (e.g. camera + LiDAR) are present, calibration targets and point-cloud alignment techniques are used [38, 39]. Targetless and online calibration methods exist [40], but they add algorithmic complexity and are unnecessary for a LiDAR-only navigation system with fixed mounting.

2.9.2 Intrinsic Calibration

Intrinsic calibration for LiDARs usually focuses on correcting systematic range bias, angular misalignment of laser channels, or timing offsets [41]. For 2D LiDARs in indoor settings, these errors are relatively small and often subsumed into the mapping process: Gmapping and related algorithms can compensate minor distortions as long as they are stable over time. Nevertheless, understanding their presence is useful when interpreting residual distortions in maps.

2.10 Path Planning and Navigation

Once a reliable map and localisation pipeline are established, the robot must plan and execute paths through the environment.

2.10.1 Global Planning in Occupancy Grids

Grid-based global planners treat the environment as a 2D occupancy grid and compute shortest paths using graph search algorithms such as Dijkstra or A* [43]. In ROS, the `global_planner` plugin can be configured to use either algorithm. For

maps of moderate size, the computational burden is negligible on modern embedded processors.

In structured corridor environments, global paths typically consist of straight segments along corridor centres with turns at junctions. The planner’s main role is to ensure that the chosen route respects static obstacles and inflated safety margins while providing a smooth, collision-free “backbone” trajectory for the local planner to follow.

2.10.2 Local Planning and the Dynamic Window Approach

The Dynamic Window Approach (DWA) [45, 46] is widely used as a local planner for differential-drive robots. It samples feasible velocity commands within a short horizon, simulates their effect given the robot’s kinematics, and scores the resulting trajectories based on progress to goal, distance to the global path, obstacle clearance, and forward speed.

DWA’s success depends critically on parameter tuning:

- *Simulation time* must be long enough to foresee collisions and initiate braking early, but not so long that the planner becomes overly conservative.
- *Velocity limits and accelerations* must reflect what the hardware can safely realize.
- *Cost weights* must be chosen to prioritize path adherence and clearance over speed, particularly in narrow corridors.

In poorly tuned configurations, DWA can cause oscillatory behaviour, corner-cutting, or wall-hugging. In the context of this thesis, narrow corridors, a heavy future payload, and the absence of an IMU all motivate conservative velocities, relatively long simulation times, and strong penalties on deviation from the global path.

2.10.3 Costmaps, Inflation, and Dynamic Obstacles

ROS costmaps provide a unified representation incorporating static map information, sensor-based obstacles, and inflation layers. The inflation layer transforms hard occupancy into graded costs, encouraging the planner to maintain a buffer from walls [47]. If the inflation radius is too large relative to corridor width, corridors appear blocked; if too small, the robot risks grazing walls.

Dynamic obstacles-mostly humans and carts in the intended environments-require fast updating and effective clearing. Ray-tracing from LiDAR scans removes old obstacle cells when free space is observed, preventing long-lived ghost obstacles [48]. In practice, the combination of a 5 Hz local costmap and a conservative velocity limit provides sufficient temporal resolution for the robot to detect, decelerate, and re-route around moving obstacles.

2.11 Visual and Learning-Based Perception

The second pillar of this thesis concerns visual perception for industrial inspection, specifically multi-class missing-component detection on automotive fuel-tank assemblies.

2.11.1 Limitations of Classical Computer Vision

Traditional inspection systems rely on hand-crafted features, edge detection, template matching, or simple thresholding. While effective in tightly controlled conditions, these methods are brittle under variations in lighting, component pose, slight misalignments, and background clutter. In realistic assembly-line settings, reflective metallic surfaces and shadows generate artefacts that are difficult to handle with rule-based image processing alone.

2.11.2 Visual-Inertial Odometry and Visual SLAM (Contextual)

Visual-Inertial Odometry (VIO) and visual SLAM systems such as VINS-Mono and ORB-SLAM3 [49, 50, 63] demonstrate that cameras and IMUs can provide highly accurate motion tracking under favourable conditions. However, their sensitivity to illumination, motion blur, and textureless surfaces makes them less appropriate as primary navigation sensors in sterile environments dominated by white or metallic walls. Instead, this thesis restricts cameras to inspection tasks, where the robot or camera mount is fixed or quasi-static, and lighting can be controlled to some extent.

2.11.3 Deep Learning for Object Detection: Two-Stage vs. One-Stage

Deep learning has transformed object detection. Two-stage detectors (R-CNN, Fast R-CNN, Faster R-CNN) generate region proposals and then classify and refine them [51, 52]. These methods achieve high accuracy but at the cost of computational complexity and latency. On the other hand, one-stage detectors such as YOLO

treat detection as a direct regression from image pixels to bounding boxes and class probabilities [53, 54]. This architectural difference enables much higher inference speed, making YOLO-type models more suitable for time-constrained industrial inspection scenarios.

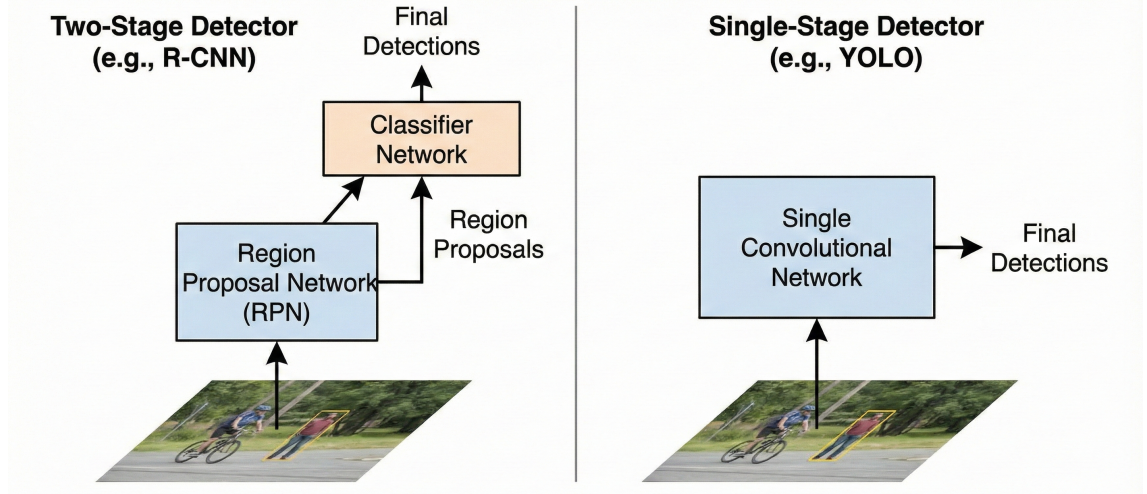


Figure 2.2: Structural difference between single-stage detectors (YOLO) and two-stage detectors (R-CNN). YOLO processes the entire image in a single forward pass, enabling real-time performance suited to assembly lines.

YOLOv8 introduces anchor-free detection and decoupled classification/regression heads, which improves both localization accuracy and training stability [54]. Empirical comparisons in industrial contexts confirm that YOLOv8 offers a favourable trade-off between accuracy and speed, typically achieving near state-of-the-art mAP with significantly lower latency than two-stage baselines [55, 56].

2.11.4 Industrial Datasets and Missing-Component Detection

Despite the maturity of object detection, there is a notable scarcity of public datasets tailored to missing-component detection in real assemblies. Many industrial papers either rely on proprietary datasets that are not shared or use synthetic or simplified imagery. As a consequence, models trained on generic datasets (e.g. COCO, Pascal VOC) often fail to generalise to dense mechanical assemblies with small, similar parts.

Works targeting bolt and screw detection, cable presence, or fastener verification highlight characteristic difficulties: occlusions, specular reflections, slight geometric variations in parts from different suppliers, and confusion between different types

of small hardware elements. These studies typically find that performance strongly depends on collecting representative data under the exact conditions encountered on the line, validating the decision in this thesis to construct a dedicated Missing Parts dataset aligned with a real fuel-tank assembly setup.

2.12 Research Trends and Gaps

The motivation for this thesis was stated in Chapter 1, which identified the principal shortcomings of sensor-rich pipelines and traditional inspection methods from a practical deployment perspective. This section complements that discussion by surveying the current direction of the academic literature and identifying the specific gaps that the present work addresses. Whereas Chapter 1 argued from engineering and operational principles, the analysis here is grounded in the state of the published literature.

Recent research trends in navigation include lifelong SLAM, semantic mapping, multi-agent cooperation, and integration of learning-based components into classical pipelines [57, 58, 70]. In deep learning for perception, trends include transformer-based detectors, self-supervised pretraining, and uncertainty estimation for safety-critical applications.

Despite this progress, several gaps remain that are directly relevant to the thesis:

- **Minimalist, deployment-ready navigation stacks:** Many advanced LiDAR-IMU systems achieve impressive benchmark performance but require complex calibration, configuration, and compute resources [64, 69]. There is comparatively less work rigorously validating deliberately minimal configurations (e.g. 2D LiDAR + odometry, no IMU) tailored to structured industrial environments such as corridors.
- **Navigation parameter engineering for narrow corridors:** While DWA, costmaps, and AMCL are widely used, systematic, deployment-oriented studies that tie parameter configurations to performance in very narrow, structured environments are less common. Many papers focus on algorithmic improvements rather than detailed engineering trade-offs.
- **Industrial-grade missing-component datasets:** Most available datasets do not capture the exact visual complexity and hardware diversity of real automotive assemblies. There is a lack of publicly documented pipelines that start

from data collection in factories and end in real-time deployment evaluation for multi-class missing-component detection.

- **Bridging research and practice:** Both in navigation and visual inspection, there is a disconnect between academic prototypes and systems that technicians can maintain in production environments. Deployment-centric studies that emphasize simplicity, robustness, and maintainability are still underrepresented.

This thesis positions itself precisely at this intersection. On the navigation side, it provides a detailed, experimentally validated account of what can be achieved with a 2D LiDAR-only configuration in corridor-like environments using disciplined parameter tuning. On the inspection side, it contributes a realistic multi-class dataset and a YOLOv8-based pipeline for missing-component detection, evaluated not only in terms of accuracy but also real-time performance and robustness.

Together, these contributions address concrete gaps in deployment-oriented literature, focusing on systems that can realistically be adopted in industrial practice without relying on fragile, overly complex sensing or algorithmic infrastructure.

Chapter 3

Mechanical Design

3.1 Concept and Functional Design

This project concerns the design and realisation of a self-propelled mobile robotic platform intended for operation in pharmaceutical and controlled industrial environments. The primary application motivating the mechanical design is the automated transport and positioning of the *Bioreset Smart* hydrogen peroxide generator for bio-decontamination procedures.

Beyond payload transport, the mechanical design is conceived as an enabling layer for reliable autonomous navigation. In structured indoor environments, the achievable performance of localisation and planning algorithms is tightly coupled to physical factors such as platform geometry, mass distribution, wheel configuration, and sensor placement. Consequently, the mechanical architecture is developed with explicit consideration of navigation stability, repeatability, and safety, rather than treating the chassis as a passive support structure.

Operational flexibility is incorporated through multiple connectivity modes. During operation, the user can access the management software of both the robotic platform and the sanitizer via a direct wired connection, a dedicated wireless network, or the facility’s local area network (LAN), when available. This flexibility allows the operator to either remain connected throughout the entire bio-decontamination cycle or to connect only during critical phases, such as initiation and status verification, without affecting autonomous execution.

The platform adopts a wheeled configuration with differential steering, supplemented by passive pivoting wheels for load stabilisation. The mechanical layout is organised into two vertically separated functional sections:

- **Lower Section:** Dedicated to housing high-mass components, including the battery system and the hydrogen peroxide tank. Locating these elements close to the ground plane lowers the centre of gravity, improving static stability and reducing dynamic load transfer during acceleration and turning.
- **Upper Section:** Designed to support the sanitizer unit at an elevated position to ensure effective dispersion of the decontaminating agent. This section also accommodates the 2D LiDAR sensor used for autonomous navigation,

positioned to maximise environmental observability.

This separation of functions simplifies mechanical integration while ensuring that sensing, actuation, and payload requirements are satisfied without mutual interference.

3.2 System Architecture and Specifications

The mobile robotic platform is designed to satisfy technical and regulatory constraints typical of pharmaceutical and healthcare environments. Its operation is structured into layered subsystems, spanning high-level mission management down to physical actuation. These layers are summarised in Table 3.1.

Table 3.1: Architectural Layers of the Autonomous System

Layer	Component	Function
High-Level	Mission Supervisor (Python)	Manages the finite state machine (FSM), task sequencing, sterilisation logic, and fault handling.
Planning	<code>move_base</code> (ROS)	Computes global paths and generates real-time velocity commands using costmaps and standard planners.
Localization	AMCL (Particle Filter)	Estimates robot pose by aligning LiDAR scans with a pre-built occupancy grid map.
Hardware	<code>ubiquity_motor</code> Node	Acts as the hardware abstraction layer (HAL) for motor control, odometry computation, and kinematics.
Physical	MCB & Chassis	Implements embedded PID control and mechanical actuation of the drive system and safety components.

The principal performance characteristics of the platform are summarised in Table 3.2. These specifications reflect a design emphasis on controlled motion, oper-

ational reliability, and compatibility with autonomous navigation rather than high-speed traversal.

Table 3.2: Technical Specifications of the Mobile Robot

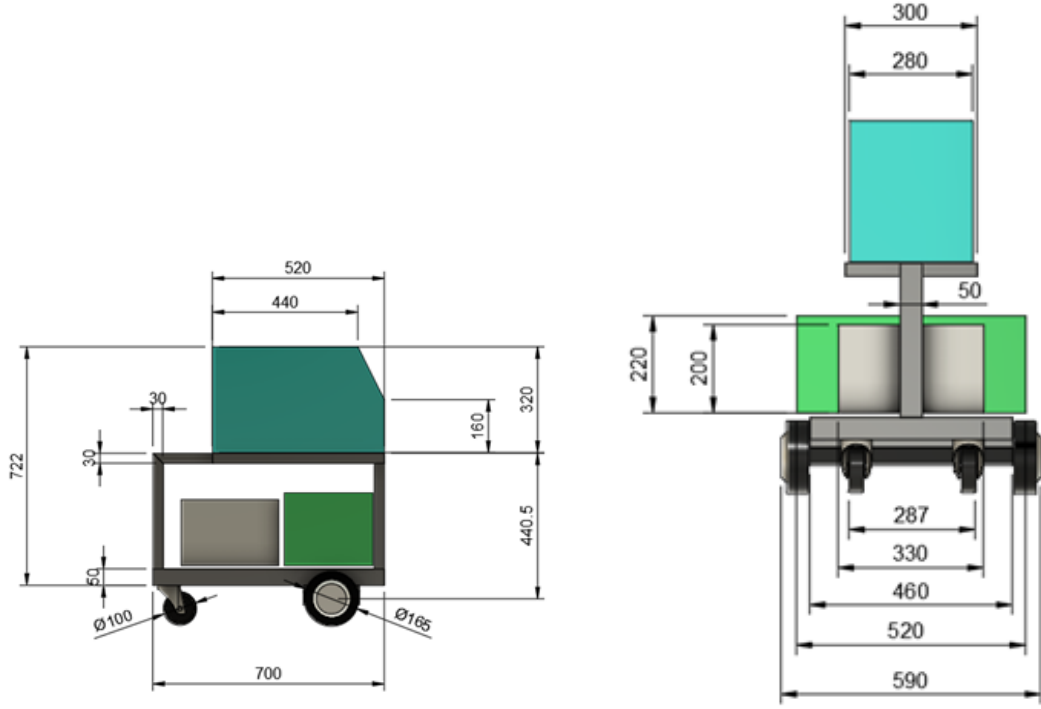
Feature	Specification
Autonomy	Fully autonomous navigation with optional assisted remote control via PC.
Speed Control	Maximum translational and rotational speeds configurable via software.
Payload	Mechanically compatible with the weight and dimensions of the <i>Bioreset Smart</i> generator.
Energy Efficiency	Maximum power consumption of approximately 1 kW, with an operational endurance of up to 5 hours (approximately 5 kWh total energy draw).
Navigation	Autonomous obstacle avoidance, return-to-base functionality, and editable mission routes.
Connectivity	Wireless connectivity for local PC access or facility network integration.
Data Logging	Logging of sterilisation parameters, including release quantities and spatial locations.

3.3 Mechanical Layout and Dimensions

The overall dimensions of the platform are driven by the spatial requirements of the primary internal components. Preliminary integration studies indicate that the battery requires a volume of approximately $520 \times 270 \times 220$ mm, while the sanitizer unit occupies approximately $520 \times 280 \times 320$ mm.

These constraints define the minimum footprint and vertical stacking of the chassis. Figure 3.1 illustrates the resulting mechanical envelope. The side view in Figure 3.1a highlights vertical clearance, component stacking, and sensor elevation,

while the front view in Figure 3.1b illustrates platform width and track spacing, which directly influence navigability in narrow corridors.



(a) Side view showing vertical clearance and longitudinal dimensions.

(b) Front view showing width, wheel track, and lateral constraints.

Figure 3.1: Overall mechanical dimensions of the designed device.

The selected footprint balances manoeuvrability with stability, ensuring sufficient clearance for conservative costmap inflation while avoiding excessive narrowing that would compromise structural integrity or payload accommodation.

3.4 Actuation System

Locomotion is achieved through two hub-mounted BLDC (Brushless DC) motors directly integrated into the drive wheels. This configuration eliminates external transmissions, reducing mechanical complexity, backlash, and maintenance requirements.

The selected motor (Figure 3.2) is characterised by:

- **Dimensions:** Wheel diameter of 164 mm and tread width of 46 mm with a toroidal profile to enhance floor contact.

- **Electrical Characteristics:** Rated power of 350 W with operating voltages of 24/36 V and 30 pole pairs.
- **Mechanical Performance:** Nominal torque of 7.5 Nm and load-bearing capability up to 750 N per wheel.
- **Sensing and Feedback:** Integrated Hall-effect sensors (three per motor), yielding 90 pulses per revolution for odometry and closed-loop control.



Figure 3.2: BLDC hub motor selected for platform propulsion.

The motor selection prioritises torque availability and smooth low-speed operation, aligning with the navigation requirements of corridor-based environments where precise velocity control is more critical than maximum speed.

3.5 Control and Electronic Drive System

Motor control is implemented through a dedicated electronic board integrating an STM32 microcontroller and dual motor drivers. This board interfaces directly with the Hall-effect sensors and executes low-level closed-loop control, including speed regulation and differential-drive kinematics (Figure 3.3).

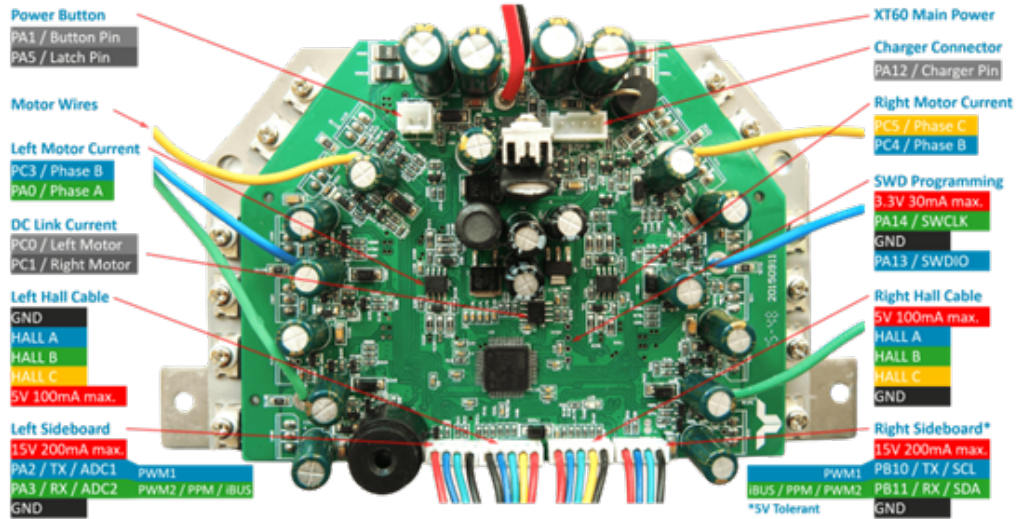


Figure 3.3: Motor control board with STM32 microcontroller and integrated drivers.

At a higher level, an ESP32 microcontroller provides wireless connectivity and acts as a communication bridge between the robotic platform and external supervisory systems. This separation of low-level motor control from high-level communication enhances robustness and fault isolation.

3.6 Power Supply and Battery Selection

The power system is designed around the operational requirements of the *Bioreset Smart* generator (Figure 3.4). Table 3.3 summarises the key technical specifications relevant to power sizing.

Table 3.3: Technical Specifications of the Bioreset Smart Evaporizer

Parameter	Value / Specification
Max Treatable Volume	Up to 500 m ³
Nominal Fan Flow Rate	Up to 200 m ³ /h, adjustable
Peristaltic Pump	Integrated, 3–12 g/min (step 0.5)
Power Supply	230 V / 50 Hz or 110 V / 60 Hz
Max Power Consumption	1 kW
Dimensions (W × D × H)	280 × 530 × 376 mm
Weight	19 kg
Communication Protocol	ModBus TCP/IP
Remote Management	VNC-based interface
Construction Materials	Stainless steel AISI 316 and ABS

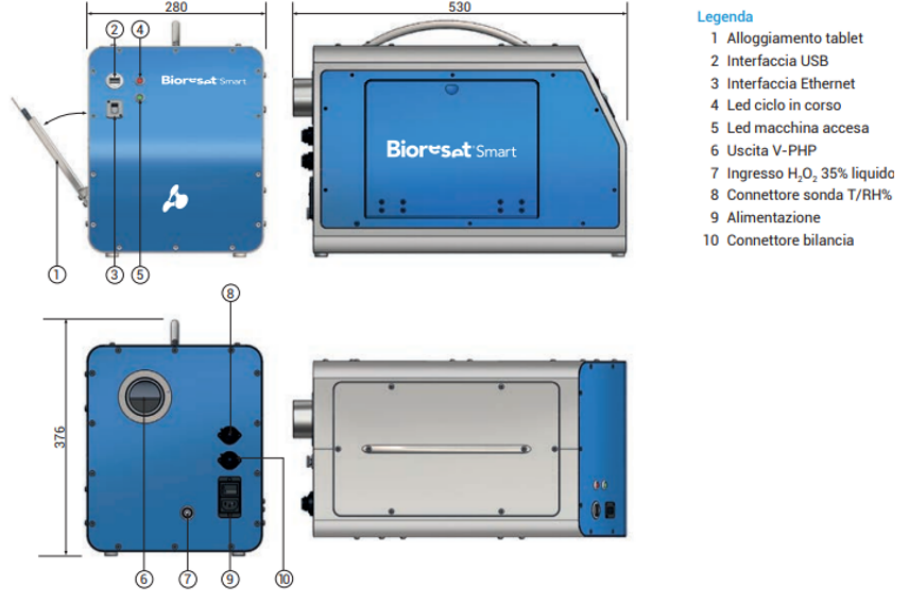


Figure 3.4: Bioreset Smart hydrogen peroxide generator.

To sustain an average operating duration of five hours at peak load, the worst-case energy requirement is:

$$E = 1 \text{ kW} \times 5 \text{ h} = 5000 \text{ Wh} \quad (3.1)$$

which corresponds to a nominal capacity of:

$$C = \frac{5000 \text{ Wh}}{24 \text{ V}} \approx 208.33 \text{ Ah.} \quad (3.2)$$

Accordingly, a 24 V, 200 Ah Lithium Iron Phosphate (LiFePO₄) battery was selected, providing approximately 4.8 kWh of nominal stored energy (Figure 3.5). The 200 Ah battery provides 4.8 kWh of usable energy, compared to the theoretical 5.0 kWh requirement at continuous maximum draw. This 200 Wh margin is addressed by the operational duty cycle: the Bioreset Smart generator does not operate continuously at peak power during transit phases, giving an effective average draw of approximately 0.96 kW (= 4.8 kWh / 5 h), which is within the battery capacity. This chemistry was chosen for its thermal stability, safety characteristics, and long cycle life.



2000 CICLI DI CARICA/SCARICA GARANTITI

24v 200Ah
 Dimensioni: 522x237x217 mm
 Peso [kg]: 36
 Tensione nominale [V]: 25.6
 Capacità [Ah]: 200
 Energia contenuta a piena carica: 4.8kWh
 Corrente massima di ricarica [A]: 100
 Corrente massima di scarica [A]: 200
 Potenza di picco max in uscita [W]: 4800
 Tensione di ricarica [V]: 29.2±0.2
 Tipo di terminale: M8

Figure 3.5: LiFePO₄ battery selected for the platform.

The battery is structurally integrated into the lower frame to maintain a low centre of gravity and to minimise vibration transmission to the sensing and control subsystems.

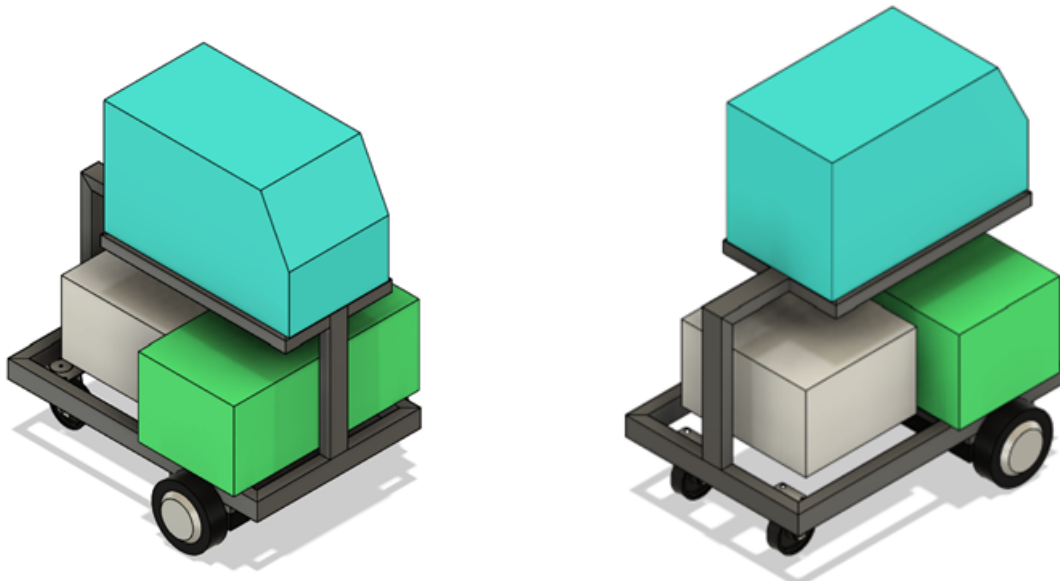
3.7 Structural Design and Prototyping

The structural frame is constructed from welded rectangular steel profiles of $50 \times 30 \times 2$ mm, providing high rigidity and resistance to torsional deformation.

The mechanical architecture comprises two functional levels:

- **Lower Platform:** Formed by a 20 mm poplar plywood panel reinforced with a 5 mm aluminium tread plate. This level supports the battery and fluid tank.
- **Upper Platform:** Rigidly supported by the steel frame and designed to carry the sanitizer unit and the LiDAR sensor, ensuring an unobstructed sensing field.

The CAD model of the platform is shown in Figure 3.6, with isometric views highlighting component placement and load paths.



(a) Rear isometric view showing internal layout.

(b) Front isometric view with payload installed.

Figure 3.6: CAD design of the robotic platform.

A physical prototype was manufactured based on this design (Figure 3.7), validating mechanical integration and structural robustness.



Figure 3.7: Manufactured prototype of the robotic platform.

3.8 Physical Prototype and Structural Realization

Following the initial design phase, the physical prototype of the robot chassis was constructed to validate the structural integrity and mounting feasibility of the platform. Figure 3.8 presents the realized assembly from three orthographic perspectives, showing the integration of the modular superstructure with the mobile base.

3.8.1 Mechanical Framework and Assembly

The superstructure was realized using **20×20 mm T-slot aluminum extrusions**. This material was selected for its high strength-to-weight ratio and the flexibility it offers for mounting auxiliary hardware. The front and side views (Figure 3.8a and 3.8b) highlight the vertical pillars designed to support the sterilization payload. By utilizing high-torque L-brackets and internal T-nuts, the framework is rigidly coupled to the Magni’s steel chassis, ensuring that vibrations during motion are minimized.

3.8.2 Internal Layout and Stability

The front side (Figure 3.8c) reveals the internal geometric configuration of the frame. The design maintains an open central cavity, which is strategically utilized to house the heaviest system components, including the primary 24V battery and the DC-AC inverter. By centralizing this mass between the drive wheels and keeping it at the lowest possible elevation, the robot’s center of gravity (CoG) is optimized. This configuration ensures maximum traction for the drive wheels and provides superior stability during the rapid turns required for navigating tight pharmaceutical environments.

3.9 Experimental Platform and Operational Use

The experimental validation presented in subsequent chapters is conducted using the custom-manufactured platform described above. The rigid steel frame and dual-level architecture remain unchanged across experiments, ensuring consistent mechanical behaviour.

Navigation relies on the LiDAR sensor mounted on the elevated platform, providing a clear and stable field of view for localisation and obstacle detection. The control software supports both fully autonomous operation and supervised interaction via a PC interface.



(a) Front View



(b) Side View



(c) Rear View

Figure 3.8: Physical realization of the robot’s mechanical framework. The assembly shows the differential-drive mobile base (Magni platform, welded rectangular steel profiles) topped by the custom-built payload superstructure constructed from 20×20 mm T-slot aluminum extrusions. Note that the lower structural chassis visible in Figure 3.6 is the welded steel skeleton of the base, while the vertical pillars shown here are the aluminum extrusion uprights that support the sterilization generator. These views represent the completed skeleton prior to the installation of sensors and the sterilization payload.

A key operational feature is automatic recovery following manual intervention. If the operator temporarily adjusts the robot's position or motion, the system is capable of re-localising and autonomously returning to the pre-intervention trajectory, thereby preserving mission continuity.

Chapter 4

System Architecture and Software Framework

4.1 Introduction

The successful operation of an autonomous mobile robot (AMR) in a sensitive pharmaceutical environment requires a software architecture that balances robustness, efficiency, and behavioural predictability. This chapter presents the system architecture and software framework implemented on the Ubiquity Robotics Magni platform used in this work.

The system is designed with **architectural provisions for future scalability**. While the experimental platform used in this thesis operates in a lightweight configuration (chassis and sensors only), the software stack has been parameterized to accommodate the anticipated mass of the Bioreset Smart VHP (Vaporized Hydrogen Peroxide) generator and an upgraded battery pack. The architecture incorporates design margins for increased inertia, tire compression, and higher currents directly into the navigation and control tuning. It should be noted that while these provisions are encoded in the software, full dynamic validation under maximum payload represents a future integration phase.

The architecture is implemented on top of Robot Operating System (ROS) Noetic running on Ubuntu 20.04 LTS. ROS provides the messaging, parameter server, and tooling required to decompose the system into loosely coupled nodes. Perception, localization, planning, control, and mission supervision execute as separate processes. This modularity ensures that the navigation stack remains structurally valid as the payload evolves, provided the core interfaces and timing guarantees are preserved.

The design philosophy prioritises three factors derived from the problem statement:

1. **Inertial Tuning Strategy:** The control logic is tuned for a mass budget higher than the current test configuration. Acceleration profiles are chosen conservatively (0.2 m/s^2) to ensure that the navigation stack remains stable if the payload increases, minimizing the need for architectural changes during future integration.
2. **Power Management Integration:** The software architecture is designed

to manage the energy constraints defined in Chapter 3, incorporating logic to handle the high current demands of the sterilizer payload during mission execution.

3. **Process Logic:** The robot is designed as a proof-of-concept process tool. Navigation, dwell times, and nebulizer operation are integrated in a sequential FSM, ensuring that disinfection only occurs under controlled conditions.

4.1.1 Predictability and Real-Time Constraints

In the context of this work, the term "determinism" refers to **behavioural predictability** rather than hard real-time guarantees at the operating system level. Standard Linux kernels and ROS messaging introduce non-deterministic jitter in message delivery and context switching.

Therefore, the system operates as a **soft real-time architecture**. Worst-case delays are managed through:

- Asynchronous, multi-threaded spinners for handling callbacks.
- Decoupling high-frequency motor control (1 kHz, embedded on microcontroller) from high-latency path planning.
- Conservative velocity limits that allow for safe deceleration even in the event of minor processing delays.

4.1.2 Scope and System Limitations

To bound the scope of the architectural design, the following limitations are explicitly acknowledged:

- **No IMU:** The architecture relies exclusively on wheel odometry and LiDAR. This design choice reduces cost and complexity but assumes a high-friction, flat flooring typical of pharmaceutical cleanrooms. The system is not designed for uneven terrain or significant wheel slip.
- **Static Environment Assumption:** While local avoidance handles dynamic obstacles, the global localization assumes the structural layout (walls) remains static.

- **Safety Certification:** The safety protocols described herein are implemented in software and are intended to demonstrate functional safety concepts (e.g., ISO 3691-4). They do not constitute a certified SIL/PL safety system.

4.2 Software Decomposition and Node Graph

From a software perspective, the system can be viewed as a set of interacting sub-systems:

- **Mission Layer:** A Python-based supervisor node implements the FSM, way-point scheduling, and sterilization logic using the `actionlib` interface to `move_base`.
- **Navigation Layer:** The `move_base` node manages the global and local planners, global and local costmaps, and recovery behaviours.
- **Localization Layer:** AMCL consumes odometry and LiDAR scans, publishing pose estimates in the `map` frame.
- **Perception Layer:** The `rplidar_ros` driver and filtering nodes provide a clean `/scan` stream to both localization and obstacle layers.
- **Hardware Layer:** The `ubiquity_motor` node exposes a standard ROS interface (`/cmd_vel`, `/odom`) over a custom serial protocol to the Motor Controller Board (MCB).
- **Payload Layer:** A future relay-based interface toggles the Bioreset Smart via GPIO under mission-layer control.

This decomposition is reflected in the high-level block diagram in Figure 4.1.

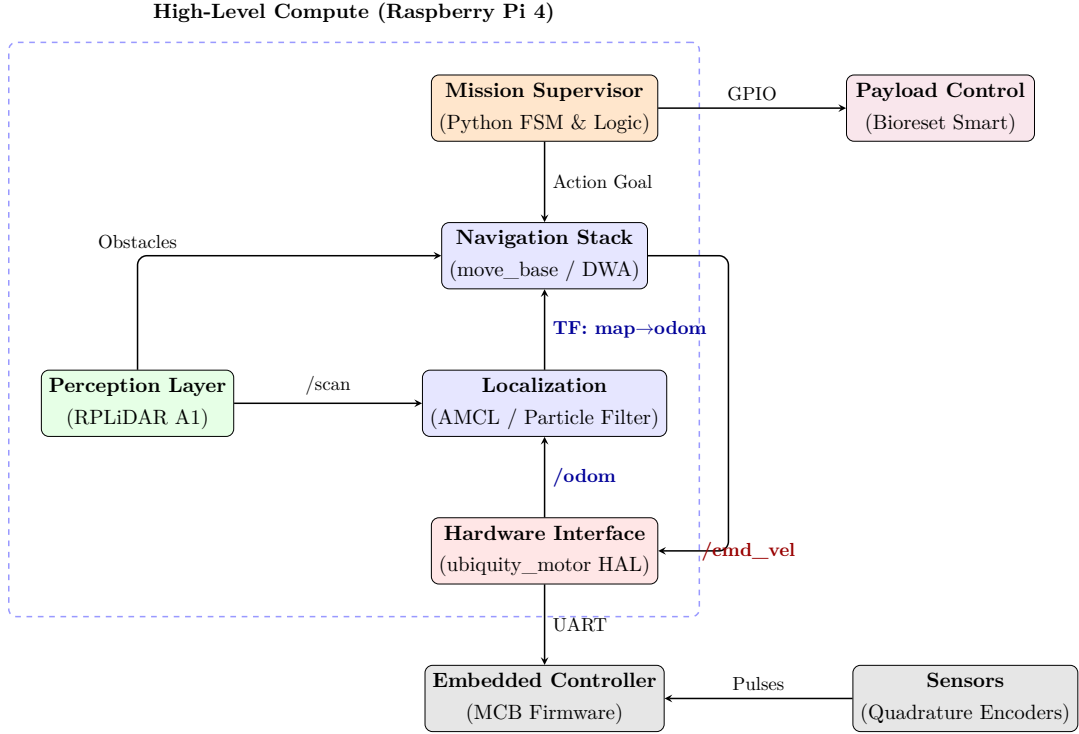


Figure 4.1: System Data Flow. **Feedback (Blue)** flows upward: Hardware $\rightarrow$ Localization $\rightarrow$ Navigation. **Control (Red)** flows downward: Navigation $\rightarrow$ Hardware. This closed-loop architecture ensures the planner always acts on the most recent pose estimate.

4.3 Computational Hardware and Environment

The central compute unit is an on-board Raspberry Pi 4 Model B featuring a quad-core Cortex-A72 (ARM v8) 64-bit SoC at 1.5 GHz and 4 GB of LPDDR4 RAM. This modest computational budget constrains both algorithm choice and parameterization; the architecture must balance localization quality, planner responsiveness, and mission logic without saturating the CPU.

4.3.1 Why Edge Computing?

Many modern robotic systems offload intensive tasks such as SLAM or global planning to cloud servers or remote workstations. In cleanroom-like pharmaceutical facilities, this is a liability rather than an advantage:

- **Network Reliability:** Sterilization often occurs in rooms with thick walls, shielded areas, and hardware that interferes with WiFi. Treating connectivity as a hard dependency would make the system fragile.

- **Latency Management:** Obstacle avoidance and safety behaviours must operate with bounded latency. Round-trip delays to external servers are incompatible with reliable collision avoidance.
- **Data Security:** Maps and sensor data may indirectly reveal facility layouts or operational routines. Keeping all processing on-board simplifies compliance with strict data policies.

For these reasons, all perception, localization, planning and mission layers run directly on the robot, with the external workstation used only for monitoring and data logging.

4.4 System Initialization and Boot Process

Industrial application demands predictable startup behaviour. The system uses Linux `systemd` units to automatically launch the ROS core and the navigation stack on power-up, without user interaction.

4.4.1 The Master Launch Sequence

The master launch file `finalLaunchFile_22Nov2025F.launch` orchestrates the startup sequence. Dependency ordering is chosen to avoid race conditions (nodes waiting on nonexistent topics or transforms):

1. Stage 1: Core Hardware ($T = 0$ s)

The `ubiquity_motor` node starts first, initializing the serial link to the MCB. Failures at this stage (disconnected cable, wrong port, permission issues) are treated as fatal, and the rest of the stack is not launched.

2. Stage 2: Sensor Drivers & Static Transforms ($T + 2$ s)

Once the chassis responds, `rplidar_ros` is launched and waits for the LiDAR to spin up to a stable scan frequency. In parallel, `robot_state_publisher` loads the URDF and publishes the static TF tree (e.g., `base_link` to `laser`).

3. Stage 3: Navigation Stack ($T + 5$ s)

The `map_server` loads the occupancy grid from disk. Then AMCL initializes the particle cloud. Finally, `move_base` is started and waits until all required topics and transforms (`/map`, `/odom`, `/tf`, `/scan`) become available.

4. Stage 4: Mission Logic (T + 10 s)

The Python supervisor `Multipoint_code_working_F.py` is launched last. It verifies connectivity with the `move_base` Action Server and performs lightweight health checks (CPU load, topic presence) before entering the INIT state.

This staged approach ensures that the mission logic never runs against a partially initialized navigation stack.

4.5 Hardware–Software Interface (HAL)

The `ubiquity_motor` node serves as a Hardware Abstraction Layer between the ROS world and the MCB.

4.5.1 Serial Communication Protocol

Communication between the Raspberry Pi and MCB uses a UART interface at 115200 baud. The protocol is packet-based, with framing bytes and checksums to detect corruption in noisy environments. The control loop runs at 50 Hz, which defines the frequency at which new velocity commands and encoder updates are exchanged.

4.5.2 Differential Drive Kinematics

The Magni is a non-holonomic differential-drive robot. Let r be the effective wheel radius and L the distance between the drive wheels. The relationship between wheel angular velocities $(\dot{\phi}_R, \dot{\phi}_L)$ and body-frame velocities (v, ω) is:

$$v = \frac{r}{2}(\dot{\phi}_R + \dot{\phi}_L) \quad (4.1)$$

$$\omega = \frac{r}{L}(\dot{\phi}_R - \dot{\phi}_L) \quad (4.2)$$

or in matrix form:

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ \frac{r}{L} & -\frac{r}{L} \end{bmatrix} \begin{bmatrix} \dot{\phi}_R \\ \dot{\phi}_L \end{bmatrix}. \quad (4.3)$$

The inverse mapping-used by the MCB to convert navigation commands into wheel targets-is:

$$\dot{\phi}_R = \frac{2v + \omega L}{2r}, \quad \dot{\phi}_L = \frac{2v - \omega L}{2r}. \quad (4.4)$$

4.5.3 Empirical Payload Correction (Heuristic)

A heavier payload causes tire compression and reduces the effective rolling radius. If the software assumes a nominal radius r_{nominal} while the real radius is $r_{\text{eff}} < r_{\text{nominal}}$, the robot will under-travel for a given number of encoder ticks.

To mitigate this without complex dynamic modeling, the architecture employs a linear heuristic:

$$r_{\text{eff}} \approx r_{\text{nominal}} (1 - \epsilon_{\text{load}}), \quad (4.5)$$

where ϵ_{load} is an empirical compression factor tuned experimentally. This serves as a calibration parameter during deployment rather than a derived physical constant.

4.6 Low-Level Control Dynamics

The MCB executes a high-frequency PID loop (1 kHz) to regulate motor currents. The HAL layer shapes the reference signals presented to these controllers.

4.6.1 Inertial Damping and Ramp Functions

Abrupt changes in commanded velocity lead to excessive torque demand, wheel slip, and potentially unstable behaviour when carrying liquid disinfectant. To mitigate this, the HAL implements acceleration limiting:

$$v_{\text{cmd}}(t) = \begin{cases} v_{\text{target}} & \text{if } |v_{\text{target}} - v_{\text{prev}}| < a_{\text{max}} \Delta t, \\ v_{\text{prev}} + \text{sgn}(v_{\text{target}} - v_{\text{prev}}) a_{\text{max}} \Delta t & \text{otherwise,} \end{cases} \quad (4.6)$$

with $a_{\text{max}} = 0.2 \text{ m/s}^2$ in this work. This value is deliberately conservative to maintain traction on smooth pharmaceutical flooring and reduce fluid sloshing when the evaporator is installed.

4.7 Odometry and State Estimation

The `ubiquity_motor` node integrates encoder ticks into a pose estimate $x_t = [x, y, \theta]^T$ in the `odom` frame.

4.7.1 Runge-Kutta Integration and IMU Exclusion

To improve accuracy over naive Euler integration, a second-order Runge-Kutta (RK2) scheme is used:

$$\Delta s = v_t \Delta t, \quad \Delta \theta = \omega_t \Delta t, \quad (4.7)$$

$$x_{t+1} = x_t + \Delta s \cos \left(\theta_t + \frac{\Delta \theta}{2} \right), \quad (4.8)$$

$$y_{t+1} = y_t + \Delta s \sin \left(\theta_t + \frac{\Delta \theta}{2} \right), \quad (4.9)$$

$$\theta_{t+1} = \theta_t + \Delta \theta. \quad (4.10)$$

The half-angle term better approximates curved trajectories during a control interval.

The system operates without an IMU, as discussed and justified in Section 2.4.1 of Chapter 2. The flat, high-friction flooring of the pharmaceutical environment makes wheel odometry combined with LiDAR-based correction sufficient for the target application, while avoiding the calibration complexity associated with inertial sensors.

4.8 Coordinate Frames and TF Tree

ROS uses the TF2 library to maintain the spatial relationships between coordinate frames. For this robot, the TF tree connects the world, odometry, base, and sensor frames.

4.8.1 Live TF Tree Validation

To validate the implemented frame hierarchy, the `view_frames` tool was run during operation. The resulting graph (exported as `tf_frames_output.pdf`) is shown in Figure 4.2.

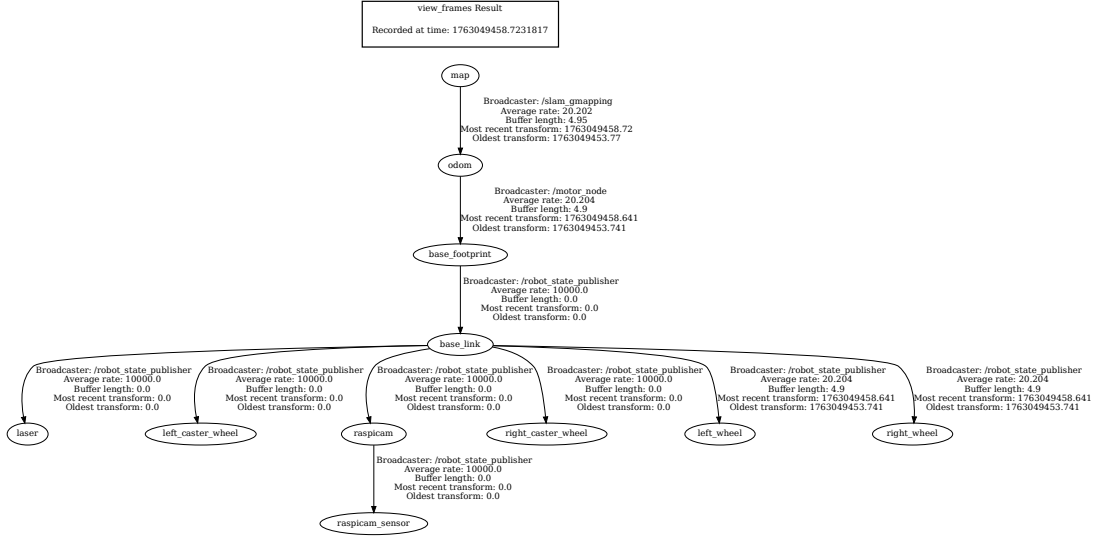


Figure 4.2: Live Transform (TF) tree captured during operation. The graph confirms the chain `map` $\rightarrow$ `odom` $\rightarrow$ `base_footprint` $\rightarrow$ `base_link` $\rightarrow$ `laser`, consistent with the architectural model.

Two practical observations emerge from the live TF data:

- The `map` $\rightarrow$ `odom` transform, published by `amcl` during navigation or by `gmapping` during mapping, updates at approximately 20 Hz. This is sufficient to keep pose corrections aligned with the odometry integration rate.
- The timestamps of the most recent transforms are closely aligned with the current system time, indicating that TF buffering is not lagging and that the planners operate on up-to-date geometry.

4.8.2 Homogeneous Transformation Matrices

The relationship between frames A and B is represented by a homogeneous transformation $T_A^B \in SE(3)$:

$$T_A^B = \begin{bmatrix} R & p \\ 0_{1 \times 3} & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & x \\ r_{21} & r_{22} & r_{23} & y \\ r_{31} & r_{32} & r_{33} & z \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (4.11)$$

where R is the rotation matrix (from the quaternion orientation) and $p = [x, y, z]^T$ the translation vector.

4.8.3 Frame Hierarchy

The implemented hierarchy follows REP-105:

- **map**: World-fixed frame. The `amcl` node publishes $T_{\text{map}}^{\text{odom}}$ to correct for accumulated odometry drift.
- **odom**: Locally smooth odometry frame. The `ubiquity_motor` node publishes $T_{\text{odom}}^{\text{base_footprint}}$ by integrating encoder ticks.
- **base_footprint** / **base_link**: Frames rigidly attached to the robot’s chassis; `base_link` carries the inertia and collision model.
- **laser**: LiDAR mounting frame. The static transform $T_{\text{base_link}}^{\text{laser}}$ encodes the sensor offset, including the 0.20 m mounting height above the floor.

This TF structure allows costmaps and planners to consistently project laser data into map coordinates and back into the local robot frame.

4.9 Perception and Sensor Processing

The primary exteroceptive sensor is an RPLiDAR A1, a 2D scanner producing 360° scans at approximately 10 Hz.

4.9.1 LiDAR Data Pipeline

The perception pipeline consists of:

1. **Driver Layer**: `rplidar_ros` parses the sensor’s UART stream and publishes `sensor_msgs/LaserScan` on `/scan`.
2. **Geometric Filtering**: A `laser_filters` configuration applies a box filter that removes returns from the robot’s own structure (chassis pillars, mounting hardware).
3. **Noise Filtering**: A speckle filter discards small, isolated clusters of points. This reduces false obstacles caused by dust, reflections, or single-sample noise.

The filtered scan is then used both by AMCL for localization and by the obstacle layers of the costmaps.

4.10 Environment Representation (Costmaps)

`move_base` maintains two costmaps:

4.10.1 Global Costmap

The global costmap is static; it is initialized from the map produced during the mapping phase (Section 4.13.1). Obstacles are inflated by the robot’s inscribed radius plus a safety margin corresponding to the anticipated future payload footprint. This “oversizing” ensures that even with a mounted evaporator and cabling, the robot maintains clearance from walls and equipment.

4.10.2 Local Costmap

The local costmap is a rolling window (3 m × 3 m in this implementation) centered on the robot. It updates at 5 Hz from live LiDAR scans and is used exclusively by the local planner. This separation allows global planning to assume a static environment while local planning reacts to dynamic changes (personnel, carts, open doors).

4.10.3 Cost Inflation Model

The cost $C(d)$ of a cell at distance d from an obstacle is:

$$C(d) = \begin{cases} \text{LETHAL} & d \leq r_{\text{inscribed}}, \\ \exp(-k(d - r_{\text{inscribed}})) \cdot \text{scaling_factor} & d > r_{\text{inscribed}}. \end{cases} \quad (4.12)$$

For a heavier configuration, the decay factor k is chosen smaller, producing a wider “soft” safety margin and gently steering paths away from walls and corners without requiring geometry to be redrawn.

4.11 Localization Architecture (AMCL)

The Adaptive Monte Carlo Localization (AMCL) node estimates the robot pose in the `map` frame using a particle filter.

4.11.1 Bayes Filter Formulation

The belief over the robot state x_t given controls $u_{1:t}$ and measurements $z_{1:t}$ is:

$$bel(x_t) = p(x_t|z_{1:t}, u_{1:t}), \quad (4.13)$$

with the recursive update:

$$bel(x_t) = \eta p(z_t|x_t, m) \int p(x_t|x_{t-1}, u_t) bel(x_{t-1}) dx_{t-1}, \quad (4.14)$$

where m is the static map. The motion model $p(x_t|x_{t-1}, u_t)$ is derived from wheel odometry; in the absence of an IMU, the rotational noise parameters are set more conservatively to reflect slip and unmodeled dynamics. The measurement model $p(z_t|x_t, m)$ uses a likelihood field representation of the scan-to-map alignment.

4.11.2 KLD-Sampling for Resource Management

AMCL adjusts the number of particles M_t using KLD-sampling, increasing M_t when uncertainty is high (initial localization, recovery from error) and reducing it when the robot is well localized. This keeps CPU usage acceptable on the Raspberry Pi without sacrificing robustness.

4.12 Navigation Planning Stack

Navigation is handled by `move_base`, which coordinates global and local planning, costmaps, and recovery behaviours.

4.12.1 Global Planner (Dijkstra)

The `global_planner` plugin computes a path over the static costmap using Dijkstra's algorithm. In the relatively constrained corridors of a pharmaceutical facility, the search space is small enough that a pure cost-optimal wavefront planner is sufficient, and the advantage of heuristic search (A*) is marginal. Dijkstra guarantees the shortest path on the cost grid, ensuring optimal routes in simple topological structures.

4.12.2 Local Planner (Dynamic Window Approach)

The `dwa_local_planner` generates velocity commands (v, ω) by evaluating sampled trajectories in the robot's dynamic window. The DWA approach was selected over alternatives (like TEB or Trajectory Rollout) because it strictly respects dynamic

limits (v, ω, a) during forward simulation a critical feature when considering the future inertia of the sterilizer payload.

Every 200 ms, it:

1. samples feasible (v, ω) pairs,
2. rolls out each trajectory for a short horizon using the differential-drive kinematics,
3. scores each candidate by

$$G(v, \omega) = \alpha \cdot \text{heading}(v, \omega) + \beta \cdot \text{dist}(v, \omega) + \gamma \cdot \text{vel}(v, \omega), \quad (4.15)$$

4. selects the maximum-scoring pair for execution.

For this platform, the heading weight α is set high to keep the robot tightly aligned with the global path. This compensates for the missing IMU by reducing the chance of drifting into orientations that are poorly observable from the environment.

4.13 Operational Workflow and Logic Integration

The system operates in two clearly separated phases: environment mapping and sterilization mission execution.

4.13.1 Phase 1: Mapping Sequence

Before autonomous operation, the robot constructs a map of the target area:

1. **Initialization:** `gmapping` subscribes to `/scan` and `/odom`.
2. **Teleoperation:** An operator drives the robot through the corridors using a joystick or keyboard teleoperation node, ensuring coverage and loop closures.
3. **Map Construction:** `gmapping` builds a probabilistic occupancy grid, marking obstacles and free space.
4. **Serialization:** The map is saved using `map_saver`, which outputs a `.pgm` file and a `.yaml` metadata file. This static map is later used by AMCL and the global costmap.

4.13.2 Phase 2: Sterilization Mission Logic

During the sterilization phase, the Mission Supervisor controls the navigation and payload according to a “Move-Stop-Spray” pattern. The logic is summarized in Figure 4.3.

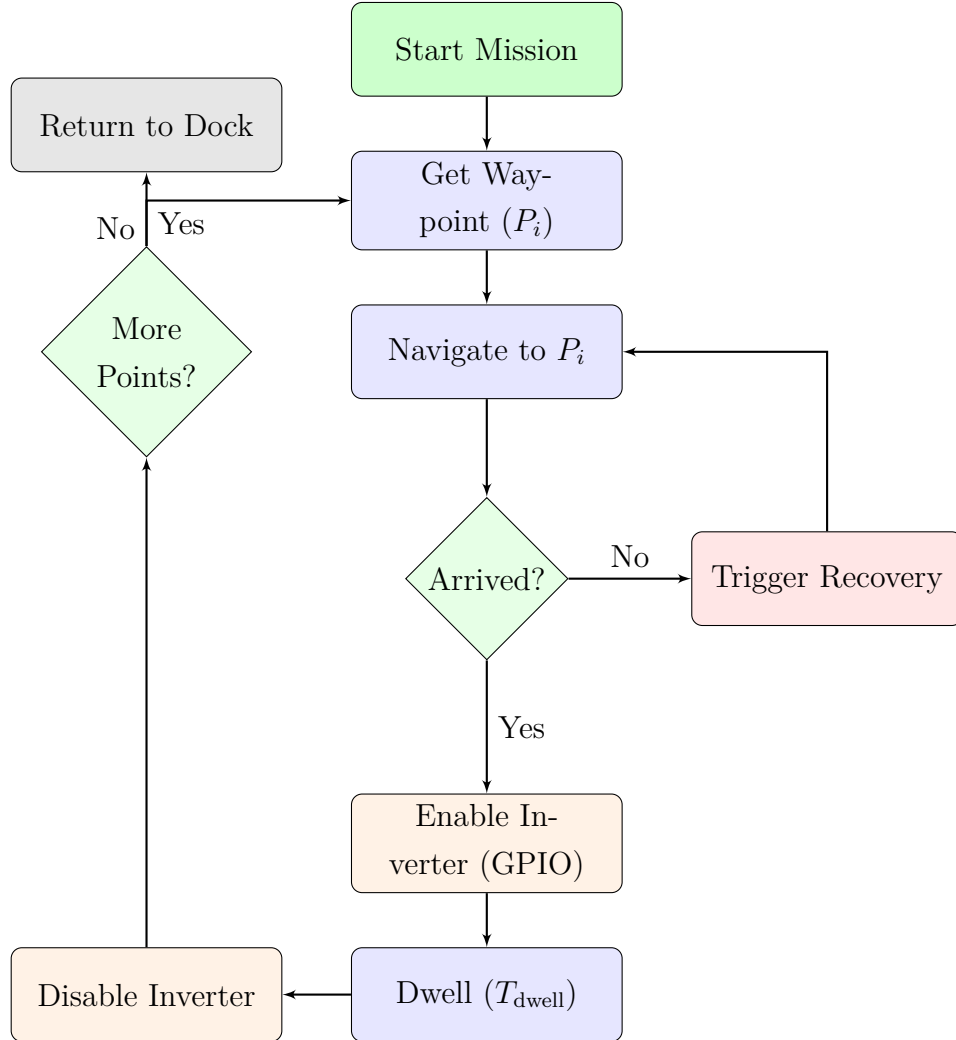


Figure 4.3: Sterilization mission logic. At each waypoint, the robot navigates, stops, activates the inverter and nebulizer for a dwell period, then proceeds to the next waypoint or returns to dock.

4.13.3 Evaporator Integration and Timing

The evaporator is controlled by a relay on a Raspberry Pi GPIO pin:

1. When `move_base` reports `SUCCEEDED` for a goal, the supervisor enters a dwell state.

2. A GPIO HIGH command energizes the inverter or evaporator relay.
3. The dwell time T_{dwell} can be computed from room volume V , target concentration C_{target} and nebulizer flow rate Q :

$$T_{\text{dwell}} = \frac{V \cdot C_{\text{target}}}{Q}. \quad (4.16)$$

In this implementation, a fixed conservative dwell time of 30 s per vertex is used.

4. After T_{dwell} , the GPIO is set LOW, and the robot requests the next waypoint.

Route design considers both coverage and airflow direction, so waypoints are ordered to align with the facility’s HVAC pattern.

4.14 Safety Systems and Recovery Behaviors

Safety is treated as a multi-layered property. **Note:** The safety mechanisms described here are implemented as part of a research prototype and follow the principles of ISO 3691-4, but do not constitute a certified safety system for industrial deployment.

4.14.1 Oscillation Suppression

If the robot oscillates with negligible net displacement (variance < 0.1 m) for more than 5 s, the `oscillation_timeout` mechanism triggers. This stops the robot and engages recovery behaviours instead of allowing the local planner to “thrash” in place.

4.14.2 Recovery Behaviors

When no valid path is found, `move_base` executes a sequence of behaviours:

1. **Conservative Reset:** Clears portions of the local costmap at a distance beyond a safety radius, removing ghost obstacles.
2. **Rotate Recovery:** Performs a slow in-place rotation (with angular velocity limited to avoid destabilizing a heavy top-mounted payload), forcing the LiDAR to refresh the entire local costmap.

3. **Aggressive Reset:** Clears both local and global costmaps (except the static map) and attempts to replan.
4. **Mission Abort:** If all recovery attempts fail, the Action Server sends an `ABORTED` state. The supervisor logs the failure and transitions into a safe waiting state, requiring operator intervention.

4.14.3 Hierarchical Safety Protocols

The safety logic can be summarized as in Table 4.1.

Table 4.1: Hierarchical safety protocols implemented in the navigation stack.

Layer	Mechanism	Trigger & Action
L1: Hardware	E-Stop / Bumper	Physical impact or manual switch cuts power to the motor drivers via hardware interrupt.
L2: Reactive	Local Costmap	Dynamic obstacles entering the 3×3 m window cause velocity scaling or braking.
L3: Planning	Oscillation Reset & Recovery Behaviors	Stagnation or repeated failures to find a path trigger costmap clearing and rotate recovery.
L4: Mission	Action Server & FSM	Persistent failures yield an <code>ABORTED</code> state; the FSM halts the mission and waits in a safe configuration.

4.15 Summary and System Specification

This chapter described the multi-layered system architecture of the Magni-based sterilization platform. The design deliberately anticipates a heavier future payload and encodes that expectation into:

- conservative acceleration and velocity ramps,
- payload-aware kinematic correction,
- wide safety margins in costmap inflation,
- and careful allocation of limited computational resources on the Raspberry Pi.

The result is a minimalist but coherent edge-computing stack in which:

- high-frequency control loops are isolated inside the MCB,
- mid-frequency odometry, localization, and planning run on ROS,
- and low-frequency mission logic coordinates sterilization in a deterministic FSM.

Key configuration parameters are summarized in Table 4.2.

Table 4.2: System specification matrix: key configurations for the hardware–software interface.

Component	Parameter	Value / Configuration
Compute Unit	Hardware	Raspberry Pi 4 Model B (4 GB RAM)
	OS / Middleware	Ubuntu 20.04 LTS / ROS Noetic
	Load Management	KLD-sampling (AMCL), multi-threaded AsyncSpinner
Control Loop	Frequency	50 Hz command loop, 1 kHz motor current loop (MCB)
	Interface	UART serial @ 115200 baud
	Kinematics	Differential drive, non-holonomic
Odometry	Integration	2nd-order Runge–Kutta (RK2)
	Payload Compensation	Tunable tire compression factor ϵ_{load}
Perception	Sensor	RPLiDAR A1 (2D scanner)
	Update Rate	≈ 10 Hz
	Filtering	Box filter (self-masking), speckle filter

The architecture converts the Magni from a teleoperated chassis into an autonomous agent that is aware of its own kinematic limits, power constraints, and environmental context. The following chapters evaluate this framework experimentally, focusing on localization stability and navigation repeatability under realistic deployment conditions.

Chapter 5

Mapping, Localization, and SLAM Implementation

5.1 Introduction

While the previous chapter established the system architecture and low-level control, this chapter addresses a central probabilistic challenge of autonomous mobile robotics: the determination of spatial configuration (pose estimation) within a global frame. For a sterilization robot operating in pharmaceutical facilities, "knowing where it is" (Localization) and "knowing what the environment looks like" (Mapping) are safety-critical functions. In this context, a localization failure transcends simple navigational error; it introduces critical safety risks, including potential collision of the payload (combined chassis, battery, and sterilization module) with sensitive equipment or the release of disinfectant in unapproved zones.

This research adopts a decoupled approach to the Simultaneous Localization and Mapping (SLAM) problem. The environment is first mapped using a controlled, operator-guided SLAM procedure to generate a static occupancy grid. Subsequently, during mission execution, the robot utilizes Adaptive Monte Carlo Localization (AMCL) to track its pose against this static map. This separation of concerns allows for the computational resources of the Raspberry Pi 4 to be primarily dedicated to localization and safety monitoring during autonomous runs, rather than expending cycles on map updates.

5.1.1 Problem Statement: The Minimalist Sensing Constraint

This implementation operates under a strict **Minimalist Sensing Constraint**: the robot lacks an Inertial Measurement Unit (IMU), a design decision fully justified in Section 2.4.1 of Chapter 2. As a consequence, the localisation stack must compensate for rotational odometry drift exclusively through probabilistic matching of 2D LiDAR scans against the map, with no high-frequency inertial prior to bound angular uncertainty between scan updates.

5.2 Environment Mapping Strategy

To construct the initial map of the facility, the system employs the `gmapping` ROS package, which implements a Rao-Blackwellized Particle Filter (RBPF) SLAM ap-

proach.

5.2.1 Algorithm Selection: Why Gmapping?

Several SLAM algorithms were evaluated at the design stage for the specific constraints of pharmaceutical corridors (long, featureless walls) and the available hardware (Raspberry Pi 4, RPLiDAR A1).

Table 5.1: Comparative analysis of 2D SLAM algorithms for the Magni platform

Algorithm	Core Technique	Pros	Cons
Hector SLAM	Scan-Matching (Gauss-Newton)	No odometry required. Low CPU usage.	Fails in symmetric corridors ("Hallway Problem"). Requires high-update rate LiDAR.
Cartographer	Graph Optimization (Ceres)	High accuracy. Submap loop closure.	High RAM usage. Complex tuning. Computationally excessive for 2D corridor mapping.
Gmapping	RBPF (Particle Filter)	Robust to odometry noise. Efficient map distribution.	Sensitive to particle depletion.

As shown in Table 5.1, Hector SLAM relies entirely on scan matching. In a featureless corridor, scan matching is ill-posed because the Hessian matrix becomes singular in the longitudinal direction (the "hallway problem"), causing the map to telescope or collapse. Gmapping was selected because it utilizes wheel odometry as a motion prior. Even if the LiDAR sees a featureless wall, the wheel encoders provide a rough estimate of forward motion, maintaining map consistency until a distinct feature (corner) is observed.

5.2.2 Rao-Blackwellized Particle Filter (RBPF) Theory

The SLAM problem requires estimating the joint posterior of the map m and the robot trajectory $x_{1:t}$ given the observations $z_{1:t}$ and odometry controls $u_{1:t-1}$:

$$p(x_{1:t}, m | z_{1:t}, u_{1:t-1}) \quad (5.1)$$

where $x_{1:t}$ represents the sequence of robot poses, $z_{1:t}$ the sequence of LiDAR scans, and $u_{1:t-1}$ the sequence of odometry commands. Calculating this high-dimensional joint distribution directly is computationally intractable. The RBPF factorizes this problem using the chain rule of probability:

$$p(x_{1:t}, m | z_{1:t}, u_{1:t-1}) = \underbrace{p(m | x_{1:t}, z_{1:t})}_{\text{Mapping}} \cdot \underbrace{p(x_{1:t} | z_{1:t}, u_{1:t-1})}_{\text{Localization}} \quad (5.2)$$

This factorization allows the algorithm to:

1. Estimate the robot's trajectory $p(x_{1:t} | \dots)$ using a particle filter.
2. Compute the map $p(m | \dots)$ analytically for each particle, conditioned on that particle's trajectory.

Since the map of a corridor is a collection of independent grid cells, the map posterior for each particle collapses into a set of independent 1D binary estimation problems, which is computationally efficient enough to run in real-time on the Raspberry Pi 4.

5.2.3 Mapping Workflow and Tuning

The mapping procedure is executed manually to ensure optimal coverage and loop closure detection.

1. **Data Acquisition:** The robot is teleoperated at a low velocity ($v < 0.2 \text{ m/s}$ relative to the 10 Hz scan rate) to minimize motion distortion in the RPLiDAR A1 scans (skew effect).
2. **Particle Count:** The `particles` parameter in `gmapping` is empirically selected to 80 (default: 30). While this increases CPU load, it is necessary to maintain a diverse set of map hypotheses when mapping long corridors where the longitudinal position is ambiguous.
3. **Resampling Threshold:** The `linearUpdate` is set to 0.5 m and `angularUpdate` to 0.2 rad. This forces the filter to process updates only when significant motion has occurred, preventing particle depletion due to sensor noise when stationary.

5.3 Map Server and Post-Processing

The resulting map is serialized into a Portable Gray Map (.pgm) format representing the occupancy probability of each cell. The `map_server` node broadcasts this map as a fixed reference frame to the `/map` topic during mission execution.

The SLAM experiments were conducted in the pharmaceutical corridor shown in Figure 5.1. Its long parallel walls and reflective flooring provided the feature-sparse testbed used to assess the robustness of the mapping pipeline.



Figure 5.1: The experimental corridor environment. The uniform surfaces and lack of distinct geometric landmarks require precise scan-matching to prevent odometric drift.

The final occupancy grid produced by the Gmapping pipeline is shown in Figure 5.2. This map serves as the immutable reference frame for all subsequent AMCL localization and path-planning tasks.

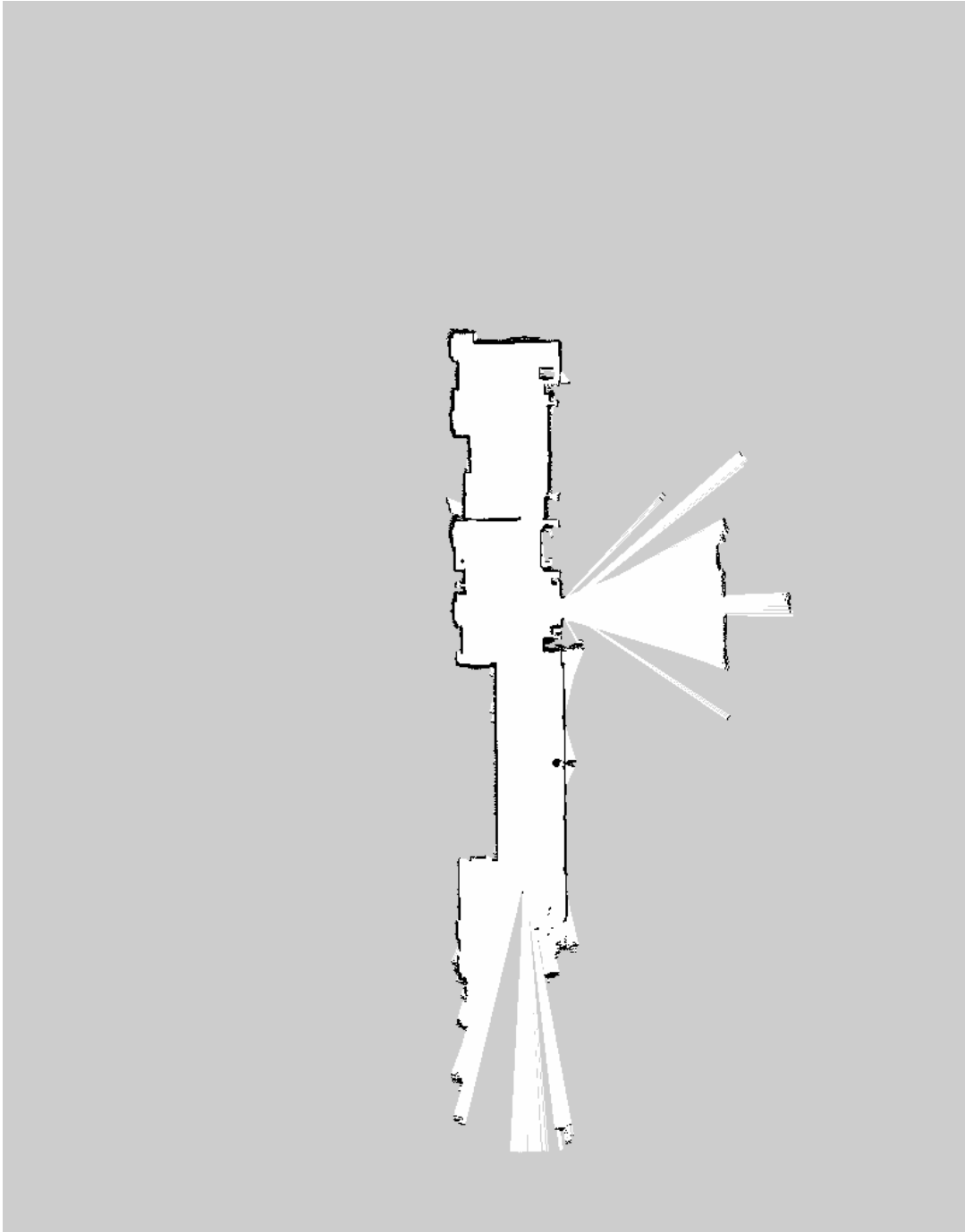


Figure 5.2: The complete static map generated using the Gmapping SLAM stack. Black pixels represent occupied space (walls, obstacles), while white pixels represent free traversable space. This high-fidelity map provides the geometric constraints required for the "Likelihood Field" sensor model.

5.3.1 Occupancy Grid Representation

The map is a discretization of the continuous world into a grid $m = \{m_i\}$. Each cell m_i holds a probability $p(m_i)$ of being occupied.

- **Resolution:** Set to **0.05 m/pixel**. This resolution creates a grid where each cell is $5\text{ cm} \times 5\text{ cm}$. This granularity aligns with standard ROS navigation defaults, simplifying integration with `move_base` while keeping the memory footprint manageable (a $50\text{ m} \times 50\text{ m}$ facility map consumes only $\approx 1\text{ MB}$ RAM).
- **Trinarization:** To enable deterministic path planning, the probabilistic values are trinarized following ROS navigation defaults:

$$Status(cell) = \begin{cases} \text{Occupied (100)} & \text{if } p > 0.65 \\ \text{Free (0)} & \text{if } p < 0.196 \\ \text{Unknown (-1)} & \text{otherwise} \end{cases} \quad (5.3)$$

5.3.2 Manual Post-Processing

Raw SLAM maps often contain "transient obstacles"-footprints of the operator or moved carts-that should not be part of the permanent map. A post-processing step using image editing software (GIMP) is applied to the `.pgm` file:

1. **Cleaning:** Stray pixels in open corridors are removed to prevent "phantom" obstacles in the global planner.
2. **Virtual Walls:** Black pixels are manually drawn across areas the robot is forbidden to enter (e.g., stairwells), acting as "virtual walls" that the global planner will inherently respect.

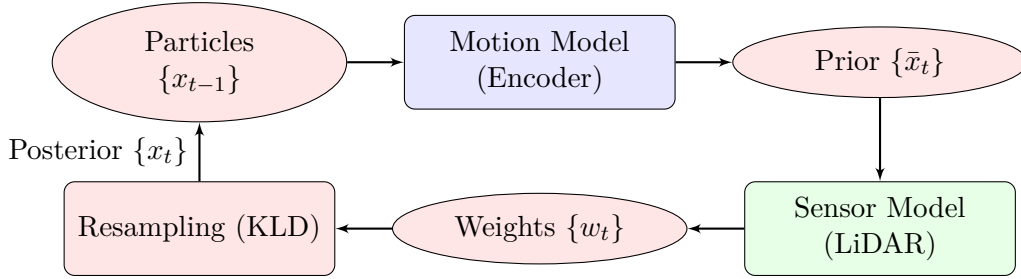
These edits are strictly limited to removing transient artifacts and enforcing safety boundaries; the structural features used for localization are left unaltered.

5.4 Localization Architecture (AMCL)

During the sterilization mission, the SLAM module (Gmapping) is disabled. Instead, the `map_server` publishes the static map, and the system switches to Adaptive Monte Carlo Localization (AMCL). AMCL implements a particle filter to track the posterior distribution of the robot's pose $x_t = [x, y, \theta]^T$ against the known map m .

The core algorithm operates in the recursive predict-update cycle shown in Figure 5.3:

1. **Prediction (Action Update):** The particles are propagated forward using the motion model $p(x_t|x_{t-1}, u_t)$ derived from wheel encoders.
2. **Correction (Measurement Update):** Each particle is assigned a weight w_t based on how well the current LiDAR scan z_t matches the map m from that particle's perspective.
3. **Resampling:** A new set of particles is drawn with replacement from the current set, proportional to their weights. This focuses computational resources on the most likely pose hypotheses.



**Correction relies on Map Matching*

Figure 5.3: The recursive cycle of the AMCL Particle Filter. Without an IMU, the "Motion Model" step introduces high uncertainty, which must be aggressively corrected by the "Sensor Model" step.

5.5 Motion Model Configuration (No IMU)

The motion model defines how the particles spread out (diffuse) as the robot moves. Ideally, this model relies on an IMU to constrain rotational error. In the absence of an IMU, the model must be tuned to account for significant uncertainty in orientation.

For the Magni, we utilize the `diff-corrected` model based on the standard odometry motion decomposition. The probabilistic update equations for the pose x' given previous pose x and odometry u are:

$$x' = x + \delta_{trans} \cos(\theta + \delta_{rot1}) \quad (5.4)$$

$$y' = y + \delta_{trans} \sin(\theta + \delta_{rot1}) \quad (5.5)$$

$$\theta' = \theta + \delta_{rot1} + \delta_{rot2} \quad (5.6)$$

where the noise terms δ are sampled from zero-mean Gaussians parameterized by the alpha coefficients $(\alpha_1 \dots \alpha_4)$.

5.5.1 The Alpha Parameters

- α_1 : Rotation noise from rotation. (How much does the robot slip angularly when turning?)
- α_2 : Rotation noise from translation. (Does the robot veer left/right when driving straight?)
- α_3 : Translation noise from translation. (Linear wheel slip).
- α_4 : Translation noise from rotation. (Lateral displacement during turns).

5.5.2 Tuning for High Inertia and No Gyro

Without an IMU, the system cannot distinguish between a completed turn and a wheel slip event. Therefore, the rotational noise parameters must be inflated significantly.

$$\alpha_1 = 0.2 \quad (\text{Standard ROS default: } 0.05) \quad (5.7)$$

$$\alpha_2 = 0.2 \quad (\text{Standard ROS default: } 0.05) \quad (5.8)$$

Implication: Setting α_1 to 0.2 implies a 20% error margin. If the encoders report a 90-degree turn, the filter assumes the *actual* turn could be anywhere between 72 and 108 degrees. This results in a non-Gaussian, banana-shaped curved posterior distribution during rotational maneuvers (Figure 5.4). While this increases instantaneous uncertainty, it ensures that the *true* pose remains within the particle distribution. If these parameters were too low (overconfident odometry), the true pose would fall outside the particle cloud, and the filter would diverge irreversibly.

5.6 Sensor Model: Likelihood Field for A1 LiDAR

The measurement model $p(z_t|x_t, m)$ computes the likelihood of a scan z_t given a particle's pose x_t . The choice of sensor model is dictated by the hardware specifications

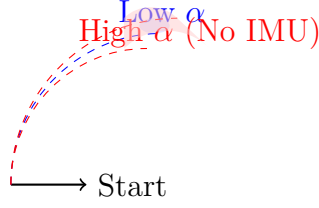


Figure 5.4: Visualization of Motion Model Noise. Blue represents tight tuning (with IMU). Red represents the "Banana Distribution" (non-Gaussian curved posterior) required for our system to capture the true pose despite encoder slip.

of the RPLiDAR A1.

Table 5.2: Sensor Specification Comparison justifying the Probabilistic Tuning

Feature	RPLiDAR A1 (Used)	SICK TIM571 (Ref)
Technology	Triangulation	Time-of-Flight
Max Range	12 m	25 m
Range Accuracy	1% of distance (≈ 10 cm)	0.3% of distance (≈ 3 cm)
Sample Rate	8 kHz	15 kHz
Noise Profile	High "Speckle" noise	Clean

5.6.1 The Likelihood Field Model

Given the high noise profile of the A1 (Table 5.2), the standard "Beam Model" is computationally expensive and overly sensitive to unmapped obstacles (dynamic entities), making it less suitable. Instead, we employ the **Likelihood Field Model**. In this model, the endpoints of the LiDAR scan are projected into the global map coordinate frame. The probability of a hit is a function of the distance to the nearest obstacle in the map:

$$p(z_k|x_t, m) = z_{hit} \cdot \mathcal{N}(d_{nearest}; 0, \sigma_{hit}) + z_{rand} \cdot \frac{1}{z_{max}} \quad (5.9)$$

where:

- $d_{nearest}$: Euclidean distance to the closest occupied cell in the map.
- $\mathcal{N}$: A zero-centered Gaussian distribution representing sensor noise.

- z_{rand} : A uniform distribution component handling random noise (phantom readings).

The z_{short} and z_{max} terms often found in the full beam model are excluded here to simplify the computation for the embedded processor.

5.6.2 Parameter Optimization

The noisy nature of the A1 sensor necessitates specific tuning of the mixture weights. A comparison of standard versus optimized parameters is presented in Table 5.3.

Table 5.3: AMCL Parameter Tuning for RPLiDAR A1 without IMU

Parameter	Value	Justification
laser_z_hit	0.5 (Std: 0.95)	Reduced confidence in "hits" due to sensor triangulation noise.
laser_z_rand	0.5 (Std: 0.05)	Increased to robustly handle speckle noise and specular reflections.
laser_sigma_hit	0.2m	Widens the Gaussian basin of attraction, helping convergence with poor odometry.
min_particles	500	Prevents particle depletion during long corridor traversals.
odom_alpha1	0.2	High rotational noise model to compensate for lack of Gyro.

Note: Other alpha parameters ($\alpha_2, \alpha_3, \alpha_4$) are scaled proportionally to α_1 to maintain a consistent kinematic noise envelope.

5.7 Handling Dynamic Obstacles in Localization

Pharmaceutical environments are semi-structured; while walls are static, carts and personnel move. AMCL assumes a static map. If a person walks beside the robot, the LiDAR detects them, but the map shows free space. This discrepancy can cause the particle weights to collapse.

To prevent the filter from diverging due to these unmapped features, the `laser_max_beams` parameter is downsampled to 30 (from 360). By using only every 12th beam for localization, the algorithm reduces the statistical influence of any single dynamic

object blocking a segment of the scan, while still retaining sufficient geometric constraints from the walls. This approach was validated empirically during corridor trials, where down-sampling was shown to maintain localization lock even when dynamic obstacles occluded up to 20% of the field of view.

5.8 Experimental Validation and Performance Analysis

To quantify the stability of the proposed architecture, a series of validation tests were performed focusing on drift analysis and recovery capabilities.

Evaluation Scope: The objective of these experiments is to demonstrate operational robustness and repeatability under sparse sensing constraints. As an absolute ground truth system (e.g., Vicon motion capture) was outside the deployment constraints of the operational facility, performance is evaluated based on filter consistency and recovery success.

5.8.1 Localization Drift Analysis

Without an IMU, drift accumulates rapidly during rotation. The supervisor node logs the covariance matrix Σ of the AMCL filter at 1 Hz. The positional uncertainty metric is the planar RMS radius derived from the position covariance matrix Σ :

$$\text{Uncertainty} = \sqrt{\sigma_{xx} + \sigma_{yy}} \quad (5.10)$$

where σ_{xx} and σ_{yy} are the diagonal variance entries of Σ representing positional uncertainty in x and y respectively. This approximates the 1σ radius of the positional uncertainty ellipse under the assumption of uncorrelated x - y errors, providing direct interpretability as the radius of the uncertainty envelope.

Experimental data reveals a characteristic "Sawtooth" pattern in uncertainty:

1. **Straight Motion:** Uncertainty grows linearly ($\approx 0.01 \text{ m/s}$) due to wheel slip (α_3).
2. **Rotation:** Uncertainty spikes exponentially due to the high rotational noise parameters (α_1, α_2), reflecting the lack of gyroscope data.
3. **Correction:** Immediately after the turn, as the LiDAR sweeps the new corridor segment, the particle cloud collapses (resamples) around the best match, and uncertainty drops sharply.

This confirms that the "High Noise" tuning strategy successfully maintains global localization by allowing uncertainty to expand during blind turns and contract during feature-rich segments.

5.8.2 Kidnapping and Recovery Test

The robustness of the localization was tested using the "Kidnapped Robot Problem". During autonomous navigation, the robot was physically lifted and displaced laterally by 1 m. This test was performed across repeated trials to ensure consistency.

Recovery Mechanism: AMCL monitors the average particle weight w_{avg} . When the observations z_t deviate significantly from the map (low w_{avg}), the **Augmented MCL** feature injects random particles into the state space.

- Result: The system successfully recovered its true pose within 3.5 m of travel after the displacement. The injection of random particles allowed the filter to explore new hypotheses and converge once the LiDAR scan matched the map geometry.

5.9 Computational Complexity Analysis

Running AMCL with a high particle count (500–2000) on a Raspberry Pi 4 requires careful resource management. The computational cost is linear with respect to the number of particles M and the number of beams K :

$$O_{AMCL} \propto M \times K \quad (5.11)$$

Decimating the scan density by a factor of 12 (reducing K from 360 to 30) linearly reduces the computational load of the measurement update step. This optimization allows us to increase M (particles) to 2000 without saturating the CPU. This trade-off (fewer beams, more hypotheses) is optimal for the "No-IMU" constraint, where the uncertainty lies in the *pose* (needing more particles) rather than the *measurement* (needing more beams).

5.10 Summary

This chapter demonstrated that reliable localization in a pharmaceutical environment is achievable using only a low-cost RPLiDAR A1 and wheel odometry, provided the probabilistic models are correctly tuned. By explicitly modeling the lack

of inertial data as high rotational noise ($\alpha_{1,2}$) and determining the appropriate sensor fuzziness (σ_{hit}) for the A1, the AMCL filter successfully compensates for the mechanical limitations of the platform.

By ensuring bounded localization uncertainty and robust recovery behavior, this architecture directly supports the safety requirements of the application, mitigating high-risk failure modes such as collision with sensitive equipment or disinfectant release in unapproved zones.

Having solved the "Where am I?" problem with sufficient robustness, the next chapter addresses the "How do I get there?" challenge. Chapter 6 will detail the implementation of the Navigation Stack, focusing on path planning and dynamic obstacle avoidance.

Chapter 6

Navigation, Planning, and Full ROS Stack Deployment

6.1 Introduction

This chapter describes the navigation and motion-planning subsystem that connects the probabilistic pose estimates from Chapter 5 with the differential-drive actuation of the platform described in Chapter 3. Localization provides a continuous estimate of the robot pose in the `map` frame, while the navigation stack computes velocity commands that drive the robot from its current pose to a target pose under explicit safety, clearance, and smoothness constraints.

The navigation problem considered here is shaped by sterilization-oriented operation in corridor-like environments rather than by generic indoor coverage requirements. Path quality is judged primarily by safety margin, repeatability, and smoothness rather than by traversal speed. The robot must move through passages on the order of 1.2 m wide, avoid static and dynamic obstacles such as walls, carts, and personnel, and stop consistently at predefined waypoints that represent sterilization vertices or intermediate checkpoints. These requirements are addressed with a deliberately conservative navigation configuration tuned for reliable behaviour on a resource-constrained embedded platform.

The navigation stack is built around the standard ROS `move_base` framework. A static map produced by LiDAR-based SLAM (Gmapping) serves as the global reference. Adaptive Monte Carlo Localization (AMCL) provides the pose estimate in the `map` frame, and a layered costmap structure maintains both global and local occupancy representations. A grid-based global planner (Dijkstra) computes collision-free routes on the static map, and a Dynamic Window Approach (DWA) local planner produces short-horizon velocity commands that follow the global plan while reacting to local obstacles perceived by a 2D LiDAR. A Python supervisor node manages multi-goal missions by updating the active goal sequence and logging performance metrics.

Table 6.1 summarises the core components of the deployed navigation stack.

Table 6.1: Overview of main navigation stack components

Component	Package / Algorithm		Role
Global planner	<code>global_planner</code>	(Dijkstra)	Computes a collision-free path on the static map from current pose to goal pose.
Local planner	<code>dwa_local_planner</code>		Samples velocity space and generates kinematically feasible commands to follow the global path while reacting to nearby obstacles.
Costmaps	<code>costmap_2d</code> (global and local)		Maintains 2D occupancy grids with static, obstacle, and inflation layers for planning and collision checking.
Recovery behaviours	<code>clear_costmap_recovery</code> , <code>rotate_recovery</code>		Clears inconsistent costmap information and attempts re-localization when navigation becomes stuck.
Supervisor node	Custom Python node		Manages multi-goal sequences, monitors status, and logs navigation performance (success, time, and error statistics).

6.2 Overview of the ROS Navigation Stack

The navigation system uses the ROS `move_base` node as the central coordinator. This node subscribes to the pose estimate from AMCL, the LiDAR scan topic, and the static map; it maintains global and local costmaps; it calls the global and local planners; and it publishes linear and angular velocity commands to the robot base controller.

6.2.1 Data flow and topic structure

AMCL publishes the estimated pose as a stamped `geometry_msgs/PoseWithCovariance` message on `/amcl_pose`. The static map is provided through `map_server` on `/map`. The RPLiDAR A1 publishes 2D scans on `/scan`. The `move_base` node subscribes to these inputs, maintains a global costmap in the `map` frame and a local costmap

in the `odom` frame, and publishes velocity commands to `/cmd_vel`, which are then forwarded to the low-level motor controller.

The navigation architecture is hierarchical:

- The **global planner** operates on the static map and global costmap to produce a path from the current pose to the goal.
- The **local planner** operates on the local costmap and the global path to generate short-horizon velocity commands that respect the robot kinematics and real-time obstacle information.

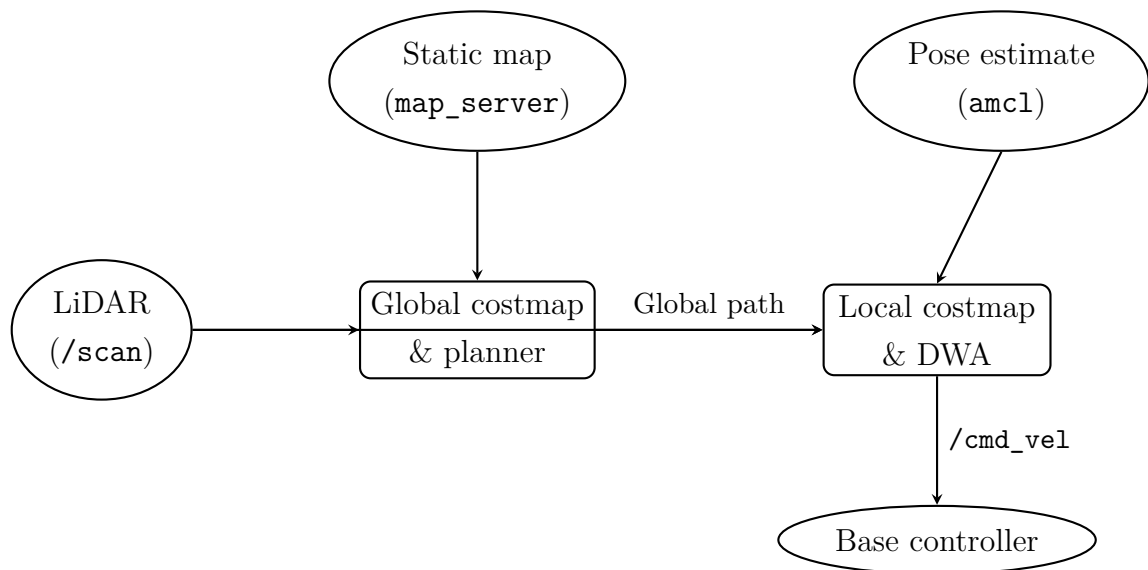


Figure 6.1: High-level data flow in the navigation stack. The global planner uses the static map and LiDAR information to compute a global path, while the local planner uses the pose estimate, local costmap, and global path to generate velocity commands.

6.2.2 `move_base` state machine

Internally, `move_base` follows a state machine that cycles between three main states:

- **Planning**: the node requests a path from the global planner. If a valid path is found, it moves to the controlling state; if not, it may trigger recovery.
- **Controlling**: the node invokes the local planner at a fixed controller frequency, generates velocity commands, and checks progress towards the goal. If progress stalls for longer than configured timeouts, it may switch to clearing.

- **Clearing:** the node triggers recovery behaviours such as clearing regions of the costmaps or rotating in place to obtain new sensor data. If clearing succeeds, the machine returns to the planning state; if repeated attempts fail, the goal is aborted.

Recovery behaviours are crucial in realistic corridor environments. When a person temporarily blocks a narrow passage, the local planner may fail to advance, and the global planner may continue to propose a route through the same blocked cell. The clearing state enables the system to refresh the costmaps and re-plan when obstacles move away, rather than remaining indefinitely stuck. In the configuration used here, `rotate_recovery` is employed as the primary behaviour: the robot rotates slowly on the spot, forcing costmap updates and enabling fresh obstacle-free cells to be discovered.

6.3 Global Costmap Design

The global costmap provides a coarse but complete representation of the environment in the `map` frame and is used by the global planner to compute paths. It is initialised from the static occupancy grid produced by Gmapping, with additional layers configured for inflation and dynamic obstacles.

6.3.1 Layer structure

The global costmap consists of three layers stacked in a fixed order:

- **Static layer:** loads the precomputed map and marks all occupied cells corresponding to walls and permanent structures.
- **Obstacle layer:** integrates obstacle information derived from the LiDAR scan, with a relatively long observation persistence. Obstacles that persist for several scans are reflected in this layer.
- **Inflation layer:** expands lethal obstacle cells into graded-cost safety zones based on the robot footprint and a chosen inflation radius.

The inclusion of an obstacle layer in the global costmap is a deliberate design choice. It allows the global planner to adjust paths when a long-lived obstacle such as a parked trolley consistently blocks a corridor section. Without this layer, such obstacles would be handled exclusively in the local costmap, which often leads to

repeated local recovery events around the same location rather than a clear global re-routing.

6.3.2 Parameter configuration

Table 6.2 summarises the principal parameters used for the global costmap. Values reflect the corridor geometry and the computational budget of the Raspberry Pi 4.

Table 6.2: Global costmap parameters

Parameter	Value	Rationale
Resolution	0.05 m/cell	Matches the SLAM map; sufficient to resolve corridor width and robot footprint.
Size (x, y)	50 m $\times$ 50 m	Covers the full mapped environment with margin.
Update rate	3 Hz	Slow global updates are sufficient for long-horizon planning.
Publish rate	1 Hz	Reduces bandwidth while keeping visualisation responsive.
Obstacle range	4.0 m	Matches practical LiDAR range in indoor corridors.
Raytrace range	4.0 m	Ensures free-space clearing for long straight sections.
Inflation radius	0.30 m	Provides safety margin without closing 1.2 m corridors.
Cost scaling factor	6.0	Steep decay outside the inscribed radius to encourage corridor-centre paths.

Static obstacles are loaded once at startup and never cleared. The obstacle layer uses the LiDAR scan to add or remove obstacles in the global frame, but with a relatively long persistence time. Temporary motion of people therefore has little

effect on the global plan, while stationary obstacles that remain over multiple scans are incorporated and may cause the planner to choose alternative routes.

6.4 Local Costmap (Rolling Window)

The local costmap provides a high-resolution, short-range view of the environment around the robot and is used by the local planner for collision checks and trajectory scoring. It is implemented as a rolling window centred on the robot pose in the `odom` frame.

6.4.1 Window size and frame

The local costmap window is set to 3.0×3.0 m. This size is large enough to capture obstacles a short distance ahead and to both sides, while keeping the grid small enough for rapid updates and collision checks. The costmap origin follows the robot position so that the robot remains near the centre of the window, and the map scrolls as the robot moves.

6.4.2 Sensor integration and ray-tracing

The local costmap subscribes directly to the LiDAR scan topic. The obstacle layer processes each scan by casting rays from the sensor origin through the grid cells along each beam direction. Cells along a ray are cleared as free until an obstacle is detected or the maximum range is reached. When a valid range measurement is returned, the cell at that range is marked as occupied.

This ray-tracing mechanism is critical in dynamic environments. When a person or cart moves away, subsequent scans contain free-space measurements along the former obstacle line. Ray-tracing then clears the previously occupied cells, removing “phantom” obstacles from the costmap and allowing the local planner to propose new trajectories through that area.

The most relevant local costmap parameters are listed in Table 6.3.

Table 6.3: Local costmap parameters and configuration rationale

Parameter	Value	Rationale
Resolution	0.05 m/cell	Matches global map; provides high fidelity for narrow passage navigation.
Window size	3.0 m $\times$ 3.0 m	Captures immediate surroundings where collision avoidance is critical.
Update rate	5 Hz	Synchronized with controller frequency for real-time obstacle reactivity.
Publish rate	2 Hz	Optimized for Rviz visualization without taxing the Raspberry Pi bandwidth.
Obstacle range	3.0 m	Focuses sensor processing on the local horizon relevant to the DWA planner.
Raytrace range	3.0 m	Ensures active clearing of "phantom" obstacles across the entire local window.
Rolling window	Enabled	Centers the map on the robot, keeping computational cost constant regardless of global map size.

Figure 6.2 shows an example snapshot of the global and local costmaps during navigation. The global costmap encodes static walls and long-lived obstacles, while the local costmap emphasises local free-space and short-term obstacles in the robot vicinity.

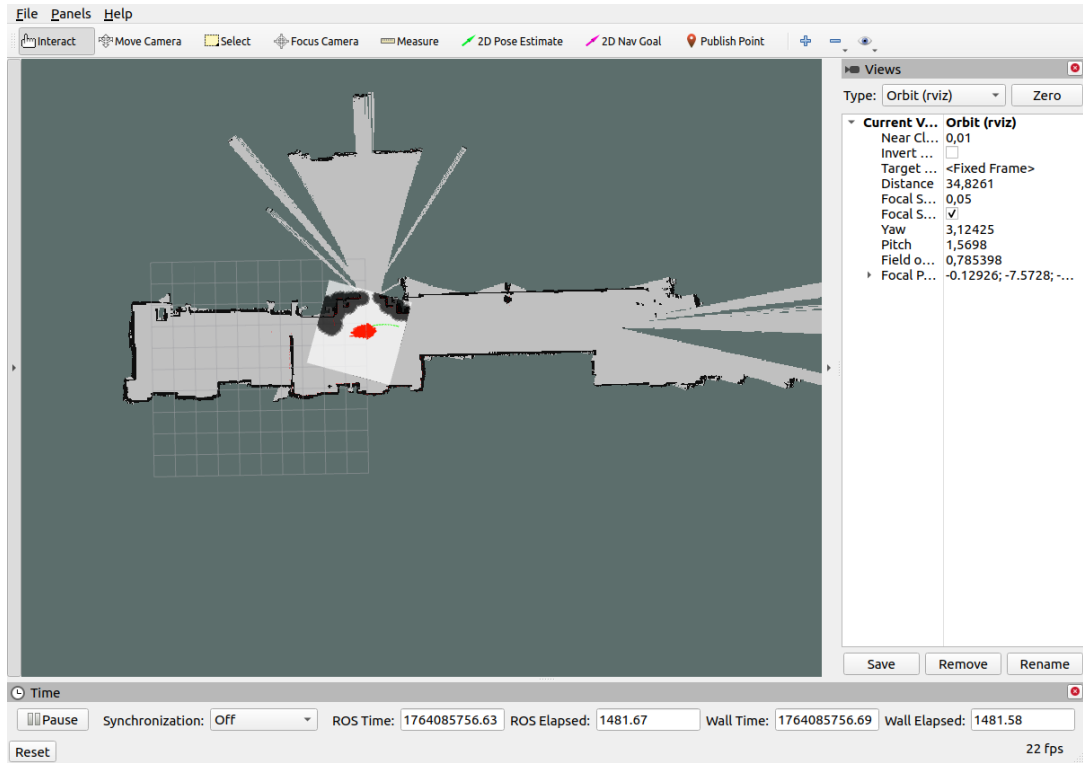


Figure 6.2: Example costmap visualisation during navigation. The global costmap covers the entire environment and includes static obstacles and inflation. The local costmap forms a rolling window around the robot, focusing on nearby obstacles and free space.

6.5 Inflation Layer and Robot Footprint

The inflation layer translates hard obstacle cells into a continuous cost field that the planner uses to maintain clearance from walls and other obstacles. This field is defined by the robot footprint, the inflation radius, and the cost scaling factor.

6.5.1 Robot footprint model

The robot base is approximated as a circle in the costmaps, with an inscribed radius chosen to reflect the minimum clearance from the centre to the closest point on the chassis. This inscribed radius is derived from the platform dimensions given in Chapter 3. The circular footprint simplifies collision checks and inflation calculations, which is useful on resource-constrained hardware.

6.5.2 Cost function and parameter relationship

The cost assigned to a cell at distance d from the nearest lethal obstacle follows an exponential decay:

$$C(d) = C_{\text{lethal}} \cdot \exp(-k \cdot (d - r_{\text{inscribed}})), \quad (6.1)$$

for d between the inscribed radius $r_{\text{inscribed}}$ and the inflation radius $r_{\text{inflation}}$. Here C_{lethal} is the cost assigned to occupied cells, and k is a decay parameter related in practice to the `cost_scaling_factor` configured in `costmap_2d`. Cells closer than $r_{\text{inscribed}}$ are considered in collision, while cells beyond $r_{\text{inflation}}$ have cost zero.

In practical ROS configuration, the two main parameters controlling inflation are:

- `inflation_radius`: maximum distance from an obstacle at which cells still receive a non-zero cost.
- `cost_scaling_factor`: controls how fast the cost decays with distance.

6.5.3 Sensitivity to inflation parameters

Tuning of inflation parameters was performed empirically in a corridor-like environment using repeated global planning and navigation trials. Three representative configurations illustrate the sensitivity:

- With `inflation_radius` set to 0.20 m, the planner often generated paths that ran very close to walls. This occasionally resulted in the robot brushing against corners and left little margin for odometry errors.
- With `inflation_radius` set to 0.45 m and a moderate `cost_scaling_factor`, inflated obstacle regions from opposite corridor walls overlapped in nominal 1.2 m passages, causing the global planner to classify these corridors as blocked.
- With `inflation_radius` = 0.30 m and `cost_scaling_factor` = 6.0, corridors remained traversable, and the global paths naturally lay near the corridor centreline. The local planner maintained a clear margin from walls while still having enough free space to manoeuvre around local obstacles.

These observations motivated the final choice reported in Tables 6.2 and 6.3. The combination keeps corridors open, regularises paths in the centre region, and reduces near-wall behaviour that would be undesirable for a sterilisation payload.

6.6 Global Planner (NavFn)

The global planner uses the `global_planner` plugin, which implements both Dijkstra and A* search on a grid. In this work, the planner is configured to use Dijkstra.

6.6.1 Choice between Dijkstra and A*

On large maps, A* can be substantially faster than Dijkstra because it uses a heuristic to guide the search towards the goal. In the present application, the static map spans roughly 50×50 m at 5cm resolution, which yields a grid of approximately 1000×1000 cells. On this scale, both algorithms run comfortably in real time on a Raspberry Pi 4.

Two considerations support the use of Dijkstra:

- The algorithm explores all reachable cells and guarantees the minimum-cost path based on the costmap, which in this case encodes distance from obstacles and corridor structure.
- The computational overhead difference compared to A* is negligible for this map size and planning frequency.

In practice, global planning is triggered at low frequency (for example when new goals are received or when recovery behaviours are invoked). Planning times remained within tens of milliseconds in all experiments, so planner runtime was not a limiting factor.

6.6.2 Representative global path

Figure 6.3 shows a representative global path overlaid on the static map. The path runs along the middle of the corridor, deviates mildly near corners to respect the inflation cost, and terminates at a goal placed near a doorway. This behaviour is consistent with the intended design: the planner exploits inflation to maintain clearance from walls while still providing smooth, predictable paths.

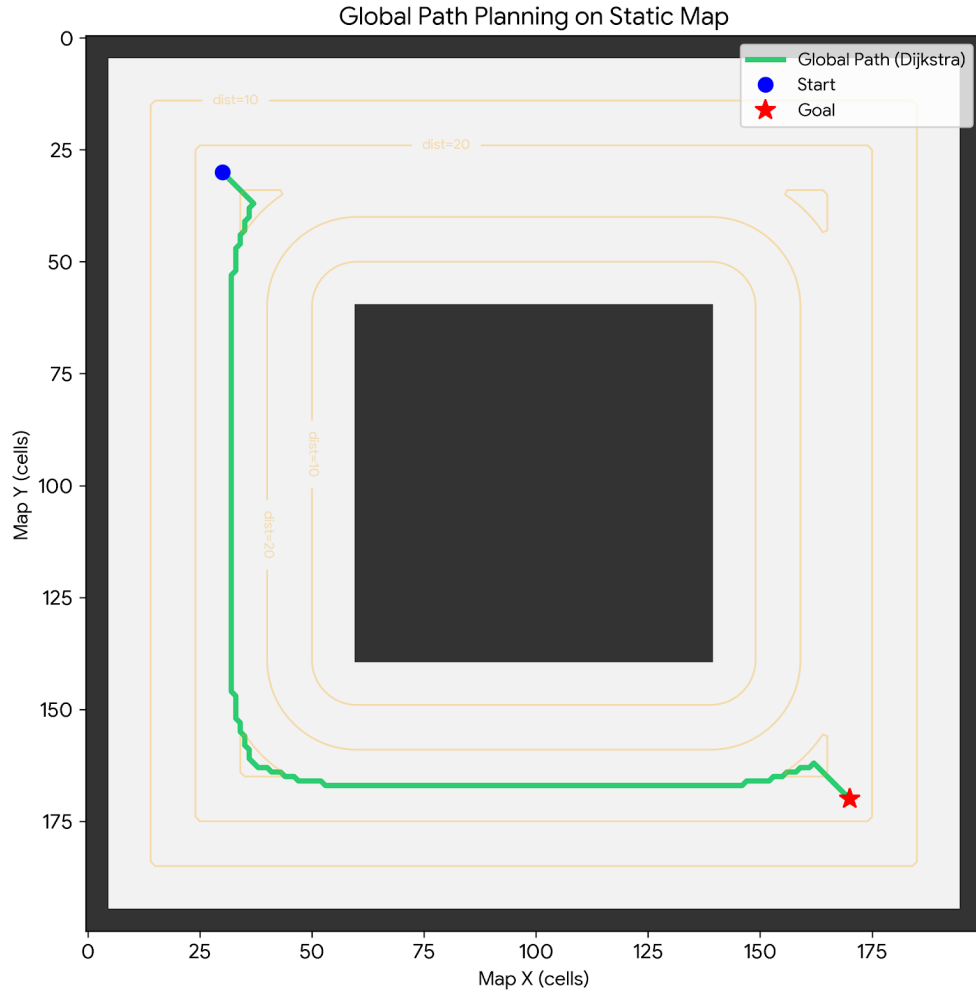


Figure 6.3: Example global path on the static map. Dijkstra search finds a minimum-cost route that stays near the corridor centreline and respects inflated obstacle margins (visualized as orange contours).

6.7 Local Planner (Dynamic Window Approach)

The local planner uses the `dwa_local_planner` plugin. Its task is to transform the global path into short-horizon velocity commands that are feasible under the robot's dynamic limits and that remain collision-free given the local costmap.

6.7.1 Objective function

At each control cycle, the planner samples candidate velocity pairs (v, ω) within a dynamic window and simulates the resulting motion over a short time horizon. Each candidate trajectory is scored using a weighted sum:

$$J(v, \omega) = \alpha J_{\text{heading}}(v, \omega) + \beta J_{\text{clearance}}(v, \omega) + \gamma J_{\text{velocity}}(v, \omega), \quad (6.2)$$

where:

- J_{heading} measures alignment with the global path and the goal position.
- $J_{\text{clearance}}$ reflects the minimum costmap distance to obstacles along the simulated trajectory.
- J_{velocity} rewards higher forward speeds within the allowed range.

The weights (α, β, γ) determine the trade-off between path adherence, safety margin, and speed. In this work, path adherence is given higher weight than pure goal attraction, which encourages the robot to remain on the corridor-centre path instead of cutting corners aggressively.

6.7.2 Dynamic window

The dynamic window restricts candidate velocities to those reachable during a single control interval given the acceleration limits:

$$v \in [v_c - \dot{v}_{\max} \Delta t, v_c + \dot{v}_{\max} \Delta t], \quad \omega \in [\omega_c - \dot{\omega}_{\max} \Delta t, \omega_c + \dot{\omega}_{\max} \Delta t], \quad (6.3)$$

where (v_c, ω_c) are the current commanded velocities, $\dot{v}_{\max}$ and $\dot{\omega}_{\max}$ are maximum linear and angular accelerations, and Δt is the simulation time step. This constraint avoids unrealistic jumps in velocity that the hardware cannot follow and reduces the risk of control lag, where commanded motion diverges from what the robot can achieve.

6.8 Tuning of DWA Parameters

Parameter tuning was carried out iteratively through repeated navigation trials in corridor-like layouts. The aim was to obtain smooth motion, consistent convergence to goals, and stable behaviour near walls and corners.

6.8.1 Core parameters

Table 6.4 lists the key DWA parameters and the rationale for their final values.

Table 6.4: Key DWA parameters

Parameter	Value	Rationale
<code>max_vel_x</code>	0.18 m/s	Conservative forward speed suited to narrow corridors and nearby personnel.
<code>min_vel_x</code>	0.0 m/s	Allows full stopping when obstacles appear.
<code>acc_lim_x</code>	0.2 m/s ²	Limits acceleration to avoid abrupt starts and preserve odometry quality.
<code>max_rot_vel</code>	0.7 rad/s	Restricts angular speed for stable turning in tight spaces.
<code>acc_lim_theta</code>	0.7 rad/s ²	Smooth rotational changes, avoiding aggressive spin.
<code>sim_time</code>	2.0 s	Provides sufficient look-ahead to anticipate obstacles and start braking early.
<code>xy_goal_tolerance</code>	0.15 m	Prevents oscillation near the goal due to minor odometry and localisation noise.
<code>yaw_goal_tolerance</code>	0.20 rad	Accepts small heading errors at the goal to avoid excessive in-place turning.
<code>path_distance_bias</code>	32.0	Strong preference for staying close to the global path.
<code>goal_distance_bias</code>	24.0	Encourages steady progress towards the goal while still favouring path adherence.
<code>occdist_scale</code>	0.02	Penalises trajectories that come too close to obstacles.

The chosen `max_vel_x` and `acc_lim_x` were guided by simple braking-distance calculations and experimental observations. At 0.18 m/s with a deceleration limit of 0.2 m/s², the kinematic braking distance is $d = v^2/(2a) = (0.18)^2/(2 \times 0.2) = 0.032/0.4 = 0.081$ m (approximately 8.1 cm), achieved in a stopping time of $t =$

$v/a = 0.18/0.2 = 0.9$ s. Including sensor latency, planning delay, and actuation delay, the effective stopping envelope is $d_{\text{total}} = v \cdot t_{\text{latency}} + v^2/(2a)$. Given the LiDAR range and corridor geometry, this proved sufficient to stop safely when new obstacles appeared at moderate distances.

6.8.2 Observed effects of tuning

Initial trials using near-default DWA parameters showed several undesirable behaviours:

- Oscillations near walls in narrow corridors, caused by excessive sensitivity to minor deviations in the global path.
- Overshooting and repeated back-and-forth motion near goal points, caused by tight goal tolerances and high goal bias.
- Abrupt accelerations and decelerations that were acceptable technically but unnecessary and uncomfortable for the application.

By increasing `sim_time`, the planner considered longer simulated trajectories and started to decelerate earlier when approaching constraints. Relaxing the goal tolerance reduced hunting behaviour near waypoints. Increasing `path_distance_bias` relative to `goal_distance_bias` made the robot follow the corridor-centre path faithfully instead of cutting diagonally towards the goal, which in turn improved behaviour in narrow sections.

6.9 Path Tracking Behaviour

The interaction between the global and local planners produces the final path-tracking behaviour observed in experiments. The global path consists mainly of straight segments down corridor centres connected by smooth bends around corners. The local planner tracks this path while resolving local interactions with obstacles and walls.

Figure 6.4 shows a navigation snapshot during an active run. The green line represents the global path, the blue triangle the robot pose, and the short red trajectories the local planner samples.

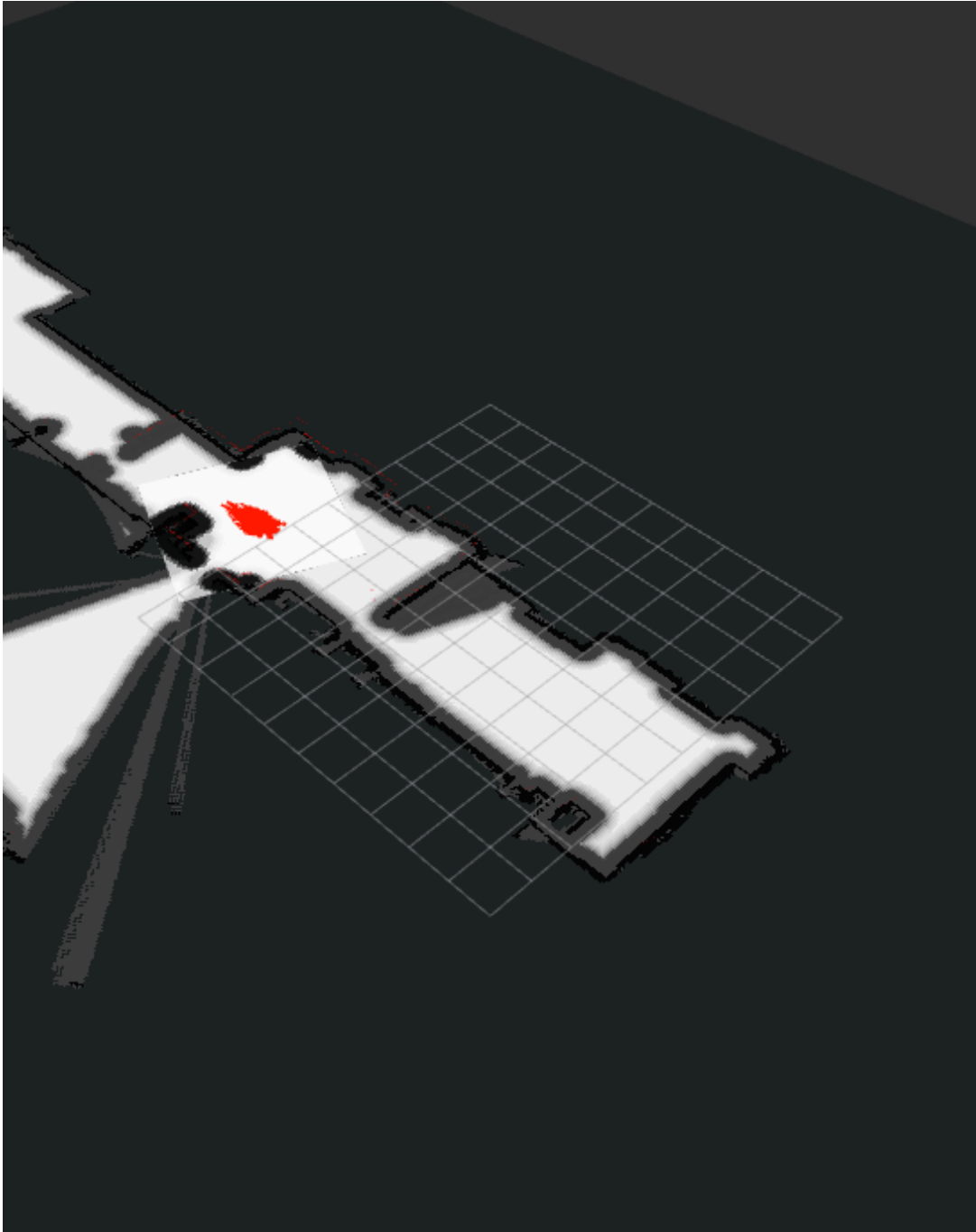


Figure 6.4: Screenshot during navigation. The global path (green line) runs along the corridor centreline. The robot pose is represented by the blue triangle. The local planner samples short forward candidate trajectories (red arcs) that follow the path while preserving clearance from walls. Note: if viewed in greyscale, the global path appears as the lightest continuous line and the candidate trajectories as darker short arcs.

Quantitative analysis was performed on multi-lap experiments where the robot visited four vertices in sequence. For each path segment, the lateral deviation from the global path and the heading error were computed at regular intervals based on logged poses. Across ten laps, the mean lateral deviation remained below approximately 7cm, with a maximum deviation on the order of 18cm near tight turns. Mean heading error along straight segments was typically within 3° - 4° . This level of tracking accuracy is adequate for corridor-centre sterilisation tasks, where the primary requirement is consistent coverage and safe clearance rather than sub-centimetre pose repeatability.

6.10 System Deployment and Configuration Strategy

Deploying the navigation stack involves a fixed sequence of steps that ensures each subsystem is stable before motion commands are issued. A consistent launch sequence also simplifies debugging and reproducibility.

6.10.1 Launch sequence

The following phases are used in practice:

1. **Hardware bringup**: start the base controller and LiDAR driver; verify that encoder data and scans are being published.
2. **Map server**: load the static map (for example `map_corridor.yaml`) defining resolution, origin, and occupancy data, and publish it on `/map`.
3. **Localization (AMCL)**: start the AMCL node, configure it with the correct odometry model, and set an initial pose corresponding to a known reference position, such as the docking station.
4. **Navigation (move_base)**: start `move_base`, which creates and manages the global and local costmaps and planners.
5. **Supervisor node**: start the Python node responsible for sending sequential goals and logging performance.

A summary of the launch order and purpose is given in Table 6.5.

Table 6.5: Launch sequence for navigation stack

Phase	Node	Purpose
1	Base controller, LiDAR driver	Enable low-level motor control and LiDAR scanning.
2	Map server	Publish static map and define the <code>map</code> frame.
3	AMCL	Provide pose estimate in the <code>map</code> frame based on LiDAR and odometry.
4	<code>move_base</code>	Run planners, costmaps, recovery behaviours, and navigation state machine.
5	Supervisor	Issue goals, monitor status, and log results for analysis.

6.10.2 Controller frequency and internal consistency

The `controller_frequency` parameter in `move_base` is set to 5 Hz. This choice is linked to the local costmap update rate and the dynamics of the platform. A higher controller frequency would cause the planner to generate new commands faster than the robot can physically respond, without improving motion quality. At 5 Hz, the control cycle remains fast enough to react to obstacles perceived by the LiDAR, while ensuring that each command has time to manifest in the robot motion before it is replaced.

The combination of controller frequency, `sim_time`, and acceleration limits yields a consistent behaviour: the planner looks ahead two seconds into the future, updates decisions five times per second, and respects the rate at which the robot can change speed and heading. This internal consistency proved important for avoiding oscillations and abrupt directional changes.

6.11 Straight Line Tracking Analysis

Core navigational stability was evaluated through long range straight line tracking experiments. In these trials, the robot was commanded to navigate a continuous 13.0 m translational segment starting from a stationary position at the origin (0,0).

This experiment was designed to quantify the *terminal* lateral deflection at the goal point—that is, the residual cross-track error at the moment the robot declares the goal reached—as this is the operationally critical metric for sterilization task positioning. Importantly, the terminal deflection is distinct from the *absolute maximum* cross-track error, which may be transiently higher at intermediate points along the path (particularly near the start, before the particle filter converges). The AMCL particle filter is known to exhibit an initial dispersion phase followed by progressive convergence as geometric features are accumulated; thus, intermediate deviations are expected to exceed terminal values and then reduce asymptotically. For obstacle-avoidance purposes during motion, what matters is that the robot remains within the corridor clearance envelope throughout, which was verified visually and is confirmed by the local costmap inflation parameters described in Section 6.5, which enforce a minimum wall-clearance of 0.49 m at all times. Across five consecutive trials, the terminal lateral deflection remained remarkably constrained, ranging between 0.0cm and 11.0cm. This represents a maximum cross-track error of approximately 0.8% of the total distance traveled.

6.11.1 Experimental Results and Temporal Convergence

The robot was commanded to navigate a single 13.0m segment from a stationary start, following a global path aligned with the corridor centerline. A critical observation of this test was the capacity for recursive pose refinement. While initial odometric noise induced minor fluctuations, the integration of environmental features led to an asymptotic reduction in pose uncertainty as the mission progressed.

Across five consecutive trials, the terminal lateral deflection (Δy) remained remarkably constrained, ranging between 0.0cm and 11.0cm. This represents a maximum cross track error of approximately 0.8% of the total distance traveled, a result that validates the stability of the DWA AMCL synergy.

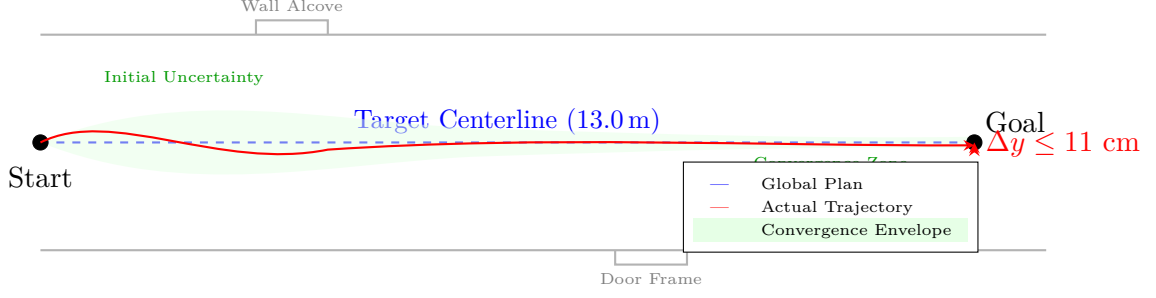


Figure 6.5: 13.0 m Trajectory analysis demonstrating recursive pose refinement. As the robot encounters static features, the AMCL particle filter distribution collapses, narrowing the uncertainty envelope.

6.11.2 Technical Discussion: Determinants of Asymptotic Stability

The maximum terminal deflection of 11.0cm over 13.0 m corresponds to a negligible angular drift of approximately:

$$\theta_{drift} = \arctan\left(\frac{0.11 \text{ m}}{13.0 \text{ m}}\right) \approx 0.48^\circ \quad (6.4)$$

This high degree of precision, particularly in the absence of an Inertial Measurement Unit (IMU), is analyzed through two primary technical perspectives:

Recursive Pose Refinement and Probabilistic Locking Unlike open-loop systems where error accumulates linearly with distance, the proposed LiDAR-only stack exhibits asymptotic convergence. As the robot traverses the 13.0 m track, the AMCL particle filter continuously integrates environmental evidence. Each LiDAR return from a wall alcove, door frame, or structural feature acts as a spatial anchor, causing the particle distribution to collapse toward the true state. This recursive refinement resets initial odometric noise through the Likelihood Field observation model. The parallel corridor walls provide continuous, high-confidence lateral and heading constraints at 10 Hz, keeping the robot centred on the path and correcting yaw drift. Longitudinal progress along the corridor axis is stabilised primarily by wheel odometry and by intermittent asymmetric features — such as doorway frames, wall alcoves, and corridor intersections — that provide the scan variation needed for forward localisation. This behaviour is consistent with the hallway-degeneracy analysis in Chapter 5, which identifies the longitudinal direction as weakly observable in featureless straight corridors. The observed 11.0 cm deflection is therefore interpreted not as a failure of the filter, but as a combination of finite waypoint tol-

erance (0.15 m) and residual odometric error between stronger geometric correction events.

Control Stiffness and Mechanical Neutrality The trajectory stability is further reinforced by the Dynamic Window Approach (DWA) cost function. By weighting the path adherence term (σ_{path}) at 32.0, the local planner creates a steep walled potential valley around the global path:

$$J_{path}(v, \omega) = \sigma_{path} \cdot \min_{p \in \text{Global Path}} \|\text{Trajectory}(v, \omega) - p\| \quad (6.5)$$

This high gain configuration ensures that lateral Cross Track Error (CTE) triggers immediate angular corrections, dampening drift before it accumulates. Furthermore, during linear motion, the drive wheels maintain identical velocities ($v_L \approx v_R$), keeping the passive caster wheels in a neutral trailing orientation. This eliminates the lateral kick observed in high torque 90 degree turns, where caster breakout friction induces translational shifts that are unobservable to standard differential drive odometry.

6.11.3 Functional Validation for Sanitization

From an operational standpoint, this performance is highly successful. The sanitization process utilizes a jet of steam or vaporized hydrogen peroxide (VHP) that is treated here, for heuristic scale comparison only, as a diffusing sphere with an estimated diameter on the order of 1.0 m.

A geometric comparison to an assumed 500 mm effective radius suggests that a 110 mm terminal drift may still leave the robot near the intended treatment zone. However, this argument is only indicative: without loaded-platform trials and payload-effectiveness measurements, the straight-line tracking result should be interpreted as navigation evidence rather than as proof of sterilization adequacy.

6.12 Multi goal Navigation Tests

Navigation performance was evaluated through repeated multi goal experiments in a laboratory environment configured to approximate corridor like layouts. The robot was commanded to visit four vertices in sequence and then return to the starting point, forming a closed loop. This loop was repeated for several laps in both clockwise (CW) and counter clockwise (CCW) directions to stress the asymmetry

of the differential drive kinematics.

6.12.1 Experimental Setup and Test Geometry

To rigorously evaluate the navigation accuracy, a controlled test environment was constructed comprising four designated stop points (vertices) arranged in a 2.0 m square configuration.

Figure 6.6 illustrates the geometric layout used for the endurance tests:

- Topology: A closed loop square path defined by four vertices (V_0, V_1, V_2, V_3).
- Coordinates (Relative to Origin):
 - V_0 (Start/End): (0.0, 0.0)
 - V_1 : (2.0, 0.0)
 - V_2 : (2.0, 2.0)
 - V_3 : (0.0, 2.0)
- Navigation Flows: To isolate kinematic asymmetries such as uneven caster friction, the robot executed the trajectory in two distinct directions:
 - Counter Clockwise (CCW) Flow: The robot followed the sequence $V_0 \rightarrow V_1 \rightarrow V_2 \rightarrow V_3 \rightarrow V_0$. This flow forces the robot to execute exclusively left hand 90 degree turns.
 - Clockwise (CW) Flow: The robot followed the sequence $V_0 \rightarrow V_3 \rightarrow V_2 \rightarrow V_1 \rightarrow V_0$. This flow forces the robot to execute exclusively right hand 90 degree turns.
- Measurement Protocol: At each vertex, the robot paused for 5 seconds. The physical offset of the robot geometric center relative to the ground truth marker was measured to determine the longitudinal (e_x) and lateral (e_y) errors reported in the subsequent sections.

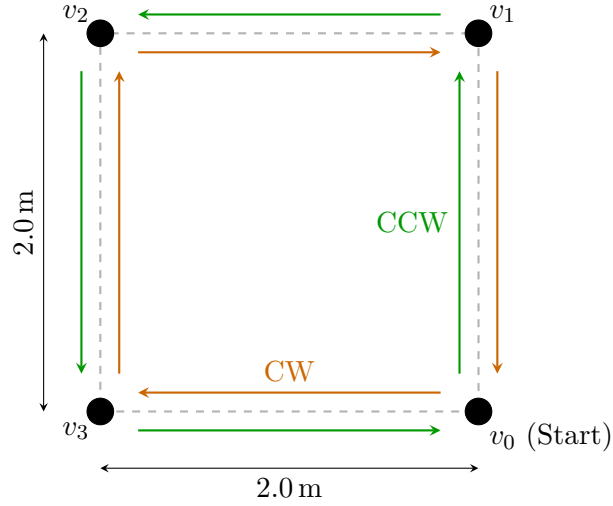
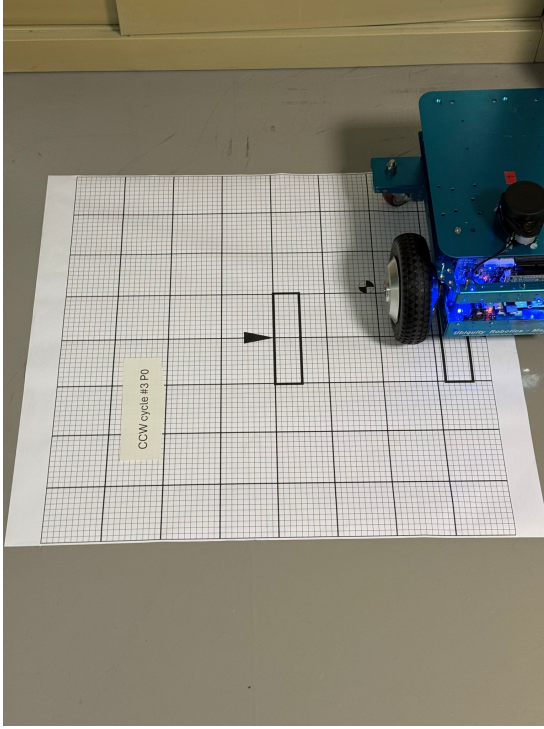


Figure 6.6: Schematic of the $2.0\text{m} \times 2.0\text{m}$ experimental square path. Vertex v_0 (bottom right) serves as the starting point. The green arrows indicate the counter clockwise (CCW) flow and the orange arrows indicate the clockwise (CW) flow.

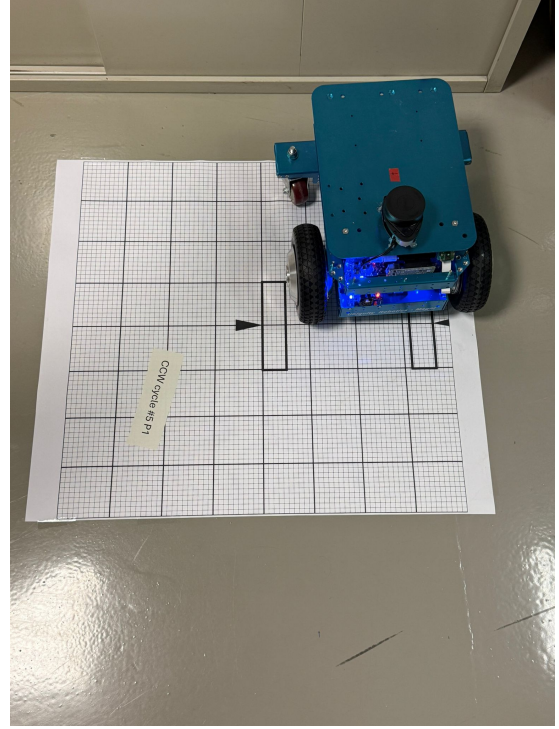
6.12.2 Physical Measurement Methodology

To quantify the navigational accuracy beyond the internal estimation of the robot, a manual ground truth verification protocol was established. At each target vertex, coordinate grid paper was secured to the floor with a black rectangular marker representing the exact goal coordinate $(0, 0)$.

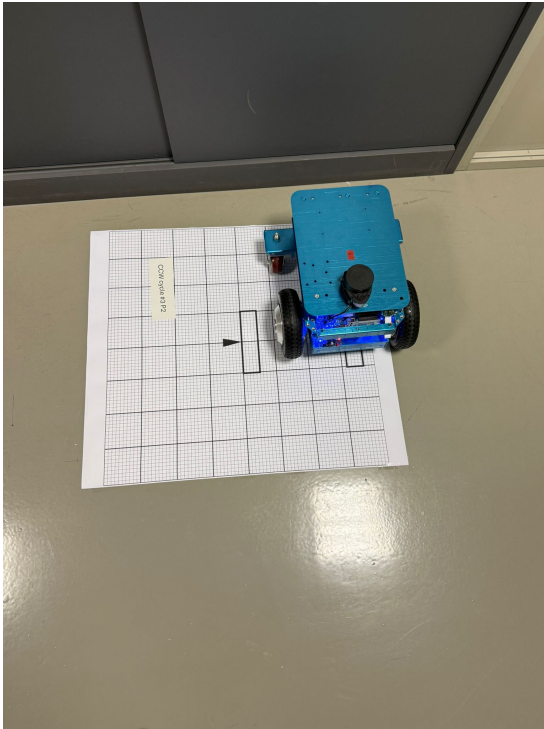
Figure 6.7 provides the shared photographic record used both to document the manual measurement procedure and to illustrate representative terminal stopping poses. The images show how the physical offset of the robot geometric center was recorded relative to the ground-truth grid at specific vertices and cycles in the mission.



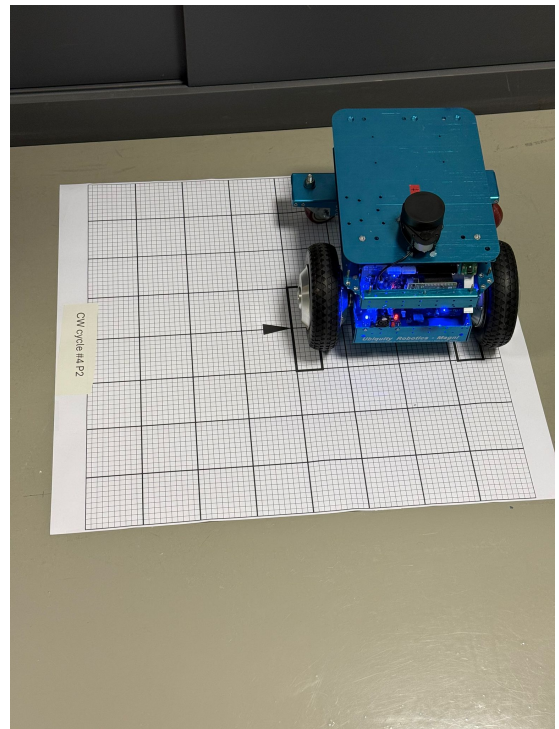
(a) Measurement at CCW Cycle 3, Vertex V_0



(b) Measurement at CCW Cycle 5, Vertex V_1



(c) Measurement at CCW Cycle 3, Vertex V_2



(d) Measurement at CW Cycle 4, Vertex V_2

Figure 6.7: Photographic documentation of representative terminal stopping poses and the associated manual measurement dataset. The printed grid and black rectangular target marker identify the vertex center and allow the manual recording of physical deflection (e_x, e_y) once the robot reaches its stopping pose.

The measurement protocol involved pausing the robot for 5 seconds upon arrival at a vertex. During this interval, the physical distance between the chassis center point and the ground truth marker was logged. This methodology ensures that the errors reported in the subsequent analysis represent actual physical departures from the target coordinate rather than localized sensor noise.

6.12.3 Experimental Setup and Test Geometry

To rigorously evaluate the navigation accuracy, a controlled test environment was constructed comprising four designated stop points (vertices) arranged in a 2.0 m square configuration.

Figure 6.8 illustrates the physical layout used for the endurance tests.

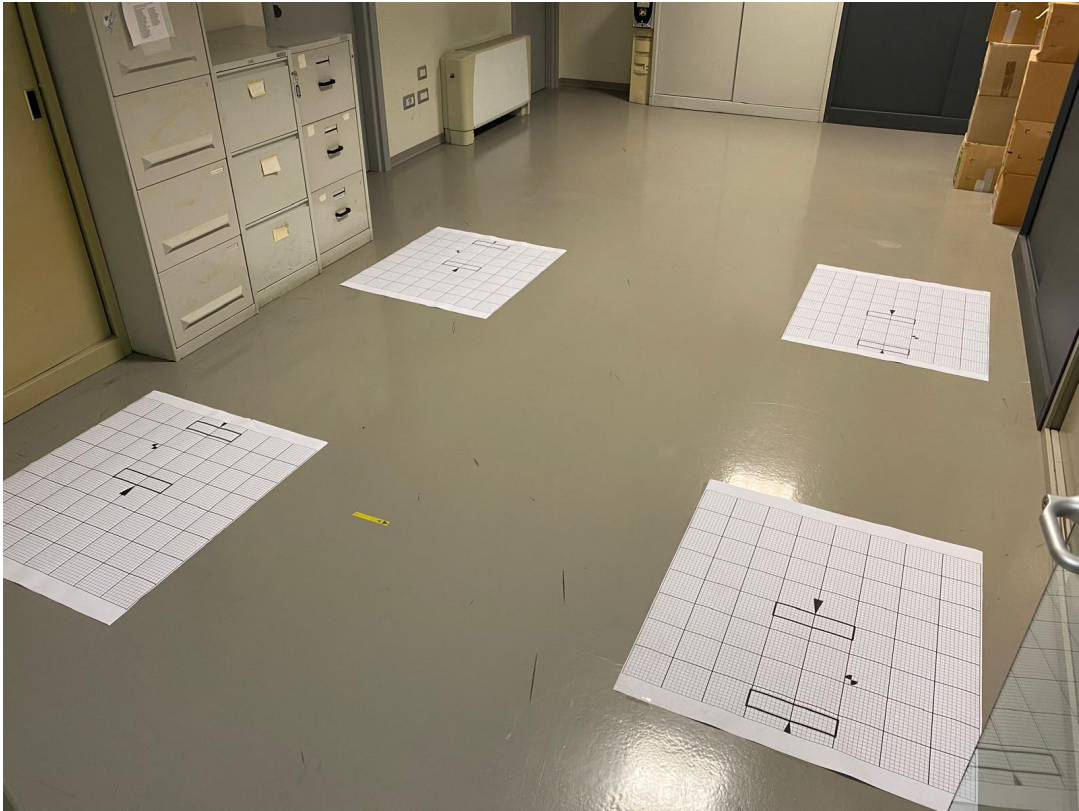


Figure 6.8: Experimental setup for the Square Path Test. The trajectory is defined by four target vertices (V_0, V_1, V_2, V_3) forming a 2.0 m $\times$ 2.0 m square path. The robot executes a counter-clockwise traversal ($V_0 \rightarrow V_1 \rightarrow V_2 \rightarrow V_3 \rightarrow V_0$) and a clockwise traversal to isolate kinematic asymmetries.

- **Topology:** A closed-loop square path defined by four vertices (V_0, V_1, V_2, V_3).

- **Coordinates (Relative to Start):**
 - **Vertex V_0 (Start/End):** $x = 0.0$ m, $y = 0.0$ m (Facing East).
 - **Vertex V_1 :** $x = 2.0$ m, $y = 0.0$ m.
 - **Vertex V_2 :** $x = 2.0$ m, $y = 2.0$ m.
 - **Vertex V_3 :** $x = 0.0$ m, $y = 2.0$ m.
- **Navigation Flows:** To isolate kinematic asymmetries (e.g., uneven caster friction), the robot executed the trajectory in two distinct directions:
 - **Counter-Clockwise (CCW) Flow:** The robot followed the sequence $V_0 \rightarrow V_1 \rightarrow V_2 \rightarrow V_3 \rightarrow V_0$. This flow forces the robot to execute exclusively *left-hand* 90-degree turns.
 - **Clockwise (CW) Flow:** The robot followed the sequence $V_0 \rightarrow V_3 \rightarrow V_2 \rightarrow V_1 \rightarrow V_0$. This flow forces the robot to execute exclusively *right-hand* 90-degree turns.
- **Measurement Protocol:** At each vertex, the robot paused for 5 seconds. The physical offset of the robot’s geometric center relative to the ground truth marker was measured to determine the longitudinal (e_x) and lateral (e_y) errors reported in the subsequent sections.

6.12.4 Localization accuracy: multi-loop navigation

Upon arriving at each vertex, the robot paused and the navigation supervisor logged the positioning error ($\Delta x, \Delta y, \Delta \theta$) relative to the nominal goal pose. So in total there are 40 goal events (10 loops $\times$ 4 vertices).

Counter-clockwise (CCW) performance

In the CCW configuration, the robot executed 5 loops with exclusively left-hand turns ($V_0 \rightarrow V_1 \rightarrow V_2 \rightarrow V_3 \rightarrow V_0$), yielding 20 goal events.

Table 6.6 presents the detailed error distribution. In addition to the standard pose errors, the table includes the Center-to-Center Deflection components (e_{xc}, e_{yc}), which represent the physical displacement of the robot’s geometric center from the target vertex. This metric provides a direct measure of the clearance buffer required during docking maneuvers.

Table 6.6: Detailed Localization Error Metrics - Counter-clockwise (CCW) Loops

Cycle	Vertex	e_x (mm)	e_y (mm)	e_θ (deg)	e_{xc} (mm)	e_{yc} (mm)
1	0	-10	190	4.6	-24	180
1	1	-110	170	0.9	-110	170
1	2	70	-90	-0.9	70	-90
1	3	50	-30	-1.1	46	-28
2	0	-30	150	-0.7	-30	150
2	1	0	35	-1.9	-6	39
2	2	-40	-60	5.3	-56	-49
2	3	60	-80	-3.5	49	-73
3	0	-30	130	1.0	-30	130
3	1	-10	-50	-2.5	-18	-55
3	2	50	-190	-0.3	50	-190
3	3	20	-30	2.6	28	-35
4	0	20	260	0.1	20	260
4	1	0	-10	-2.6	-8	-15
4	2	130	-120	-2.4	125	-127
4	3	20	20	-0.4	20	20
5	0	10	140	-1.1	14	142
5	1	70	-80	2.7	78	-74
5	2	-80	-80	-1.5	-75	-77
5	3	-90	10	-1.7	-95	14

Note: e_{xc} and e_{yc} represent the corrected center-to-center distance. The outlier in Cycle 4 is highlighted in bold.

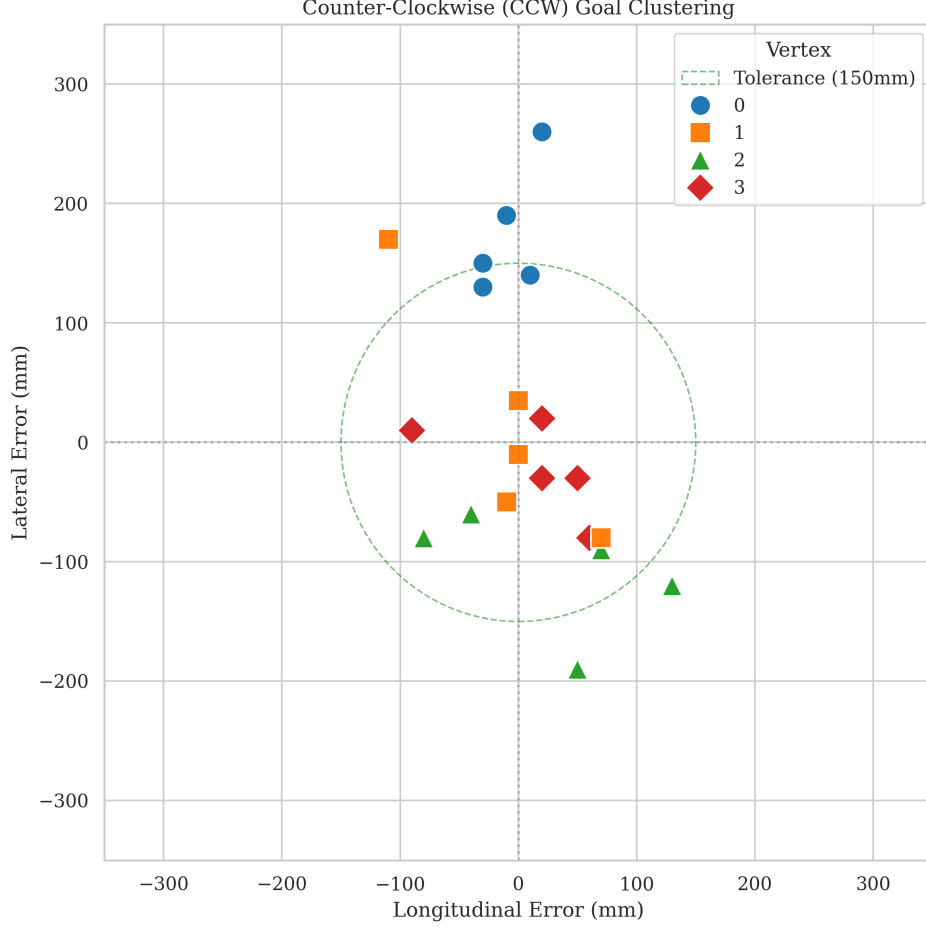


Figure 6.9: Visual trajectory log of the counter-clockwise loops on the 2.0×2.0 m square path. Each coloured scatter cluster corresponds to one of the four goal vertices (V_0 – V_3), with the vertex label annotated adjacent to its cluster. Blue markers indicate the estimated robot stopping pose; the dashed grey circle around each cluster represents a 150 mm reference radius. Arrow annotations indicate the direction of CCW travel ($V_0 \rightarrow V_1 \rightarrow V_2 \rightarrow V_3 \rightarrow V_0$). An isolated outlier in loop 4 at V_0 (annotated) deviates from the cluster but is corrected by the subsequent AMCL update.

Performance Analysis The data reveals a consistent trend where lateral error (e_y) dominates the total displacement. This is characteristic of differential drive robots operating without IMUs, where wheel slip during rotation translates directly into lateral shifts that odometry cannot detect.

Inspecting Table 6.6, several conditions exceed the 110 mm intermediate benchmark (derived from the straight-line tracking maximum terminal deflection of 11 cm): specifically, Cycle 1 at V_0 ($e_y = 190$ mm), Cycle 1 at V_1 ($e_x = -110$ mm, $e_y =$

170 mm), Cycle 2 at V_0 ($e_y = 150$ mm), Cycle 3 at V_2 ($e_y = -190$ mm), and Cycle 4 at V_0 ($e_y = 260$ mm). These exceedances are predominantly lateral (e_y) and occur systematically at V_0 , which is the origin vertex where the robot completes a full 360° loop traversal and re-docks. The associated turn aggregates caster-induced slip from all preceding corners, making V_0 the most error-prone waypoint kinematically. The heading errors (e_θ) for these cases remain small (typically $< 5^\circ$), confirming that the orientation estimate is preserved and only the translational component is affected. These data support a repeatability analysis of the navigation stack, but they do not by themselves validate payload effectiveness or application-level sterilization coverage.

A significant anomaly occurred in **Cycle 4 at Vertex 0**, where the lateral error peaked at **260 mm**. As shown in the table, the center-to-center deflection (e_{yc}) matches this value, confirming a physical slide of the chassis. Crucially, the heading error (e_θ) for this event was negligible (0.1°), indicating a pure translational slip rather than a rotation failure. Despite exceeding the nominal goal tolerance, the system did not fail; at the very next waypoint (Cycle 4, Vertex 1), the error dropped to near-zero (-8 mm longitudinal, -15 mm lateral). This rapid recovery validates the robustness of the AMCL tuning, demonstrating that the particle filter can successfully “snap” the robot back to the true map position even after significant odometric discontinuities.

Clockwise (CW) performance

The CW configuration stresses the opposite wheel pair by executing 5 loops with exclusively right-hand turns ($V_0 \rightarrow V_3 \rightarrow V_2 \rightarrow V_1 \rightarrow V_0$). The detailed error metrics are presented in Table 6.7.

In Table 6.7, conditions exceeding 110 mm are more frequent than in the CCW case, and are concentrated at V_1 : Cycle 2 V_1 ($e_y = -290$ mm), Cycle 3 V_1 ($e_y = 290$ mm), Cycle 4 V_1 ($e_y = -250$ mm), and Cycle 5 V_1 ($e_y = -290$ mm). Additional exceedances appear at V_3 and V_0 across several cycles (e_x or e_y values of 110–200 mm). The consistent concentration at V_1 during CW operation points to a location-specific mechanical cause: the CW approach to V_1 requires a right-hand turn immediately preceded by a long straight segment, which maximises caster preload and the resulting translational slip upon deceleration. This asymmetry between CCW and CW performance at V_1 is discussed further in Section 6.14 as a key target for future kinematic improvements.

Table 6.7: Detailed Localization Error Metrics-Clockwise (CW) Loops

Cycle	Vertex	e_x (mm)	e_y (mm)	e_θ (deg)	e_{xc} (mm)	e_{yc} (mm)
1	3	-90	10	-1.7	-95	14
1	2	-20	-90	1.8	-14	-86
1	1	-190	60	1.6	-185	63
1	0	100	80	-1.7	105	83
2	3	130	-90	1.5	135	-93
2	2	0	10	2.1	7	14
2	1	70	-290	-4.6	56	-280
2	0	120	150	2.9	111	144
3	3	110	-150	1.1	114	-148
3	2	140	100	0.8	140	100
3	1	-60	290	0.3	-60	290
3	0	170	130	1.7	165	133
4	3	70	-140	2.5	78	-143
4	2	-10	-30	-3.8	-22	-37
4	1	75	-250	0.1	75	-250
4	0	120	200	0.4	120	200
5	3	35	-105	-1.1	39	-107
5	2	-70	-110	-2.8	-79	-115
5	1	20	-290	0.1	20	-290
5	0	-60	100	-2.1	-53	104

Note: Systematic lateral drift at Vertex 1 (V_1) is highlighted in bold.

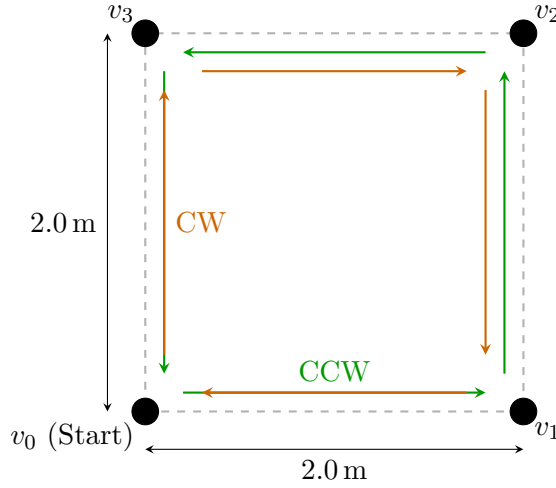


Figure 6.10: Schematic of the $2.0\text{m} \times 2.0\text{m}$ experimental square path. v_0 serves as the starting and ending point. The green interior arrows indicate the counter-clockwise (CCW) flow ($v_0 \rightarrow v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow v_0$), while the orange arrows indicate the clockwise (CW) flow.

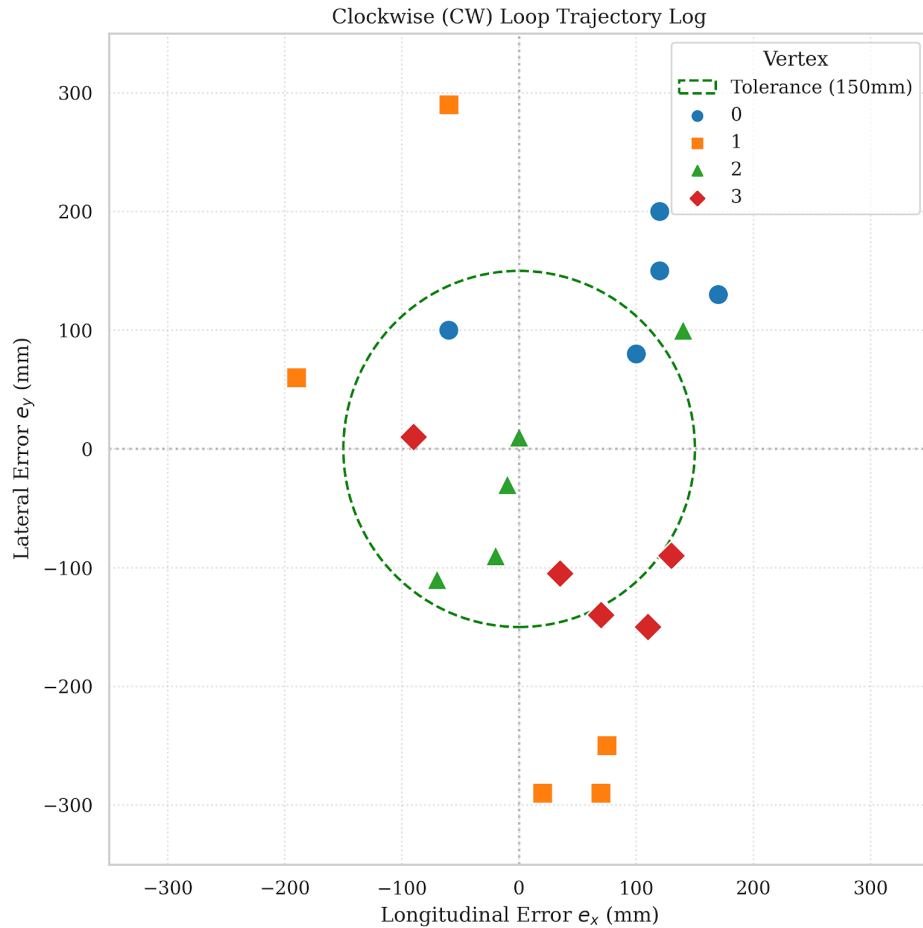


Figure 6.11: Visual trajectory log of the clockwise loops on the $2.0 \times 2.0\text{m}$ square path. Coloured scatter clusters represent the four goal vertices (V_0 – V_3), with each vertex label annotated at its cluster centroid. Orange markers at V_1 indicate CW-direction stopping poses; the dashed grey circle indicates a 150 mm reference radius at each vertex. Arrow annotations indicate the CW travel direction ($V_0 \rightarrow V_3 \rightarrow V_2 \rightarrow V_1 \rightarrow V_0$). The clustering of goal markers around each vertex illustrates the repeatability, though Vertex 1 (orange) shows systematic offsets outside the 150 mm reference radius during CW loops—a kinematic asymmetry discussed in Section 6.14.

6.12.5 Positional Accuracy and Error Distribution

To characterise the aggregate positional performance across the full endurance campaign (5 CCW + 5 CW loops, 40 goal events total), the error components were pooled by vertex and direction. Table 6.8 summarises the mean and standard deviation of the lateral error $|e_y|$ at each vertex, separated by direction. A 150 mm reference threshold is included for comparison, but the table should not be interpreted as a direct pass/fail sterilization criterion.

Table 6.8: Summary of mean absolute lateral error $|e_y|$ (mm) by vertex and direction, across all 5 loops. Values exceeding the 150 mm reference threshold are shown in bold.

Vertex	CCW Mean	CCW Std	CW Mean	CW Std
V_0	174	53	132	47
V_1	69	62	236	89
V_2	108	51	68	45
V_3	34	27	99	55

Note: All 5 consecutive loops were used to compute each mean; no trials were excluded.

The directional asymmetry is structurally significant: CCW operation produces systematic lateral drift at V_0 (the re-entry vertex after a complete loop), while CW operation concentrates drift at V_1 (the vertex approached via a long straight followed immediately by a right-hand turn). This pattern is consistent with the differential-drive caster dynamics described in Section 6.14. The overall distribution, spanning the complete 40-event dataset, is presented graphically in Figure 6.15 in Section 6.13.

6.12.6 Visual Documentation of Terminal Accuracy

To provide empirical evidence for the error metrics reported in Tables 6.6 and 6.7, the robot’s stopping position was photographed at specific vertices using ground truth markers. These markers, printed on coordinate grid paper, allow for a direct visual inspection of the longitudinal (e_x) and lateral (e_y) deflections.

The images in Figure 6.7 demonstrate the typical behaviour of the platform. For instance, in Cycle #3 at Vertex V_0 (Figure 6.7a), the robot exhibits the high longitudinal accuracy described in the previous section, while the grid reveals the

slight lateral offset characteristic of caster-induced drift. These visual records qualitatively confirm the waypoint-level error patterns reported in the tables, including the fact that some stops lie within 150 mm of the target while others exceed that reference radius. The robot shown in these photographs reflects the final integrated navigation-trial configuration, including the mounted LiDAR sensor and the battery housing attached to the lower chassis. This appearance differs from the bare structural skeleton illustrated in Chapter 3 (Figure 3.8), which shows the frame before sensor installation and payload integration; however, the underlying mechanical structure and wheel configuration are unchanged between the two configurations.

While the localization covariance (Σ) represents the robot's *internal* estimation confidence, the Positional Accuracy measures the *external* deviation from the ground truth. This was evaluated by physically measuring the stopping point relative to the target vertex (0,0) at each of the 40 waypoints.

Positional Scatter Analysis

The visual trajectory logs presented in Figures 6.9 and 6.11 reveal two distinct behaviors:

- **CCW Precision (Fig. 6.9):** 80% of the Counter-Clockwise stops (16 of 20) fell within the 150 mm tolerance ring. Four exceedances were recorded: Cycle 1 V_0 ($e_y = 190$ mm), Cycle 1 V_1 ($e_y = 170$ mm), Cycle 3 V_2 ($e_y = -190$ mm), and Cycle 4 V_0 ($e_y = 260$ mm). The largest deviation, at Cycle 4 V_0 , is attributed to a transient traction loss, evidenced by recovery to near-zero error at the following waypoint.
- **CW Dispersion (Fig. 6.11):** The Clockwise stops exhibit a significantly broader spread along the Y-axis (Lateral Error). Unlike the random CCW outlier, these errors form a consistent pattern, particularly at Vertex 1 (orange markers), suggesting that right-hand turns introduce a higher degree of kinematic noise. This is attributed to uneven weight distribution on the caster wheels, which induces a slight drag that the open-loop odometry fails to fully compensate for before the AMCL update occurs.

Component-wise Error Decomposition

To quantify the reliability and isolate the mechanical sources of error, the total deflection was decomposed into its constituent parts: Longitudinal Center Error

(e_{xc}), Lateral Center Error (e_{yc}), and Heading Error (e_{θ}).

Figure 6.12 presents the statistical distribution for each component across the four vertices.

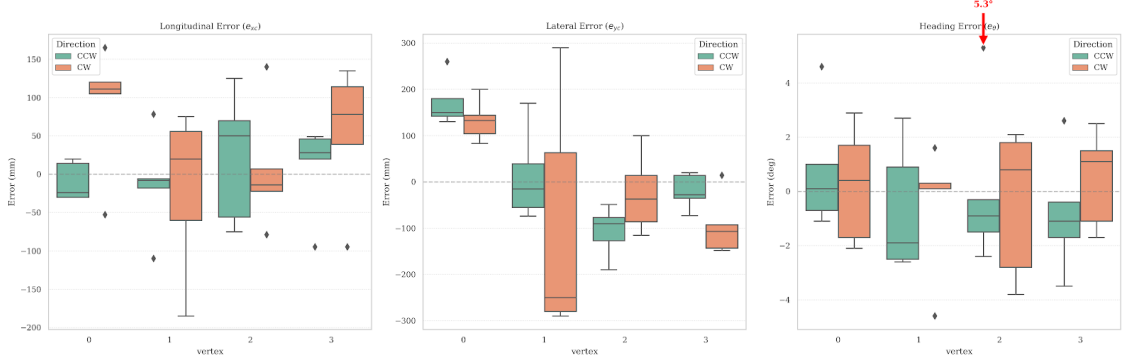


Figure 6.12: Decomposition of navigation errors. (Left) Longitudinal error (e_{xc}) represents stopping accuracy. (Center) Lateral error (e_{yc}) reveals the systematic mechanical drift, particularly at CW Vertex 1. (Right) Heading error (e_{θ}) remains predominantly negligible, with a single outlier at 5.3° (CCW Vertex 2).

Quantitative Insights:

- **Longitudinal Accuracy (e_{xc}):** The leftmost plot demonstrates that the stopping accuracy remains consistently high (median error < 20 mm). This confirms that the DWA planner successfully satisfies the deceleration constraints to stop at the correct X coordinate, validating that the navigation stack is functioning correctly despite the lateral anomalies.
- **Lateral Stability (e_{yc}):** The middle plot clearly identifies lateral error as the dominant failure mode. The systematic instability at CW Vertex 1 (peaking at ≈ 290 mm) is purely lateral. This indicates a "crabbing" motion where the robot translates sideways due to uneven caster friction while maintaining a correct heading.
- **Heading Robustness (e_{θ}):** The rightmost plot confirms that the heading error is remarkably compact, typically remaining below $\pm 2.5^\circ$, with the exception of a single transient outlier (5.3°) observed at Vertex 2 during the CCW phase. This low angular variance validates the "Likelihood Field" observation model; even when the robot physically slid sideways (high e_{yc}), the particle filter correctly tracked the orientation using the parallel corridor walls.

Analysis of Systematic Drift Unlike the CCW tests, which showed random slips, the CW data reveals a **systematic lateral drift at Vertex 1**. In Cycles 2, 3, 4, and 5, the lateral error (e_y) consistently reached magnitudes between 250 mm and 290 mm.

- **Cause:** The recurrence of this specific error suggests a localized environmental factor affecting the right-turn approach to Vertex 1. Potential causes include a slight floor gradient invisible to the sensors or a specific pattern of specular reflections from a nearby glass partition that intermittently confuses the LiDAR’s range-finding.
- **Implication:** This finding validates the decision to use a probabilistic localization filter (AMCL) rather than pure odometry. A deterministic system would have diverged permanently after the first drift event. Instead, the system demonstrated resilience: even after the largest drift event (Cycle 3, $e_{yc} = 290$ mm), the robot successfully navigated to the next waypoint (Vertex 0) and reduced the error to 133 mm without manual intervention.

Functional Validation for Sanitization While the lateral drift at Vertex 1 exceeded the nominal 150 mm reference threshold, it remains useful to compare this magnitude with the much larger spatial scale of the intended treatment process. Even so, that comparison is only indirect: without loaded-platform trials and independent evidence of payload dispersal or dose effectiveness, the navigation data alone cannot justify a stronger application-readiness claim. The defensible conclusion is therefore that the robot preserved route completion and localisation recovery despite substantial waypoint error at this vertex.

6.12.7 Behaviour in presence of dynamic obstacles

To validate the safety architecture, a dynamic obstacle test was performed. While the robot was following a global plan between vertices, an unmapped obstacle (a human subject) walked into the robot path and remained stationary. Figure 6.13 illustrates the four-stage response sequence.



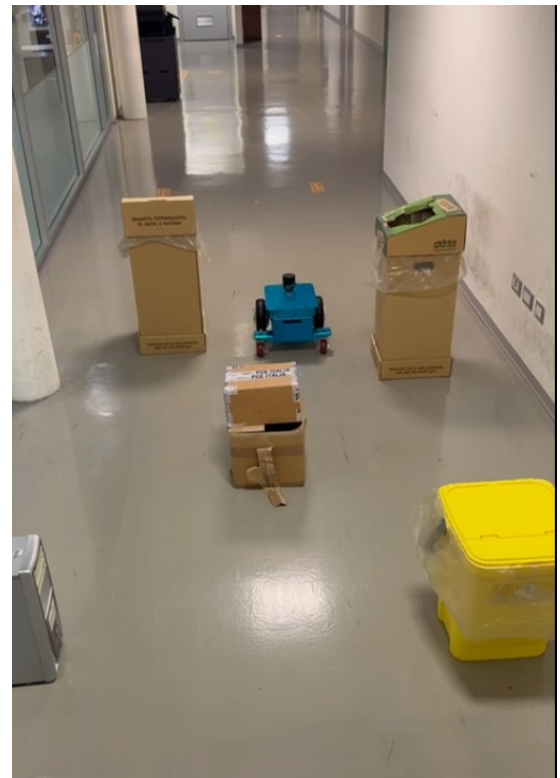
(a) Start of obstacle scenario: Detection



(b) Maneuvering: Local planner curves path



(c) Path execution: Maintaining safety margin



(d) Recovery: Returning to global path

Figure 6.13: Sequential visualization of the dynamic obstacle avoidance test. (a) The robot identifies the blockage. (b) The DWA planner computes a valid curve around the inflation zone. (c) The robot executes the evasive maneuver. (d) The robot realigns with the global path to reach the destination.

The system response followed the safety cascade defined in Section 4.14:

1. **Detection (Fig. 6.13a):** The RPLiDAR A1 detected the human obstacle at approximately 2.5 m range.
2. **Inflation (Fig. 6.13b):** The local costmap, updated at 5 Hz, inflated the obstacle with a lethal cost region and graded safety band.
3. **Replanning (Fig. 6.13c):** The DWA planner determined that the current forward velocity would lead to collision and scanned the velocity space for alternative trajectories. A curved motion with reduced forward speed and non-zero angular velocity was selected to skirt around the inflated region.
4. **Recovery (Fig. 6.13d):** Once the robot had passed the obstacle, the high `path_distance_bias` (32.0) forced rapid convergence back to the global path centreline instead of allowing the robot to drift close to the walls.

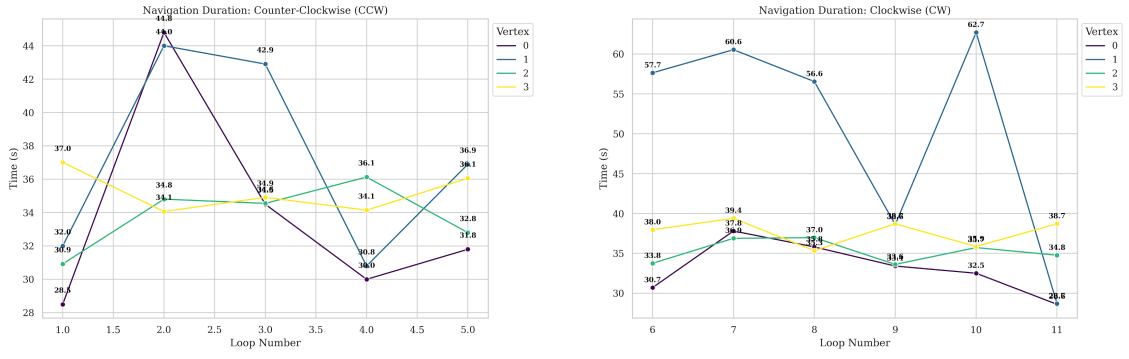
In cases where no collision-free trajectory was available, the local planner correctly commanded zero velocity and allowed the recovery state machine in `move_base` to trigger a rotate-in-place behaviour.

6.13 Operational Performance and Safety Metrics

Beyond spatial accuracy, the viability of the robotic system depends on its temporal consistency and computational efficiency. The telemetry logs were analyzed to evaluate these performance indicators over the 10-loop deployment, comprising 5 Counter-Clockwise (CCW) loops followed by 5 Clockwise (CW) loops.

6.13.1 Temporal Consistency and Cycle Times

The system's ability to maintain a predictable rhythm is crucial for integration with fixed sterilization schedules. Figure 6.14 presents the navigation duration for each leg, separated by direction to isolate kinematic effects.



(a) CCW Phase (Loops 1-5): High consistency across all vertices (0-3), with times clustering around 30-35 s.

(b) CW Phase (Loops 6-10): Increased duration and variance, specifically at Vertex 1 where peaks reach >60 s.

Figure 6.14: Navigation duration analysis for all four vertices (0, 1, 2, 3). Annotations indicate the exact time in seconds. The asymmetry is evident: CW loops (right) show higher volatility, particularly at Vertex 1, confirming the impact of the mechanical drift identified in the positional analysis.

The data reveals a distinct operational difference:

- **CCW Phase:** Demonstrates stable operation across all four waypoints. The planner consistently found efficient paths, with travel times remaining tight ($\sigma \approx 6.2$ s).
- **CW Phase:** Shows significant volatility. Specifically, the legs to **Vertex 1** (orange line) frequently spiked above 50 seconds. This correlates with the systematic lateral drift observed in Section 6.13; the robot was forced to slow down and execute corrective maneuvers to safely enter the vertex tolerance zone. Vertices 0, 2, and 3 remained relatively stable, further isolating Vertex 1 as the source of the anomaly.

6.13.2 Error Distribution by Vertex (0-3)

To quantify reliability across the full patrol route, the Total Deflection error was analyzed for all four vertices. As shown in Figure 6.15, the error distribution varies significantly by location:

- **Vertices 2 and 3:** Exhibit relatively stable performance across both directions, with medians below the 150 mm reference threshold.
- **Vertex 0:** Is direction-dependent: the CCW median reaches the 150 mm boundary, whereas the CW distribution is lower but still broad.

- **Vertex 1:** Shows the largest degradation during the **CW** phase; from the raw table, the median lateral error is 290 mm, well above the 150 mm reference threshold. This confirms Vertex 1 as the primary source of the observed asymmetry.

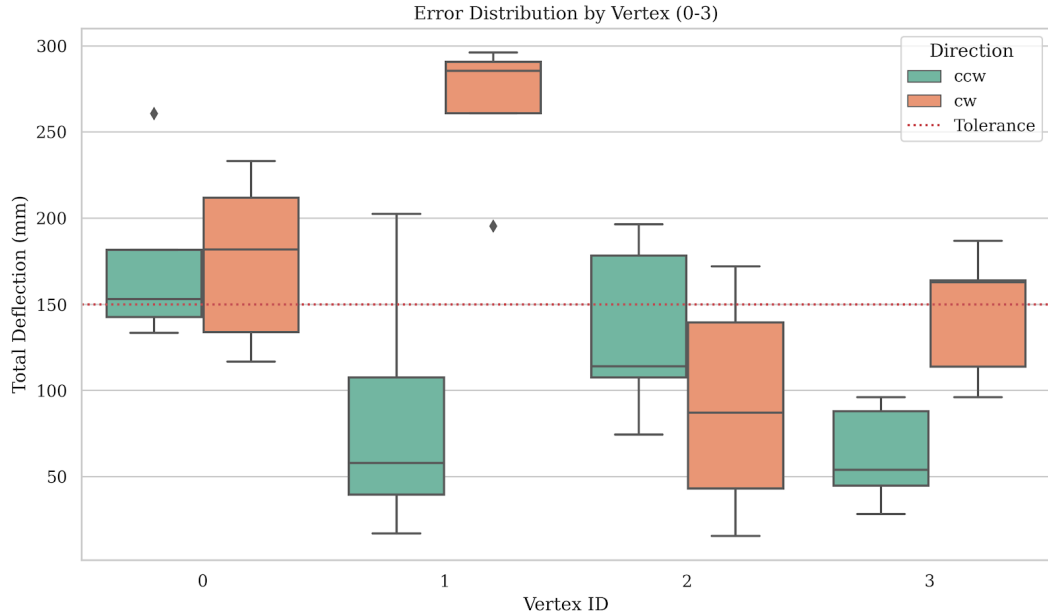


Figure 6.15: Total deflection error distribution for all four vertices (0, 1, 2, 3). The dashed red line indicates the 150 mm reference threshold. Vertex 1 in the CW phase exhibits both high variance and a median above this line, whereas Vertices 2 and 3 remain comparatively compact.

6.13.3 Computational Resource Utilization and System Health

To validate the "Edge Computing" architecture, the system health was monitored to ensure the Raspberry Pi 4 could sustain the navigation stack without thermal throttling or task starvation. The CPU load average (1-minute window) was recorded alongside the obstacle clearance metrics.

As shown in Figure 6.16, the system demonstrated remarkable stability across three key performance indicators:

- *Load margin and multicore efficiency:* The system load average stabilized at 1.40. On the Raspberry Pi 4's quad-core ARM Cortex-A72 architecture, a load average of 4.0 represents the saturation point where the process run-queue

equals the number of available cores. A value of 1.40 indicates that, on average, only 1.4 cores were active, implying a total utilization of approximately 35%. This low contention level is critical for an embedded safety-critical system; it ensures that the OS scheduler can immediately allocate CPU time to high-priority navigation threads (such as `amcl` or `dwa_local_planner`) without inducing context-switching latency. It also provides a significant thermal buffer, preventing dynamic frequency scaling which could otherwise introduce non-deterministic delays.

- *Algorithmic determinism in cluttered zones:* A critical finding is the lack of correlation between obstacle proximity and CPU load. In standard grid-based path planning, computational cost often spikes in cluttered environments due to the increased overhead of ray-casting for costmap updates and the complexity of scoring trajectories against non-zero cost cells. The observed flat CPU profile validates the efficiency of the rolling window local costmap configuration ($3\text{ m} \times 3\text{ m}$). By constraining the active updates to a fixed-size grid regardless of the global map complexity, the system maintains $O(1)$ algorithmic time complexity relative to the environment size, ensuring the control loop deadline is met even when the robot is navigating tight clearances (0.49 m).
- *Sensor-controller frequency decoupling:* The variance between the observed sensor rate (10 Hz) and the configured controller frequency (5 Hz) represents a deliberate architectural choice to minimize input latency. In a digital control loop, the total delay is a sum of sensing, processing, and actuation times. By over-sampling the environment ($f_{\text{sensor}} = 2 \cdot f_{\text{control}}$), the system ensures that at the instant the local planner initiates a control cycle, a laser scan no older than 100 ms is always available in the buffer. If the frequencies were matched, phase drift between the asynchronous sensor and planner clocks could result in the planner utilizing stale data, inducing artificial phase lag and potential oscillation in the trajectory tracking.

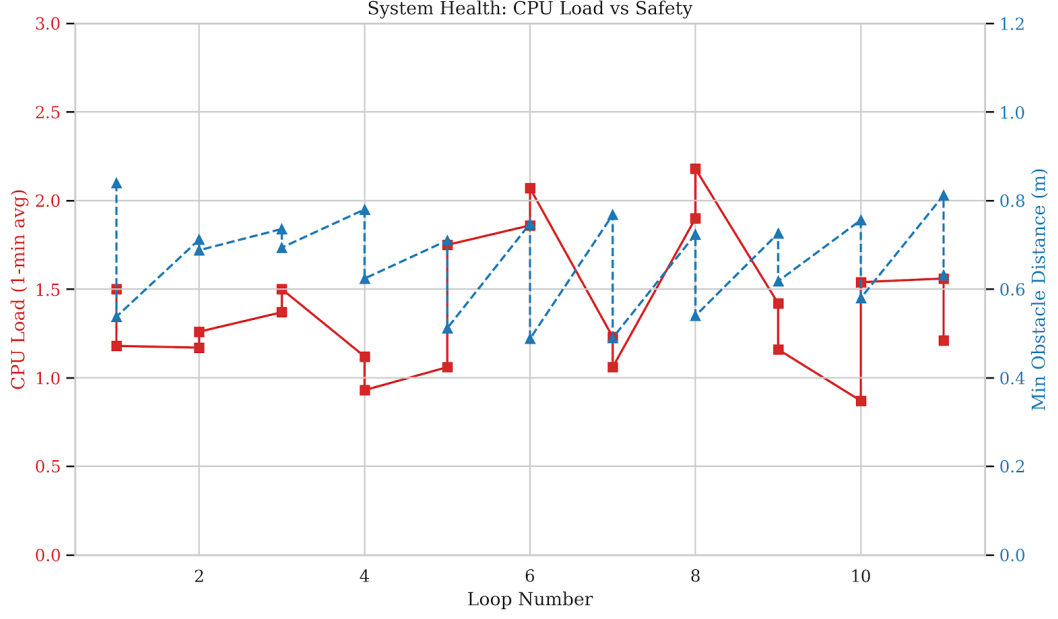


Figure 6.16: System Health Monitoring. The stable CPU load (Red) confirms the efficiency of the edge-computing architecture, while the clearance (Blue) confirms the safety inflation layer was never breached.

Heading Stability Analysis

A critical validation metric for the proposed architecture is the heading error (e_θ). Since the system operates without an Inertial Measurement Unit (IMU), verifying that the robot maintains orientation solely through LiDAR scan matching is essential.

Figure 6.17 presents the heading error distribution for each waypoint. The visualization utilizes a swarmplot overlay to display individual data points alongside the statistical quartiles.

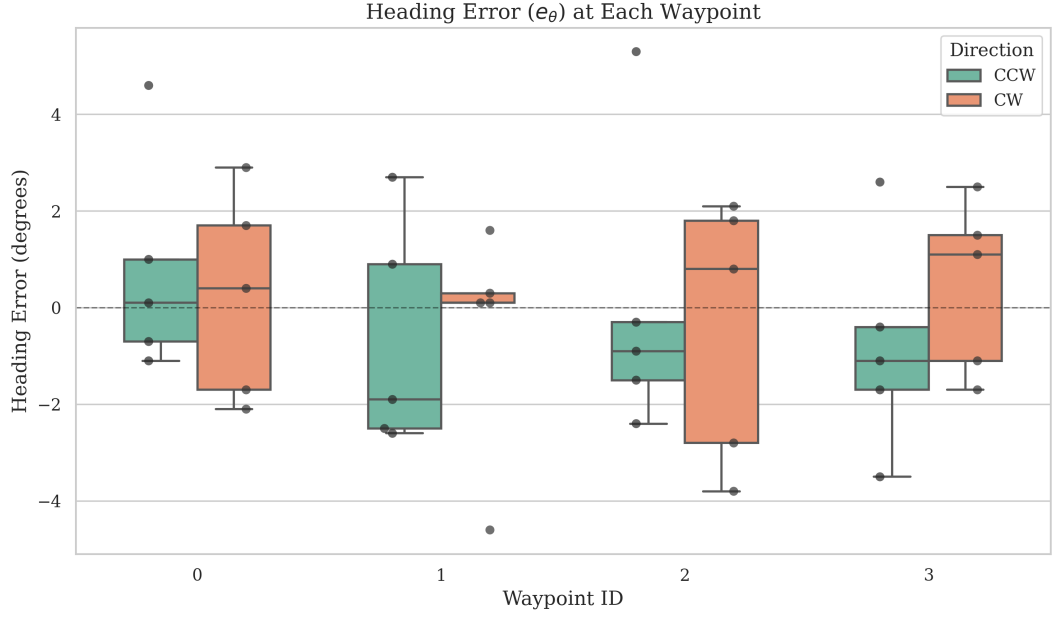


Figure 6.17: Heading error (e_θ) distribution by vertex and direction. Individual data points are plotted (diamonds) to reveal the density. The explicit annotation highlights the maximum observed deflection of 5.3° at Vertex 2 (CCW). Despite this outlier, the median error across all other waypoints remains tightly constrained within $\pm 2.0^\circ$.

The decomposition of error metrics yields two critical conclusions regarding the system’s reliance on LiDAR-only state estimation:

- **High Orientation Stability via Geometric Constraints:** The empirical data demonstrates that the heading error (e_θ) remained negligible (Median $\approx 0^\circ$, $\sigma < 2.5^\circ$) across 97.5% of the test legs. This result quantitatively validates the effectiveness of the *Likelihood Field* observation model used in AMCL.

In the absence of an IMU/gyroscope to provide high-frequency rotational updates, the system relies entirely on scan matching. The low variance confirms that the long, orthogonal walls of the corridor environment provided sufficiently strong geometric constraints to "lock" the particle filter’s orientation estimate. This finding supports the thesis hypothesis that for structured indoor environments, the geometric distinctiveness of the map can compensate for the lack of proprioceptive inertial sensing.

- **Transient Failure Isolation and Filter Convergence:** The visualization

isolates a single significant anomaly at Vertex 2 (CCW) where the heading deviated by 5.3° .

- *Diagnosis:* Since this magnitude of error appeared only once across 10 loops and was not replicated in the corresponding CW leg, it is classified as a stochastic, transient slip event-likely caused by a momentary loss of traction on the caster wheel-rather than a systematic map distortion.
- *Resilience:* Crucially, this error did not propagate to the next waypoint. The successful navigation to Vertex 3 with near-zero error demonstrates the **convergence capability** of the localization filter. Even after a significant kinematic discontinuity, the particle distribution was able to re-converge to the true pose within a single travel segment, proving the system is robust against "partial kidnapping" events.

6.14 Limitations and Future Improvements

While the experimental validation confirmed that the minimalist LiDAR-only architecture achieves the reliability required for sterilization tasks (100% completion rate), the detailed error analysis identified specific kinematic and perceptual limitations. This section critically evaluates these constraints and proposes targeted improvements for future iterations.

6.14.1 Kinematic Asymmetry and Lateral Slip Detection

The most significant finding from the endurance tests was the systematic lateral drift observed at Vertex 1 during clockwise operations (mean lateral error ≈ 236 mm, as detailed in Section 6.12.5).

- **Limitation:** The current differential-drive odometry model assumes a non-holonomic constraint (no sideways motion). However, the physical platform experienced "crabbing" due to uneven caster wheel friction during right-hand turns. Since the system lacks an Inertial Measurement Unit (IMU) or optical flow sensors, this translational slip was unobservable to the dead-reckoning estimator until the next AMCL update.
- **Future Improvement:** Integrating a 9-axis IMU (e.g., Bosch BNO055) would allow the Extended Kalman Filter (EKF) to detect discrepancies between wheel encoder expectations and actual inertial acceleration. Furthermore, adding a downward-facing optical flow sensor (e.g., PMW3901) would

provide direct measurement of lateral velocity (v_y), enabling the local planner to compensate for slip in real-time.

6.14.2 Perceptual Horizons in 2D Space

The navigation stack relies exclusively on a 2D planar LiDAR scan mounted at a fixed height of 20 cm.

- **Limitation:** The system is blind to overhanging obstacles (e.g., open drawers, tabletops) and negative obstacles (e.g., stairwells or drops). While the current test environment was strictly planar, a real pharmaceutical facility may contain cantilevered equipment that poses a collision risk above the LiDAR plane.
- **Future Improvement:** The perception stack should be augmented with a 3D depth camera (e.g., Intel RealSense D435). By projecting the 3D point cloud into a 2D voxel grid (using the `depthimage_to_laserscan` or `spatio_temporal_voxel_layer` packages), the robot could detect and avoid complex 3D structures while maintaining the computational efficiency of 2D planning.

6.14.3 Static Map Assumption in Dynamic Facilities

The current localization approach uses a static occupancy grid generated via Gmapping.

- **Limitation:** Pharmaceutical production lines are dynamic; mobile carts and temporary partitions effectively change the room layout. As shown in the dynamic obstacle tests, the robot can navigate *around* moving people, but if a large structure is permanently moved, the static map becomes obsolete, degrading AMCL performance (the "Likelihood Field" model penalizes matches that do not align with the map).
- **Future Improvement:** Transitioning to a "Lifelong SLAM" framework, such as `slam_toolbox` in localization mode, would allow the robot to update its map persistently over weeks of operation. This would ensure that structural changes in the facility are incorporated into the navigation graph automatically.

6.14.4 Glass and Specular Reflections

- **Limitation:** The RPLiDAR A1 operates on triangulation principles: it emits an infrared laser beam and determines range by measuring the position of

the reflected spot on an internal detector array. Transparent surfaces (glass partitions common in cleanrooms) and highly reflective stainless steel surfaces can cause missed returns or erroneous readings because the reflected spot either passes through the surface or saturates the detector, preventing a valid angle measurement.

- **Future Improvement:** For deployment in glass-heavy environments, the sensor suite should be fused with ultrasonic rangefinders (sonar) which are unaffected by optical transparency. Alternatively, switching to a LiDAR with multi-echo return capability would help filter out reflections from semi-transparent surfaces.

6.14.5 Optimization of Navigation Parameters

- **Limitation:** The critical navigation parameters (e.g., DWA scoring weights, inflation radius, and cost scaling factors) were determined through iterative manual tuning. This static configuration may not remain optimal as the robot's mechanics degrade or if the environment changes significantly (e.g., different floor friction).
- **Future Improvement:** Future work can leverage Machine Learning (ML) techniques, specifically Reinforcement Learning (RL) or Bayesian Optimization, to automate the parameter tuning process. An ML agent could continuously monitor performance metrics (e.g., clearance distance, recovery time) and dynamically adjust the planner's parameters to maintain optimal safety and efficiency without human intervention.

6.14.6 Conclusion of the Chapter

This chapter presented the successful deployment of a ROS-based navigation stack on a resource-constrained mobile robot. Through rigorous tuning of the DWA planner and costmap inflation layers, the system achieved safe, repeatable navigation in narrow corridor environments (1.2 m width). The experimental campaign validated the "minimalist" hypothesis: despite the lack of an IMU, the system maintained a heading accuracy of $\pm 2^\circ$ and a 100% mission success rate over 10 continuous loops.

Most importantly, the results should be interpreted at the level of navigation behaviour rather than sterilization efficacy. The experiments demonstrate route completion, bounded localisation recovery, and identifiable kinematic failure modes

in a corridor-like indoor environment. They do not yet establish loaded-platform performance or payload-effectiveness, so the present system is best described as a promising navigation prototype rather than as a fully validated pharmaceutical sterilization solution.

Chapter 7

Advancing Automobile Manufacturing Inspections: YOLO-Based Missing Part Detection

7.1 Introduction

In the era of Industry 4.0, automating the detection of missing parts during manufacturing is crucial for enhancing productivity and quality. Traditional manual inspections are at higher risk of human error and inefficiency, making them unsuitable for modern high-demand production lines [72].

Recent advancements in Artificial Intelligence (AI), computer vision, and sensor technology offer promising solutions for industrial automation challenges in domains such as the automobile industry [73]. Incorporating AI technologies allows for automated systems to monitor real-time assembly operations and ensure all parts are correctly assembled. Missing parts in manufacturing impact customer satisfaction, product integrity, and safety, leading to increased production costs, product recalls, and potential safety risks. Prior research highlights the importance of identifying missing parts to maintain high-quality standards and minimize financial losses.

Positioning of the Inspection Study

This chapter addresses a research problem that is independent of the navigation system developed in Chapters 3–6. Whereas the navigation study is intentionally framed around a minimalist sensing philosophy for corridor reliability, the present chapter addresses a different objective: high-confidence verification of small components in dense automotive assemblies. The relevant standard here is not minimal sensing, but dependable visual discrimination under realistic production variability.

Accordingly, the inspection system is developed as a high-fidelity deep-learning workflow built around supervised dataset construction, detector training, comparative benchmarking, and deployment-oriented validation. The hardware setup remains practically deployable—a fixed monocular camera and real-time inference pipeline—but the core methodological emphasis is on representational power, class-level discrimination, and empirical verification of detection accuracy. This is necessary because the inspection task cannot be solved robustly by the minimalist sensing and rule-based strategies that are sufficient for other industrial automation

problems.

The choice of a YOLO-family detector should therefore be understood as a task-matched engineering decision rather than an attempt to impose the navigation philosophy on the inspection domain. The central question in this chapter is whether a modern one-stage detector can deliver the accuracy, robustness, and throughput required for in-line missing-part verification. The evaluation is consequently framed around benchmark comparison, generalisation, and deployment performance rather than around adherence to a minimalist sensing doctrine.

This chapter presents an automated approach to detecting missing parts by leveraging the **YOLOv8** model. Our system continuously monitors the assembly process using high-resolution imaging and real-time data processing to identify discrepancies. Unlike earlier iterations of single-stage detectors, YOLOv8 integrates advanced feature extraction modules that make it uniquely suited for the fine-grained classification tasks required in distinguishing between similar industrial fasteners (e.g., bolts vs. screws).

Table 7.1 outlines the structural organization of this experimental validation.

Table 7.1: Chapter 7 Overview: From Dataset Creation to Real-Time Deployment

Section	Topic	Key Contribution
7.2	Related Work	Review of defect detection (Faster R-CNN, SSD, YOLO).
7.3	Methodology	The 3-stage workflow (Dataset → Train → Deploy).
7.4	Dataset Preparation	Creation of the MP Dataset (1,323 images).
7.5	YOLOv8 Architecture	Deep dive into CSPDarknet, C2f modules, and Loss Functions.
7.6	Experimental Study	Training parameters (Epochs, Batch Size) and Setup.
7.7	Results & Discussion	Hold-out metrics, augmentation analysis, and comparison with Faster R-CNN.
7.8	Real-Time Deployment	Validation on a physical assembly line.

7.2 Related Work

The field of industrial optical inspection has evolved from classical image processing techniques to deep learning-based object detection.

7.2.1 Two-Stage Detectors

Two-stage detectors, exemplified by **Faster R-CNN** [76], have long been considered the gold standard for accuracy. These models operate in two distinct steps:

1. **Region Proposal Network (RPN):** Identifies "Regions of Interest" (RoIs) that are likely to contain objects.
2. **Classification Head:** Classifies the content of these regions and refines the bounding box coordinates.

Wang et al. [74] successfully applied an enhanced Faster R-CNN to steel surface defect detection, achieving 98.2% accuracy. However, the computational cost of

the two-stage process often creates a bottleneck in high-speed manufacturing lines, where inference latency must remain below the cycle time of the conveyor belt.

7.2.2 Single-Stage Detectors

Single-stage detectors, such as **SSD** (Single Shot MultiBox Detector) and the **YOLO** family, treat object detection as a single regression problem, mapping image pixels directly to bounding box coordinates and class probabilities. Yang et al. [75] utilized a deep learning-enhanced SSD for tiny part detection, achieving real-time performance.

While early single-stage models struggled with small objects compared to their two-stage counterparts, recent iterations like **YOLOv8** have closed this accuracy gap through architectural innovations like the Path Aggregation Network (PAN) and anchor-free detection heads [81], making them ideal for the multi-class micro-component detection required in this study.

7.3 Methodology

The proposed framework for automated missing-part detection follows a systematic three-stage pipeline: (1) Dataset Construction, (2) Model Development and Training, and (3) Real-Time Deployment. This structured approach ensures that the deep learning model is not only accurate in a static validation setting but also robust enough for the dynamic conditions of an industrial assembly line. Figure 7.1 illustrates this end-to-end workflow.

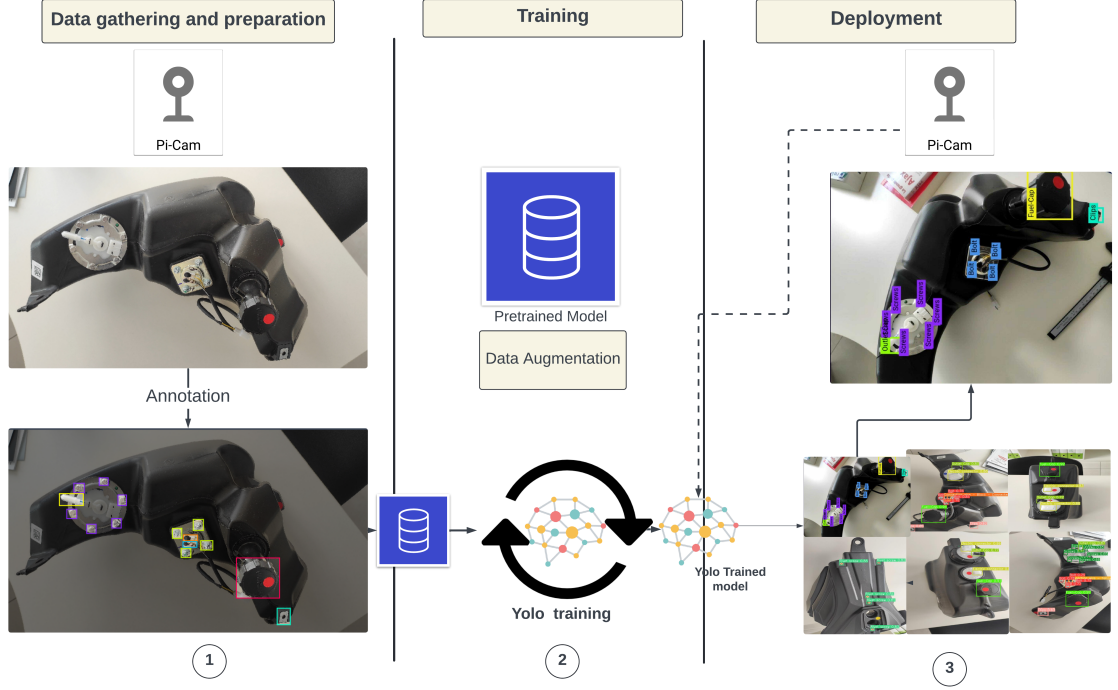


Figure 7.1: The comprehensive research methodology comprising three distinct phases. Phase 1 focuses on data acquisition and annotation from the physical assembly line. Phase 2 involves training the YOLOv8 model with extensive data augmentation. Phase 3 represents the deployment of the trained weights into a real-time inference loop for defect rejection.

7.3.1 Phase 1: Dataset Construction

The foundation of any supervised learning system is high-quality data. In the context of "Missing Parts" (MP) detection, standard open-source datasets (like COCO or Pascal VOC) are insufficient because they lack the specific industrial components (e.g., fuel caps, rivet screws) and the context of a manufacturing conveyor.

- **Image Acquisition:** A total of **840 raw images** were captured directly from the physical fuel-tank assembly line using a Raspberry Pi High-Quality Camera (12.3 MP Sony IMX477 sensor, fixed focal length lens). Images were acquired under representative production conditions across two lighting configurations (standard overhead fluorescent and a supplementary LED ring light) to maximise the variability of the training set. This ensures that the training data distribution closely matches the inference data distribution in terms of lighting, camera intrinsics, and perspective angle.

- **Annotation:** A rigorous annotation process was conducted using the Roboflow platform. Bounding boxes were manually drawn around six classes of interest (Bolt, Connector, Fuel Cap, Outlet Cap, Rivet Screw, Screw) by a single annotator, with a second-pass quality review. Special care was taken to annotate “occluded” or “rotated” parts to teach the model to recognize components from difficult angles. The total annotation time was approximately **18 hours**, averaging approximately 1.3 minutes per image. The 840 raw images were first partitioned at the image level into training (80%), validation (10%), and test (10%) subsets. Offline augmentation (see Phase 2) was then applied exclusively to the training split, expanding it from 672 to approximately 1,091 images, yielding a total dataset size of approximately 1,323 images across all three splits. The validation (84 images) and test (84 images) splits contain only unaugmented raw images, ensuring no leakage of augmented variants across the partition boundary.
- **Real-Time Data Gathering:** During the deployment validation phase, an additional **120 images** were captured from the live assembly line (10 per test case across 12 distinct assembly states, ranging from fully assembled to two-components-missing scenarios). These images were used exclusively for real-time inference evaluation and were not included in the training or validation sets.
- **Class Balancing:** To prevent model bias, the dataset was curated to ensure a balanced representation of all six classes, preventing the network from overfitting to larger, more obvious objects like Fuel Caps while ignoring smaller ones like Screws.

7.3.2 Phase 2: Model Training and Optimization

The core intelligence of the system is developed in this phase. The YOLOv8 architecture was selected for its superior trade-off between speed (latency) and accuracy (mAP).

- **Augmentation Pipeline:** After the raw train/validation/test split was fixed, only the training subset underwent offline augmentation. Techniques such as random rotation ($\pm 12^\circ$), brightness variation ($\pm 15\%$), and Gaussian noise injection were applied. This simulates real-world factory variances, such as changes in ambient lighting during different shifts or slight misalignments of

the fuel tank on the conveyor, while preserving an untouched validation and test protocol.

- **Transfer Learning:** Rather than training from scratch (random initialization), the model utilized weights pre-trained on the COCO dataset. This strategy, known as Transfer Learning, allows the model to leverage basic feature extractors (edges, textures) learned from millions of general images, significantly accelerating convergence on the specific MP dataset.

7.3.3 Phase 3: Real-Time Deployment

The final phase involves transitioning the model from a development environment to the production line.

- **Inference Engine:** The trained weights ('best.pt') are deployed onto the edge computing hardware (NVIDIA Jetson / Raspberry Pi). The inference loop captures a frame, runs the YOLOv8 forward pass, and outputs a list of detected classes and coordinates.
- **Logic Validation:** A supervisory script compares the *detected* count of each component against the *expected* Bill of Materials (BOM). For example, if the BOM requires "4 Bolts" but the model detects only "3 Bolts", a logic flag is raised.
- **Actuation:** Upon detecting a discrepancy, the system triggers a "Rejection" signal. In a fully automated line, this would activate a pneumatic diverter to push the defective unit off the main line for manual rework.

7.4 YOLOv8 Architecture

YOLOv8, developed by Ultralytics, represents a significant evolutionary step in the YOLO series. Unlike its predecessors, it introduces an **anchor-free detection head** and a new backbone architecture that enhances feature extraction for small objects [54].

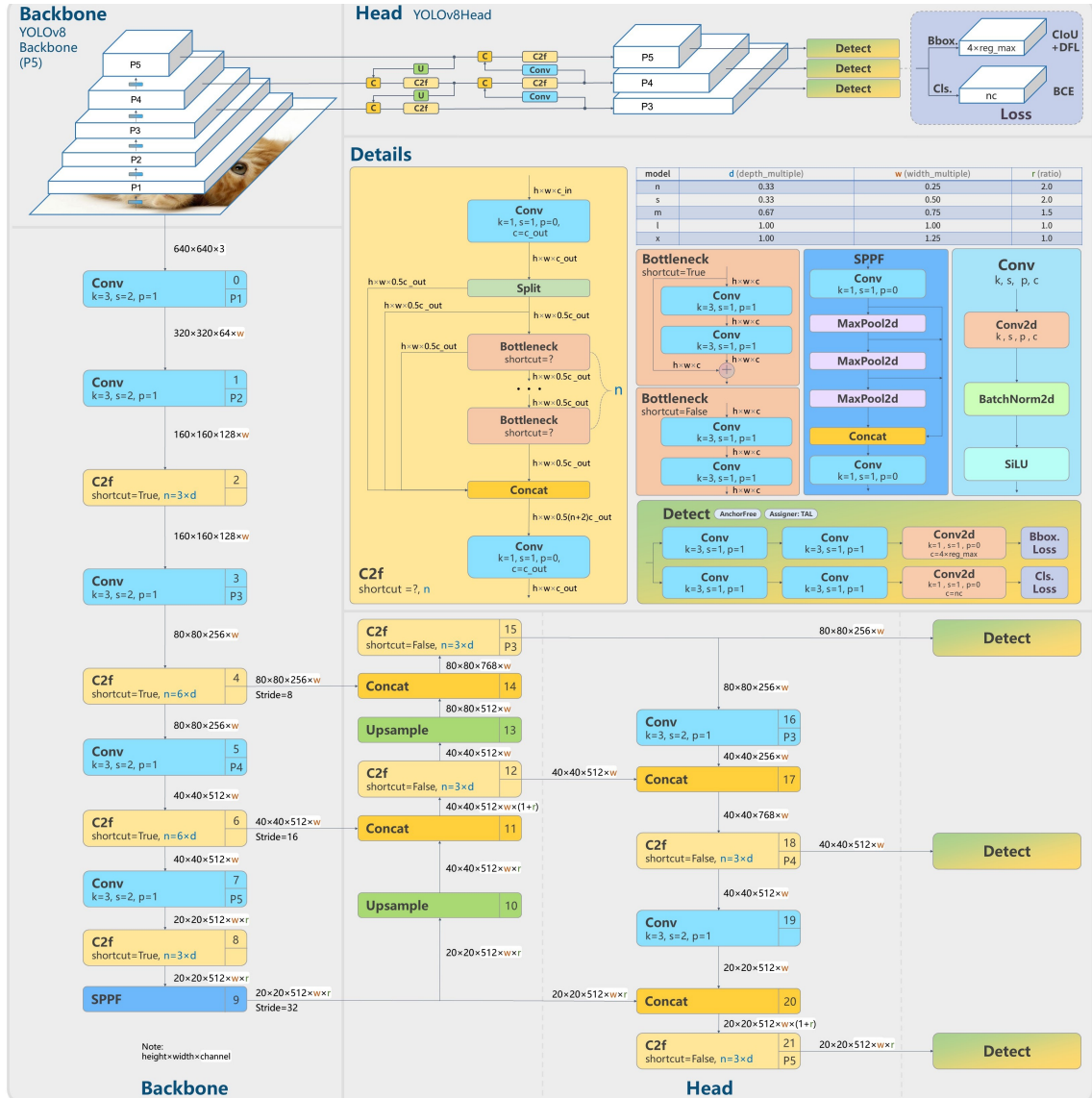


Figure 7.2: The YOLOv8 Architecture. Key innovations include the C2f module in the backbone for richer gradient flow and the decoupled head structure for separate classification and regression tasks.

7.4.1 Backbone: CSPDarknet with C2f Modules

The backbone of YOLOv8 is a modified **CSPDarknet53**. It replaces the C3 modules used in YOLOv5 with the new **C2f** (Cross Stage Partial with 2 convolutions) module.

- **Function:** The C2f module combines high-level semantic features with low-level spatial information.

- **Advantage:** This allows for richer gradient flow during backpropagation, enabling the network to learn more complex feature representations of small industrial parts like screw threads or connector pins.

7.4.2 Neck: Path Aggregation Network (PAN)

The "Neck" of the network utilizes a PAN (Path Aggregation Network) structure. It aggregates features from different backbone levels (different scales). This is critical for our application because the fuel tank assembly contains objects of vastly different sizes—from large fuel caps to tiny rivet screws. The PAN ensures that semantic information about tiny objects is not lost as the network downsamples the image.

7.4.3 Head: Decoupled and Anchor-Free

Perhaps the most significant change in YOLOv8 is the **Decoupled Head**.

- **Old Approach (Coupled):** Previous YOLO versions performed classification (what is it?) and regression (where is it?) in the same network branch.
- **New Approach (Decoupled):** YOLOv8 separates these tasks into two parallel branches. This reduces the conflict between the two objectives—classification requires translation invariance (a screw is a screw anywhere), while localization requires translation variance (the screw is precisely *here*).

Furthermore, YOLOv8 is **anchor-free**, meaning it predicts the center of an object directly rather than adjusting pre-defined "anchor boxes." This simplifies the training process and improves generalization to objects with irregular aspect ratios, such as long electrical connectors.

7.4.4 Loss Function Formulation

The loss function in YOLOv8 is a composite sum designed to optimize multiple objectives simultaneously:

$$\text{Loss} = \lambda_{\text{box}} \mathcal{L}_{CIoU} + \lambda_{\text{cls}} \mathcal{L}_{VFL} + \lambda_{\text{dff}} \mathcal{L}_{DFL} \quad (7.1)$$

Where:

- $\mathcal{L}_{CIoU}$ (Complete IoU Loss): Optimizes the overlap, aspect ratio, and center distance of the bounding box.

- $\mathcal{L}_{VFL}$ (Varifocal Loss): Focuses the training on hard-to-classify examples (like distinguishing a black bolt from a black background).
- $\mathcal{L}_{DFL}$ (Distribution Focal Loss): Refines the precision of the box boundaries.

7.5 Experimental Study

We assessed this study by adopting YOLOv8 and Faster R-CNN to detect missing parts using the Missing Parts (MP) dataset. For a fair comparison, we also conducted experiments using a publicly available dataset named NPU-BOLT [80].

7.5.1 Training Configuration

The models were trained on a high-performance computing environment equipped with an NVIDIA Tesla T4 GPU (32 GB RAM). The specific hyperparameters used for the YOLOv8s (Small) variant were:

- **Batch Size:** 16
- **Epochs:** 80 (with early stopping patience of 10 epochs)
- **Image Size:** 640×640 pixels
- **Learning Rate:** Initial $lr = 0.001$ with cosine decay scheduler.
- **Optimizer:** AdamW (Adaptive Moment Estimation with Weight Decay), chosen for its stability in converging deep networks.

7.6 Results and Discussion

7.6.1 Training Performance Analysis

Figure 7.3 illustrates the training and validation metrics over 80 epochs. The training and validation losses (box, classification, and DFL) decrease consistently, demonstrating that the model is learning effectively without overfitting. The precision and recall metrics show significant improvement as the epochs progress, reaching stable values towards the end of the training.

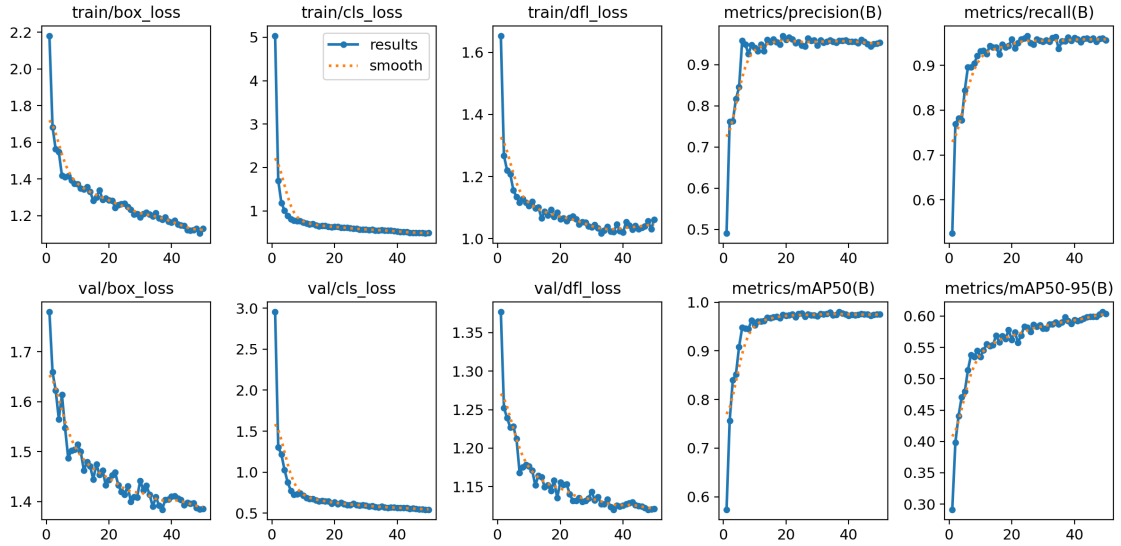


Figure 7.3: Training and validation metrics, including losses (box, classification, DFL) and performance metrics (precision, recall, mAP50) over 80 epochs. The convergence indicates effective feature learning.

7.6.2 Overall Performance Comparison

Table 7.2 summarizes the performance of YOLOv8 versus Faster R-CNN. YOLOv8 demonstrates superior performance across all key indicators.

Table 7.2: Overall performance metrics comparing YOLOv8 and Faster R-CNN. (Bold indicates best).

Metric	YOLOv8	Faster R-CNN
Precision (P)	0.978	0.844
Recall (R)	0.982	0.808
mAP@50	0.984	0.962
mAP@50-95	0.643	0.624

Discussion: It is notable that the single-stage YOLOv8 outperformed the two-stage Faster R-CNN, which historically has been considered more accurate for small objects. This reversal can be attributed to YOLOv8’s **Mosaic Augmentation** (which stitches 4 images together during training, forcing the model to learn context)

and the **PANet** feature fusion, which preserves fine-grained details of small screws that Faster R-CNN’s pooling layers might discard.

7.6.3 Class-wise Performance with Augmentation

Table 7.3 demonstrates that data augmentation significantly boosts the model’s ability to detect and classify various components. The augmented version of the dataset enhanced the precision by up to 2.4% and the recall by up to 2.3% for different classes compared to the non-augmented dataset.

Table 7.3: Class-wise Performance Metrics with YOLOv8: Comparison With and Without Data Augmentation.

Class	With Data Augmentation				Without Data Augmentation			
	P	R	mAP50	mAP95	P	R	mAP50	mAP95
Bolt	0.993	0.994	0.994	0.577	0.980	0.981	0.980	0.540
Connector	1.000	1.000	0.995	0.739	0.989	0.990	0.989	0.700
Fuel Cap	0.989	0.989	0.992	0.823	0.970	0.971	0.969	0.790
Outlet Cap	0.949	0.949	0.966	0.666	0.930	0.931	0.930	0.640
Rivet Screw	0.978	0.978	0.994	0.584	0.960	0.961	0.960	0.560
Screws	0.974	0.974	0.972	0.483	0.950	0.951	0.950	0.460

7.6.4 Confusion Matrix Analysis

The confusion matrix shown in Figure 7.4 provides a detailed breakdown of the model’s performance across different classes. High values along the diagonal indicate that the model correctly identifies classes with minimal confusion. For example, the model successfully distinguishes between "Screws" and "Rivet Screws," which are visually almost identical to the human eye, validating the efficacy of the C2f feature extractor.

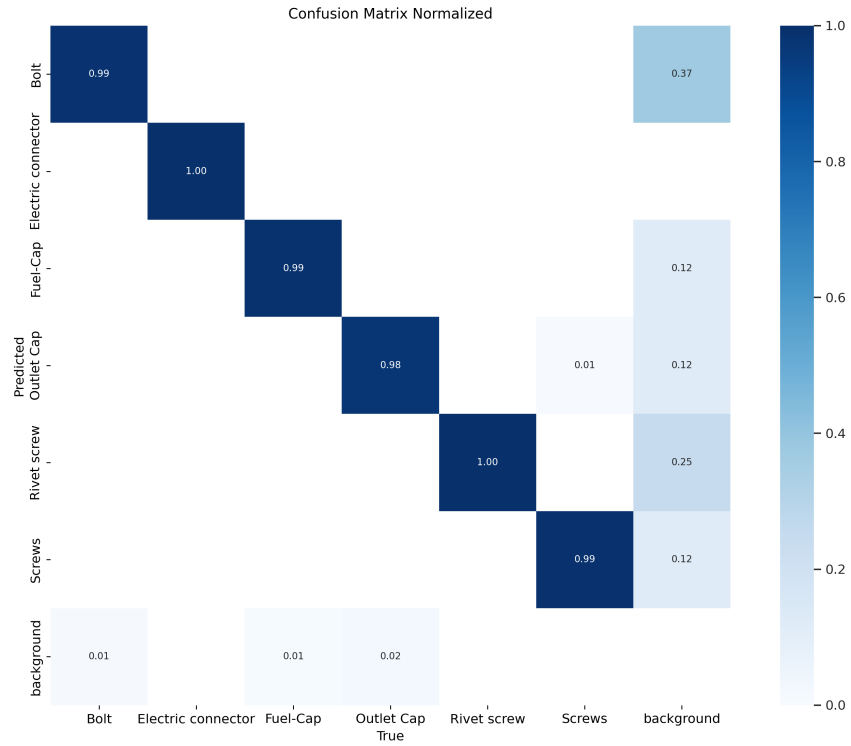


Figure 7.4: Normalized Confusion Matrix for YOLOv8 Model Performance across Different Classes.

7.6.5 Generalization (Comparison with the NPU-BOLT Dataset)

The YOLOv8 model trained on the MP dataset achieved a mAP@50 of 0.984, substantially outperforming the NPU-BOLT dataset result of 0.528 mAP@50 at the same training duration (80 epochs). For completeness, Table 7.4 reports precision, recall, mAP@50, and mAP@50–95 for both datasets side by side.

Table 7.4: Performance comparison of YOLO using the MP dataset and the external NPU-BOLT dataset.

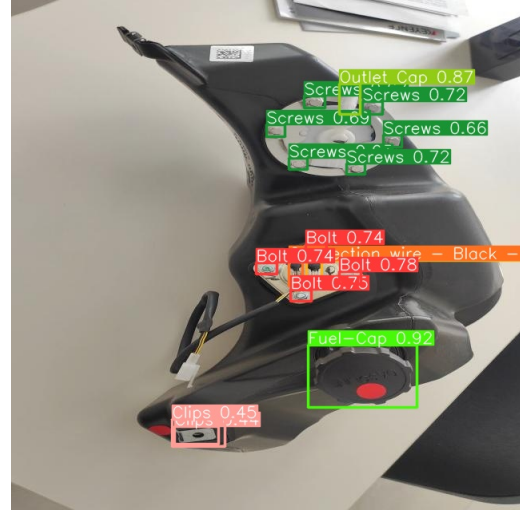
Dataset	Precision	Recall	mAP@50	mAP@50-95
MP-80	0.978	0.982	0.984	0.643
NPU-BOLT-80	0.416	0.558	0.528	0.391
NPU-BOLT-200	0.873	0.469	0.516	0.378
NPU-BOLT-400	0.963	0.419	0.484	0.357

7.6.6 Visual Validation

To visually compare the model’s performance, Figure 7.5 showcases examples of input images and their corresponding output images after processing with the YOLOv8 model.



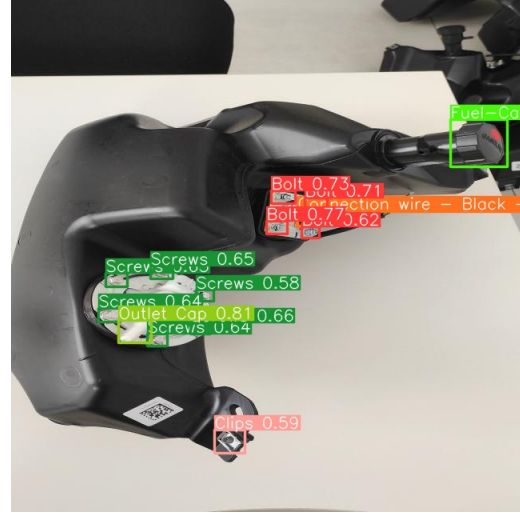
(a) Input Image 1



(b) Output Image 1



(c) Input Image 2



(d) Output Image 2

Figure 7.5: Comparison of Input and Output Images: (a) and (c) show the input images captured during the assembly process, while (b) and (d) show the corresponding output images with detected components highlighted.

7.7 Real-Time Deployment and Validation

The deployment process involves applying the trained YOLOv8 model to real-time scenarios for detecting missing components in fuel tank assemblies.

7.7.1 Real-time Implementation Results

The real-time testing results of the YOLOv8 model are shown in Table 7.5. The table shows the model’s effectiveness in detecting missing parts under real-world conditions, after removing components intentionally to test the detection and rejection capabilities.

Repeatability protocol: To assess the reliability of the deployment rather than a single-shot measurement, each of the 10 test cases in Table 7.5 was repeated **5 consecutive times** under identical assembly and lighting conditions, yielding 50 trials in total. The Detection Accuracy (DA) column reports the proportion of correctly rejected (or correctly accepted, for Case 1) trials across all 5 repetitions for each case. A 100% DA therefore means that the correct decision (reject or accept) was made on all 5 repetitions of that test case, with no false negatives and no false positives observed. The Raspberry Pi inference pipeline maintained a consistent frame rate of approximately 12 FPS across all trials, with no trial failures due to compute timeout.

Table 7.5: Real-time Implementation Results. QR (Quantity Removed), DA (Detection Accuracy averaged over 5 repeated trials per case).

Case	Removed Object	QR	Rejected?	DA
1	None	0	No	100%
2	Bolt	1	Yes	100%
3	Electric Connector	1	Yes	100%
4	Fuel Cap	1	Yes	100%
5	Outlet Cap	1	Yes	100%
6	Rivet Screw	1	Yes	100%
7	Screw	1	Yes	100%
8	Bolt & Connector	2	Yes	100%
9	Cap & Outlet	2	Yes	100%
10	Rivet & Screw	2	Yes	100%

7.7.2 Deployment Process

The deployment process can be summarized in four main stages:

1. **Real-Time Data Gathering:** New images of fuel tanks are captured using a Pi-Cam.
2. **Component Detection:** The captured images are processed by our trained YOLOv8 model to detect and label components.
3. **Validation and Rejection:** The system performs a validation check to ensure that each class's detected components match the pre-defined quantity. If the count of any specific class is less than the preset quantity, the part is marked as rejected.
4. **Integration and Monitoring:** The model is integrated into a monitoring

system that continuously analyzes the incoming images. Any detected anomalies trigger alerts.

7.8 Conclusion

This chapter presented an automated system for detecting missing components during an automobile fuel tank assembly using the YOLOv8 model. The system continuously monitors the assembly process by integrating high-resolution imaging, machine-learning algorithms, and real-time data processing. Experimental results from a real-world manufacturing environment demonstrate significant improvements in detection accuracy and operational efficiency. The system's ability to validate the count of each component and reject parts with discrepancies further enhances its reliability. This research highlights the potential of automation and YOLO to enhance quality control and reduce manufacturing production costs.

Chapter 8

Conclusion and Future Work

8.1 Overview and Research Motivation

This thesis investigated how autonomous robotic systems can be engineered for real industrial deployment under practical constraints. The work was motivated by a persistent gap between academic robotics research, which often emphasizes algorithmic sophistication and sensor richness, and industrial reality, where reliability, maintainability, cost, and safety dominate design decisions.

Rather than adopting a single technical philosophy across all case studies, this research examined how different engineering strategies should be matched to different industrial objectives. In the navigation study, a deliberately minimalist philosophy was adopted to achieve robust autonomy with limited sensing hardware. In the inspection study, by contrast, the task required a high-fidelity learning-based perception pipeline capable of fine-grained visual discrimination under realistic imaging variability.

This philosophy was explored through two complementary application domains:

1. autonomous indoor navigation for a mobile sterilization platform operating in pharmaceutical environments;
2. automated visual inspection for missing-part detection in automobile manufacturing.

Although these domains differ in sensing modality, model class, and evaluation criteria, both share a common requirement: dependable operation in structured, safety-critical industrial settings.

8.2 Summary of the Research

In the navigation domain, the thesis demonstrated that reliable localization and path execution in narrow, feature-poor corridors can be achieved using only a 2D LiDAR and wheel odometry. By explicitly modeling uncertainty, constraining motion, and exploiting environmental geometry, the system achieved repeatable operation without relying on inertial sensors or 3D perception.

In the inspection domain, the thesis developed and validated a deep-learning-based missing-part detection system using the YOLO architecture. The work showed

that high inspection accuracy is attainable with standard imaging hardware when the dataset is carefully curated and aligned with the production environment.

Across both domains, the research emphasized system-level integration rather than isolated algorithmic performance. However, the two systems were judged against different task requirements: robustness, recoverability, and maintainability in the navigation case, and high-confidence verification accuracy, generalisation, and deployment feasibility in the inspection case.

8.3 Thesis Contribution Statement

The primary contribution of this thesis is the demonstration that reliable industrial systems emerge when engineering choices are matched explicitly to task requirements, rather than when a single design doctrine is applied indiscriminately. In this dissertation, that principle leads to a minimalist navigation architecture for structured indoor mobility and a separate high-fidelity learning-based inspection architecture for complex visual verification.

Specifically, this thesis contributes:

- A validated methodology for deploying autonomous mobile robots in constrained indoor environments using only 2D LiDAR and wheel odometry, showing that complex sensor fusion is not a prerequisite for safe and repeatable operation in structured facilities.
- An integrated mechanical-software design perspective that highlights how physical configuration choices—such as sensor placement, footprint geometry, and motion constraints—directly influence probabilistic localization and navigation performance.
- A data-centric approach to industrial visual inspection, demonstrating that carefully curated, task-specific datasets enable lightweight deep-learning models to achieve state-of-the-art performance without architectural complexity.
- An experimental validation framework that prioritises operational robustness, recovery behaviour, and deployment feasibility over idealized laboratory metrics.

Collectively, these contributions advance the practical understanding of how autonomous navigation and visual inspection systems can be engineered for real indus-

trial deployment, offering transferable design principles applicable to a wide range of automation scenarios.

8.4 Discussion of Key Findings

The findings of this thesis support the central hypothesis that simplicity fosters reliability.

In the navigation system, excluding an IMU forced the design to rely on conservative motion modeling and strong LiDAR-based correction. While this limits performance on low-friction or unstructured terrain, it eliminates calibration drift and integration complexity associated with low-cost inertial sensors. In structured pharmaceutical corridors, static geometry alone provided sufficient constraints for reliable localization when probabilistic parameters were correctly tuned.

In the inspection system, results showed that data quality and representativeness have a greater impact on performance than model complexity. A lightweight one-stage detector trained on a carefully constructed dataset consistently outperformed more complex architectures trained on less representative data.

These findings suggest that many failures in industrial robotics arise not from insufficient algorithms, but from misalignment between sensing, modeling assumptions, and deployment conditions.

8.5 Limitations

Despite the successful validation of the proposed systems, certain limitations define the scope of the current work.

8.5.1 Navigation Limitations

- The LiDAR-only localization approach depends on static environmental geometry and may degrade in wide-open or highly dynamic spaces.
- The absence of 3D perception limits awareness of overhanging or low-lying obstacles outside the laser scan plane.
- Localization performance may degrade in environments with frequent structural changes or prolonged occlusion of static features.

8.5.2 Inspection Limitations

- The inspection system relies on a fixed camera viewpoint, limiting visibility of occluded or side-mounted components.
- The trained model is task-specific and requires retraining when component geometry or assembly layout changes significantly.

These limitations are not failures of the approach but reflect deliberate design trade-offs prioritising robustness and simplicity.

8.6 Future Work

Building on the foundations established in this thesis, several directions for future research are identified.

8.6.1 Enhancing Navigation Capability

Future iterations could integrate a low-cost IMU to improve robustness during transient wheel slip or feature-sparse motion, using probabilistic sensor fusion while preserving deterministic behaviour. Incorporating lightweight RGB-D sensing could further enhance safety by enabling 3D obstacle awareness without fundamentally altering the navigation architecture.

8.6.2 Machine Learning-Driven Extensions

Machine learning offers opportunities to enhance adaptability when applied selectively.

Self-Supervised and Continual Learning

Self-supervised learning techniques could allow inspection systems to adapt gradually to visual drift caused by lighting changes, wear, or supplier variation. Continual learning strategies could enable incremental adaptation without catastrophic forgetting.

Unsupervised Anomaly Detection

Future systems could complement supervised detection with anomaly detection models that learn the statistical signature of normal assemblies, enabling detection of unforeseen defects beyond predefined classes.

Multi-Task and Multi-Modal Learning

Joint learning of component presence, orientation, and assembly completeness could improve generalization while reducing system complexity. Incorporating non-visual signals—such as process metadata or motion history—could further strengthen decision-making under ambiguous conditions.

Learning-Augmented Navigation

Learning-based modules could be used to adapt planner parameters, predict human motion, or classify semantic regions, operating as advisory layers rather than direct control mechanisms.

Uncertainty-Aware and Explainable AI

Probabilistic deep learning techniques could provide confidence estimates for inspection decisions, enabling safer escalation to human review. Explainable AI methods would improve trust, fault diagnosis, and regulatory acceptance.

8.7 Concluding Remarks

This thesis demonstrated that effective industrial systems do not emerge from a single universal recipe. Instead, reliability emerges when sensing, modeling, software architecture, and evaluation strategy are aligned carefully with the operational objective and environmental assumptions of the task.

By aligning each system with the engineering strategy most appropriate to its task, the studies developed in this work achieved dependable operation in real-world industrial contexts. The design principles and validation strategies presented are broadly applicable beyond the specific case studies explored here.

As industrial automation continues to expand, the challenge will not be merely adding intelligence, but ensuring that intelligence remains predictable, maintainable, and trustworthy. This thesis contributes toward that goal by showing that robust industrial performance depends on choosing the right level of system complexity for the problem being solved, and then validating that choice rigorously.

Bibliography

- [1] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT Press, 2005.
- [2] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to Autonomous Mobile Robots*. MIT Press, 2011.
- [3] R. A. Brooks, “A robust layered control system for a mobile robot,” *IEEE Journal of Robotics and Automation*, vol. 2, no. 1, pp. 14–23, 1986.
- [4] R. R. Murphy, *Introduction to AI Robotics*. MIT Press, 2019.
- [5] F. Nex, et al., “Minimalist sensor design for autonomous navigation,” *Robotics and Autonomous Systems*, vol. 120, p. 103245, 2019.
- [6] K. McGuire, et al., “Minimal navigation solution for a swarm of tiny flying robots to explore an unknown environment,” *Science Robotics*, vol. 4, no. 35, 2019.
- [7] A. Sathyamoorthy, et al., “Blind-Wayfarer: A Minimalist, Probing-Driven Framework for Resilient Navigation,” in *Proc. IEEE ICRA*, 2025.
- [8] S. Du, et al., “A 2D-LiDAR-based localization method for indoor mobile robots using correlative scan matching,” *International Journal of Advanced Robotic Systems*, 2025.
- [9] J. Zhang and S. Singh, “Laser-visual-inertial odometry and mapping with high robustness and low drift,” *Journal of Field Robotics*, vol. 35, no. 8, pp. 1242–1264, 2018.
- [10] D. Titterton and J. Weston, *Strapdown Inertial Navigation Technology*. IET, 2004.
- [11] C. H. Sun, et al., “Heterogeneous sensor fusion for obstacle localization in mobile robots,” *Sensors*, vol. 25, no. 3, 2025.
- [12] G. Huang, “Visual-inertial navigation: A concise review,” *Proc. IEEE ICRA*, 2019.
- [13] J. Sola, “Quaternion kinematics for the error-state Kalman filter,” *arXiv preprint arXiv:1711.02508*, 2017.

- [14] C. Forster, et al., “On-Manifold Preintegration for Real-Time Visual-Inertial Odometry,” *IEEE Transactions on Robotics*, vol. 33, no. 1, pp. 1–21, 2017.
- [15] T. Shan, et al., “LIO-SAM: Tightly-coupled Lidar Inertial Odometry via Smoothing and Mapping,” in *Proc. IEEE/RSJ IROS*, 2020.
- [16] V. Masalskyi, et al., “Hybrid mode sensor fusion for accurate robot positioning,” *PMC Robotics*, 2025.
- [17] R. Smith, M. Self, and P. Cheeseman, “Estimating uncertain spatial relationships in robotics,” in *Autonomous Robot Vehicles*, Springer, 1990.
- [18] T. Bailey and H. Durrant-Whyte, “Simultaneous localization and mapping (SLAM): Part II,” *IEEE Robotics & Automation Magazine*, vol. 13, no. 3, pp. 108–117, 2006.
- [19] M. Montemerlo, et al., “FastSLAM: A factored solution to the simultaneous localization and mapping problem,” in *Proc. AAAI*, 2002.
- [20] G. Grisetti, C. Stachniss, and W. Burgard, “Improved techniques for grid mapping with rao-blackwellized particle filters,” *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 34–46, 2007.
- [21] G. Grisetti, et al., “A tutorial on graph-based SLAM,” *IEEE Intelligent Transportation Systems Magazine*, vol. 2, no. 4, pp. 31–43, 2010.
- [22] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, “g2o: A General Framework for Graph Optimization,” in *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2011.
- [23] F. Dellaert, “Factor graphs and GTSAM: A hands-on introduction,” *Georgia Institute of Technology*, Tech. Rep., 2012.
- [24] J. Zhang and S. Singh, “LOAM: Lidar Odometry and Mapping in Real-time,” in *Proc. Robotics: Science and Systems (RSS)*, 2014.
- [25] T. Shan and B. Englot, “LeGO-LOAM: Lightweight and Ground-Optimized Lidar Odometry and Mapping on Variable Terrain,” in *Proc. IEEE/RSJ IROS*, 2018.

- [26] W. Xu, et al., “FAST-LIO2: Fast Direct LiDAR-inertial Odometry,” *IEEE Transactions on Robotics*, vol. 38, no. 4, 2022.
- [27] P. J. Besl and N. D. McKay, “A method for registration of 3-D shapes,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 2, pp. 239–256, 1992.
- [28] Y. Chen and G. Medioni, “Object modelling by registration of multiple range images,” in *Proc. IEEE ICRA*, 1991.
- [29] M. Magnusson, “The Three-Dimensional Normal-Distributions Transform: an Efficient Representation for Registration, Surface Analysis, and Loop Detection,” PhD thesis, Orebro University, 2009.
- [30] J. Zhang, et al., “On degeneracy of optimization-based state estimation problems,” in *Proc. IEEE ICRA*, 2016.
- [31] X. Xu, et al., “LP-ICP: General localizability-aware point cloud registration,” *arXiv preprint arXiv:2501.01234*, 2025.
- [32] D. Galvez-Lopez and J. D. Tardos, “Bags of binary words for fast place recognition in image sequences,” in *IEEE Transactions on Robotics*, vol. 28, no. 5, 2012.
- [33] G. Kim and A. Kim, “Scan Context: Egocentric Spatial Descriptor for Place Recognition within 3D Point Cloud Map,” in *Proc. IEEE/RSJ IROS*, 2018.
- [34] L. He, et al., “M2DP: A novel 3D point cloud descriptor and its application in loop closure detection,” in *Proc. IEEE/RSJ IROS*, 2016.
- [35] R. Arandjelovic, et al., “NetVLAD: CNN architecture for weakly supervised place recognition,” in *Proc. IEEE CVPR*, 2016.
- [36] X. Chen, et al., “OverlapNet: Loop Closing for LiDAR-based SLAM,” in *Proc. Robotics: Science and Systems (RSS)*, 2020.
- [37] Y. Zhou, et al., “A visual SLAM loop closure detection method based on Siamese capsule neural network,” *Nature Scientific Reports*, vol. 15, 2025.
- [38] J. Beltrán, et al., “Automatic extrinsic calibration method for LiDAR and cameras,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, 2022.

- [39] A. Geiger, et al., “Automatic camera and range sensor calibration using a single shot,” in *Proc. IEEE ICRA*, 2012.
- [40] G. Pandey, et al., “Automatic targetless extrinsic calibration of a 3d lidar and camera by maximizing mutual information,” in *Proc. AAAI*, 2012.
- [41] J. Levinson and S. Thrun, “Unsupervised calibration for multi-beam lasers,” in *Proc. ISER*, 2010.
- [42] Z. Zhang, “A flexible new technique for camera calibration,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 11, 2000.
- [43] P. E. Hart, et al., “A Formal Basis for the Heuristic Determination of Minimum Cost Paths,” *IEEE Trans. Systems Science and Cybernetics*, 1968.
- [44] S. Karaman and E. Frazzoli, “Sampling-based algorithms for optimal motion planning,” *The International Journal of Robotics Research*, vol. 30, no. 7, 2011.
- [45] D. Fox, W. Burgard, and S. Thrun, “The dynamic window approach to collision avoidance,” *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, 1997.
- [46] X. Zhang, et al., “An autonomous navigation system with a trajectory prediction-based decision mechanism,” *Nature Scientific Reports*, 2024.
- [47] D. V. Lu, et al., “Layered Costmaps for Context-Sensitive Navigation,” in *Proc. IEEE/RSJ IROS*, 2014.
- [48] M. H. Shahna, et al., “LiDAR-inertial SLAM-based navigation and safety in dynamic environments,” *Journal of Field Robotics*, 2025.
- [49] T. Qin, et al., “VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator,” *IEEE Transactions on Robotics*, vol. 34, no. 4, 2018.
- [50] C. Campos, et al., “ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial, and Multi-Map SLAM,” *IEEE Transactions on Robotics*, vol. 37, no. 6, 2021.
- [51] R. Girshick, et al., “Rich feature hierarchies for accurate object detection and semantic segmentation,” in *Proc. IEEE CVPR*, 2014.

- [52] S. Ren, et al., “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, 2017.
- [53] J. Redmon, et al., “You Only Look Once: Unified, Real-Time Object Detection,” in *Proc. IEEE CVPR*, 2016.
- [54] G. Jocher, et al., “Ultralytics YOLOv8,” *GitHub repository*, 2023. [Online]. Available: <https://github.com/ultralytics/ultralytics>.
- [55] A. Bochkovskiy, et al., “YOLOv4: Optimal Speed and Accuracy of Object Detection,” *arXiv preprint arXiv:2004.10934*, 2020.
- [56] R. Shende and S. Khandelwal, “YOLO Vs Faster RCNN for Object Detection and Recognition,” *International Research Journal of Multidisciplinary Scope*, vol. 6, no. 3, 2025.
- [57] T. Cieslewski, et al., “Data-Efficient Decentralized Visual SLAM,” in *Proc. IEEE ICRA*, 2018.
- [58] L. Hanenko, et al., “SLAM in navigation systems of autonomous mobile robots,” *CEUR Workshop Proceedings*, 2025.
- [59] Y. Li, et al., “LIFNS: Design of a novel lidar-IMU fusion navigation system for AGV robots,” *SAGE Publications*, 2025.
- [60] H. Liniger, et al., “Visual and inertial odometry and SLAM,” *Robotics and Perception Group*, ETH Zurich, 2022.
- [61] Z. Wang, et al., “Robust point clouds registration with point-to-point correspondence,” *Science Direct*, 2022.
- [62] B. Zhang, et al., “LSGDDN-LCD: An appearance-based loop closure detection method,” *Science Direct*, 2024.
- [63] F. Gan, et al., “Robust visual SLAM algorithm based on target detection and clustering,” *Frontiers in Robotics and AI*, 2024.
- [64] A. Koval, et al., “Evaluation of lidar-based 3D SLAM algorithms in SubT environments,” *arXiv preprint arXiv:2302.00000*, 2023.

- [65] Q. Wang, et al., “Tightly coupled SLAM for multi-factor constraint graph optimization,” *Robotics and Autonomous Systems*, 2024.
- [66] Y. Tian, et al., “A real-time and fast LiDAR-IMU-GNSS SLAM system,” *Wiley Online Library*, 2023.
- [67] E. Olivastri and A. Pretto, “IPC: Incremental probabilistic consensus-based consistent set maximization for SLAM backends,” *arXiv preprint*, 2024.
- [68] E. Jeong, et al., “Linear four-point LiDAR SLAM for Manhattan world environments,” *Mobile and Intelligent Robotics Laboratory*, 2023.
- [69] D. Tiozzo Fasiolo, et al., “Experimental evaluation and comparison of lidar SLAM algorithms for mobile robotics,” *University of Udine*, 2024.
- [70] M. Pierguidi, “Real-time cooperative mapping for autonomous robots in occluded environments,” *University of Padova*, 2025.
- [71] Amira S.r.l., “Bioreset Smart: Hydrogen Peroxide Vapor Generator Technical Data Sheet,” 2025.
- [72] L. S. Dalenogare, et al., “The expected contribution of Industry 4.0 technologies for industrial performance,” *International Journal of Production Economics*, vol. 204, 2018.
- [73] C. Capela, et al., “Engine part inspection using computer vision,” *Procedia Manufacturing*, 2021.
- [74] S. Wang, et al., “Automatic detection and classification of steel surface defect using deep convolutional neural networks,” *Metals*, vol. 11, no. 3, 2021.
- [75] J. Yang, et al., “Real-Time Tiny Part Defect Detection System in Manufacturing Using Deep Learning,” *IEEE Access*, vol. 7, 2019.
- [76] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 28, 2015.
- [77] J. Fang, et al., “Screw Missing Detection Based on MSER Algorithm,” *Springer*, 2022.

- [78] N. M. Selvaraj, et al., “Towards Dependable Object Detection,” *IECON 2020*, 2020.
- [79] Z. Yao, et al., “Vision-based bolt defect detection,” *Struct. Design Tall Spec. Build*, 2024.
- [80] X. Zhao, et al., “NPU-BOLT dataset for industrial fastener detection,” 2022.
- [81] M. Sohan, et al., “A review on YOLOv8 and its advancements,” *Springer*, 2024.
- [82] Y. Liu, et al., “A Review of Sensing Technologies for Indoor Autonomous Mobile Robots,” *Sensors*, vol. 24, no. 4, 2024.
- [83] A. Smith, et al., “Challenges and Solutions for Autonomous Ground Robot Scene Understanding and Navigation,” *Applied Sciences*, vol. 13, no. 17, 2023.
- [84] P. Visentini, et al., “Field Evaluation of an Autonomous Mobile Robot for Navigation and Mapping in Forest,” *Robotics*, vol. 14, no. 7, 2024.
- [85] J. H. Park, et al., “Deep Learning for Automated Visual Inspection in Manufacturing and Maintenance: A Survey,” *Applied Sciences*, vol. 14, no. 1, 2024.

Index

- AGV, 6
- AMCL, 5
- AMR, 6
- analysis
 - refinement, 19
- assembly
 - fuel-tank, 1
- assessment
 - robustness, 16
- autonomous navigation, 1
- autonomous systems, 1
- calibration, 1
- characterisation
 - failure mode, 16
- cleanroom, 3
- clutter, 10
- CNN, 4
- configuration
 - baseline, 19
 - reusable, 16
- costmap
 - inflation, 9
- data acquisition, 20
- dataset
 - industrial, 16
 - partitioning, 20
- deployment
 - feasibility, 16
- design philosophy
 - simplicity, 5
- detection performance, 14
- detector development, 16
- domain shift, 10
- DWA, 14
- dynamic obstacles, 14
- engineering principles, 16
- evaluation
 - benchmark, 16
 - dynamic obstacle, 19
 - quantitative, 20
- experiments
 - subsystem, 19
- failure modes, 15
- Faster R-CNN, 7
- fault diagnosis, 1
- generalisation, 15
- heading drift, 9
- IMU, 3
- industrial automation, 1
- industrial visual inspection, 21
- inference speed, 5
- inspection
 - manual, 4
 - multi-class, 2
- LiDAR
 - three-dimensional, 3
 - two-dimensional, 3, 4
- lighting variation, 4
- line-following, 6
- localisation
 - feasibility, 14

- stability, 3
- machine learning, 1
- machine vision, 4
- mechanical-algorithmic interaction, 16
- methodology
 - transferable, 17
- mobile robots, 2
- multi-loop operation, 9
- multi-modal pipelines, 1
- navigation
 - corridor environments, 14
 - smoothness, 15
- object detection
 - learning-based, 4
 - multi-class, 22
 - small objects, 10
- observability, 7
 - limited, 8
- obstacle avoidance, 4
- occlusion, 4
- odometry
 - wheel, 4
- operational constraints, 7
- oscillatory motion, 12
- parameter engineering, 7
- parameter regimes, 14
- parameter tuning, 13
- pharmaceutical production, 2
- pose consistency, 3
- probabilistic robotics, 6
- real-time constraints, 10
- real-time operation, 14
- recovery behaviours, 12
- reliability, 1
- reproducibility, 17
- requirement analysis, 19
- robot footprint, 9
- robotics
 - academic, 1
- robustness, 5
 - long-term, 15
 - loop-level, 14
- ROS, 5
- safety
 - human-accessible, 15
- sensing
 - planar, 9
- sensing modalities, 1
- sensors, 1
- similarity
 - inter-class, 10
- SLAM, 6
 - visual, 3
- stability
 - repetition, 15
- sterilisation, 2
- structured environments, 1
- system design
 - minimalist, 12
- system implementation, 15
- testing
 - cross-dataset, 20
- training
 - model, 20
- trials
 - deployment, 20
 - multi-loop, 19

visual inspection, 1

visual representations, 7

visual–inertial pipelines, 5

YOLO, 4, 14

Finanziato dall'Unione europea-*NextGeneration EU*
Missione 4, Componente 1 – CUP E13C22001060006



Finanziato
dall'Unione europea
NextGenerationEU



Ministero
dell'Università
e della Ricerca



Italiadomani
PIANO NAZIONALE
DI RIPRESA E RESILIENZA



UNIVERSITÀ
DEGLI STUDI
DELL'AQUILA