



FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

AI-Enhanced Indoor Localization System for Autonomous Mobile Robots

João Afonso Braun Neto

Supervisor: Paulo Costa, Ph.D., Faculty of Engineering of University of Porto

Co-Supervisor: José Lima, Ph.D., Polytechnic Institute of Bragança

Co-Supervisor: Ana I. Pereira, Ph.D., Polytechnic Institute of Bragança

Programa Doutoral em Engenharia Electrotécnica e de Computadores

December, 2024

Faculdade de Engenharia da Universidade do Porto

AI-Enhanced Indoor Localization System for Autonomous Mobile Robots

João Afonso Braun Neto

Dissertation submitted to Faculdade de Engenharia da Universidade do Porto
to obtain the degree of

Doctor Philosophiae in Electronic & Computer Engineering

Chair of the Jury: Ph.D. Luís Miguel Pinho de Almeida

Committee Member: Ph.D. Vítor Manuel Ferreira dos Santos

Committee Member: Ph.D. António Luís Gomes Valente

Committee Member: Ph.D. António Paulo Gomes Mendes Moreira

Committee Member: Ph.D. Paulo José Cerqueira Gomes da Costa

December, 2024

I dedicate this thesis to my family.

Abstract

In today's Industry 4.0 (I4.0) environment, Autonomous Mobile Robots (AMRs) are crucial in transforming warehouse operations by enhancing automation, efficiency, and accuracy. Despite significant advancements, current localization systems for AMRs often fall short in dynamic environments due to their reliance on static assumptions, leading to decreased performance. Thus, this dissertation advances the field of indoor localization for AMRs within dynamic I4.0 environments by integrating artificial intelligence (AI) with traditional localization methodologies. The research uses advanced object detection algorithms to develop the Robust Perfect Match (RoPM), an AI-enhanced indoor localization system. This system significantly improves localization accuracy and robustness compared to conventional systems, particularly in environments with frequent and unpredictable changes. It effectively addresses the limitations of state-of-the-art localization solutions, which often rely on prohibitively expensive and power-intensive hardware. It provides a cost-effective alternative that does not assume static environmental conditions. The research employs an exhaustive experimental framework, using robotics simulation software integrated with a LIDAR model for initial simulations in a controlled setting, followed by real-world tests in a custom-designed I4.0 factory mockup. These evaluations thoroughly assess the system's performance using a variety of statistical metrics and visual techniques to ensure its practicality and effectiveness in dynamic settings. The results indicate that the RoPM system consistently outperforms the traditional localization system. It addresses the shortcomings of traditional systems that falter due to dynamic obstacles, maintaining high accuracy and operational reliability with cost-effective hardware. Its ability to detect and adapt to dynamic obstacles significantly reduces localization errors and enhances the reliability of AMR operations in I4.0 settings. This research addresses the gap between theoretical advancements and practical applications. Also, it sets a new benchmark for cost-effective, scalable solutions in dynamic environments, aligning with the strategic goals of I4.0. The RoPM system offers an economically viable solution that balances the need for dynamic adaptability with the economic constraints of industrial deployment for scalability, providing a pragmatic framework for the future of smart manufacturing and warehousing. Future work will focus on refining the covariance estimates for the Perfect Match algorithm, enhancing the synchronization of sensors' data for operational speed, and expanding system visibility with an omnidirectional camera setup to leverage the RoPM's capabilities further.

Keywords: Indoor localization system. Artificial intelligence. Autonomous mobile robots. Convolutional neural networks. Object detection. YOLO. Dynamic SLAM. Robust SLAM. Semantic SLAM. Industry 4.0. Fiducial markers. Map matching. Perfect Match. Automated warehouses.

Resumo

No ambiente atual da Indústria 4.0 (I4.0), os Robôs Móveis Autônomos (AMRs) são cruciais na transformação das operações de armazém, melhorando a automação, a eficiência e a sua consistência. Apesar dos avanços significativos, os sistemas de localização atuais para AMRs muitas vezes não são eficazes em ambientes dinâmicos devido à sua dependência de pressupostos de ambientes estáticos, resultando num desempenho reduzido. Assim, esta tese propõe avanços no campo da auto localização para AMRs em ambientes dinâmicos da I4.0, integrando inteligência artificial (IA) com metodologias de localização tradicionais. A investigação utiliza algoritmos avançados de detecção de objetos para desenvolver o Robust Perfect Match (RoPM), um sistema de auto localização apoiado por IA. Este sistema melhora significativamente a exatidão e a robustez da localização em comparação com o sistema original, especialmente em ambientes dinâmicos e imprevisíveis. O sistema proposto aborda eficazmente as limitações das soluções de auto localização atuais, que dependem muitas vezes de hardware de elevado custo e com alto consumo energético, oferecendo uma alternativa económica que não assume condições de ambiente estáticas. A investigação recorre a uma estrutura experimental exaustiva, começando pela utilização de software de simulação robótica integrado com um modelo LIDAR para simulações iniciais num ambiente controlado, seguido de testes reais numa maquete de fábrica I4.0. Estes testes analisam rigorosamente o desempenho do sistema usando uma variedade de métricas estatísticas para garantir a sua confiabilidade e eficácia em configurações de ambientes dinâmicos. Os resultados indicam que o sistema RoPM supera consistentemente o sistema de localização original uma vez que aborda as deficiências dos sistemas tradicionais que falham devido a obstáculos dinâmicos, mantendo alta precisão e confiabilidade operacional. A sua capacidade de detetar e adaptar-se a obstáculos dinâmicos reduz significativamente os erros de localização e melhora a fiabilidade das operações dos AMRs em ambientes I4.0. Esta investigação preenche a lacuna entre os avanços teóricos e as aplicações práticas e estabelece um novo padrão para soluções económicas e escaláveis em ambientes dinâmicos, alinhando-se com os objetivos estratégicos da I4.0. O sistema RoPM oferece uma solução economicamente viável que equilibra a necessidade de adaptabilidade dinâmica com as restrições económicas de implementação industrial para escalabilidade, fornecendo um quadro pragmático para o futuro da manufatura e armazenamento inteligentes. O trabalho futuro deverá focar na melhoria das estimativas de covariância para o algoritmo Perfect Match, no melhoramento do sincronismo dos dados dos sensores para a velocidade operacional e na expansão da visibilidade do sistema com uma configuração de câmara omnidirecional para aproveitar ainda melhor as capacidades do RoPM.

Keywords: Sistema de localização interna. Inteligência artificial. Robôs móveis autônomos. Redes neurais convolucionais. Detecção de objetos. YOLO. SLAM dinâmico. SLAM robusto. SLAM Semântico. Indústria 4.0. Marcadores fiduciais. Casamento de mapas. Perfect Match. Armazéns automatizados.

Acknowledgments

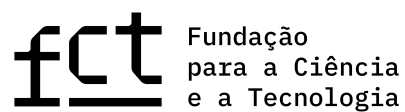
Words cannot fully express my profound gratitude to my mother, father, and grandparents, whose unwavering love and support have been the foundation of my journey. I also want to express my heartfelt gratitude to my beloved Liandra Gracher, who appeared in one of the most challenging times of my life and supported me through it; I will never forget this.

In addition, I would like to express my deepest gratitude to my supervisors, Professors Paulo Costa, José Lima, and Ana Isabel Pereira, for their invaluable guidance, patience, and support throughout this research. Their insights and expertise have shaped this work and outcome. They also contributed extensively to my personal, professional, and scientific growth. I also want to thank them for the opportunities they have given me and for believing in me.

My heartfelt thanks go to the members of CeDRI Research Center, especially Guido Berger, Arezki Chellal, Alexandre Oliveira, Flávia Pires, among others, for their camaraderie and sustained support. The discussions we shared contributed to my research and personal growth.

Lastly, I wish to extend my appreciation to all those who were involved in any capacity and helped make this thesis possible.

João Braun



The project that gave rise to these results received the support of a fellowship from "la Caixa" Foundation (ID 100010434). The fellowship code is LCF/BQ/DI20/11780028 (the contract period ran from December 9, 2020, to November 8, 2023). This work was supported by FCT - Fundação para a Ciência e Tecnologia within the scope of the PhD Grant 2023.04536.BD (the contract period ran from December 1, 2023, to November 1, 2024).

Publications

This chapter lists all the important scientific publications in indexed conferences, book chapters, and journals related to the development of this PhD research. The list of publications made during the four years of research chronologically and logically constructed the solid foundation of knowledge to perform this research work, which the publications as first author represent the core of contributions of this PhD dissertation.

List of Journal Publications:

- **First author:**

- J. Braun, J. Lima, A. I. Pereira, and P. Costa, "Kabsch Marker Estimation Algorithm, a multi-robot marker-based localization algorithm within the Industry 4.0 Context," IEEE Access, 2024.
- J. Braun et al., "A robot localization proposal for the RobotAtFactory 4.0: A novel robotics competition within the Industry 4.0 concept," Frontiers in Robotics and AI, vol. 9, p. 1023590, 2022.

- **Collaborations:**

- L. C. Klein et al., "Angle Assessment for Upper Limb Rehabilitation: A Novel Light Detection and Ranging (LiDAR)-Based Approach," Sensors, vol. 24, no. 2, p. 530, 2024.
- L. C. Klein et al., "A machine learning approach to robot localization using fiducial markers in RobotAtFactory 4.0 competition," Sensors, vol. 23, no. 6, p. 3128, 2023.

List of Conference/Book Chapters Publications (Indexed):

- **First author:**

- J. Braun et al., "Design and development of an omnidirectional mecanum platform for the RobotAtFactory 4.0 competition," in Climbing and Walking Robots Conference, 2023, pp. 114–125.
- J. Braun et al., "RobotAtFactory 4.0: a ROS framework for the SimTwo simulator," in 2022 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), 2022, pp. 205–210.
- J. Braun et al., "Object detection for indoor localization system," in International Conference on Optimization, Learning Algorithms and Applications, 2022, pp. 788–803.

- J. Braun et al., "Robot@ factory lite competition: a digital twin approach for the AGV," in 11th International Conference on Simulation and Modeling Methodologies, Technologies and Applications, SIMULTECH 2021, 2021, pp. 311–318.
- J. Braun et al., "Searching the optimal parameters of a 3D scanner through particle swarm optimization," in International Conference on Optimization, Learning Algorithms and Applications, 2021, pp. 138–152.
- J. Braun et al., "Three-dimensional scanning system based on a low-cost infrared sensor," in 2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), 2021, pp. 01–08.

• **Collaborations:**

- A. A. Chellal et al., "Energy Efficiency Analysis of Differential and Omnidirectional Robotic Platforms: A Comparative Study," in Climbing and Walking Robots Conference, 2023, pp. 103–113.
- J. Bierende et al., "Volume Estimation of an Indoor Space with LiDAR Scanner," in International Conference on Optimization, Learning Algorithms and Applications, 2022, pp. 78–92.
- A. A. Chellal et al., "Modeling of a lithium-ion battery for enhanced power management in robotics domain," in 2023 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), 2023, pp. 126–131.
- L. C. Klein et al., "Using machine learning approaches to localization in an embedded system on RobotAtFactory 4.0 competition: a case study," in 2023 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), 2023, pp. 69–74.
- L. C. Klein et al., "Deep Learning-Based Localization Approach for Autonomous Robots in the RobotAtFactory 4.0 Competition," in International Conference on Optimization, Learning Algorithms and Applications, 2023, pp. 181–194.
- L. B. Junior et al., "A Comparison of Fiducial Markers Pose Estimation for UAVs Indoor Precision Landing," in International Conference on Optimization, Learning Algorithms and Applications, 2023, pp. 18–33.
- L. Pilarski et al., "An AI-based object detection approach for robotic competitions," in 2023 3rd International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME), 2023, pp. 1–6.
- M. Nowakowski et al., "Using LiDAR Data as Image for AI to Recognize Objects in the Mobile Robot Operational Environment," in International Conference on Optimization, Learning Algorithms and Applications, 2023, pp. 118–131.
- L. B. Junior et al., "A Comparison of PID Controller Architectures Applied in Autonomous UAV Follow up of UGV," in Robot 2023: Sixth Iberian Robotics Conference: Advances in Robotics, Volume 2, 2024, vol. 2, p. 26.
- T. Ferreira et al., "Robot at factory 4.0: an auto-referee proposal based on artificial vision," in Iberian Robotics conference, 2022, pp. 475–487.
- E. Ferreira et al., "Development of a Low-Cost 3D Mapping Technology with 2D LIDAR for Path Planning Based on the A* Algorithm," in Iberian Robotics conference, 2023, pp. 53–66.

- L. E. Luiz et al., "Robot at factory lite-a step-by-step educational approach to the robot assembly," in Iberian Robotics Conference, 2022, pp. 550–561.
- L. Bonzatto Junior et al., "A Comparison of PID Controller Architectures Applied in Autonomous UAV Follow up of UGV," in Iberian Robotics conference, 2023, pp. 26–37.
- J. Lima et al., "A realistic simulation environment as a teaching aid in educational robotics," in 2022 Latin American Robotics Symposium (LARS), 2022 Brazilian Symposium on Robotics (SBR), and 2022 Workshop on Robotics in Education (WRE), 2022, pp. 430–435.
- M. Nowakowski et al., "Advance Reconnaissance of UGV Path Planning Using Unmanned Aerial Vehicle to Carry Our Mission in Unknown Environment," in Iberian Robotics conference, 2023, pp. 50–61.

“It is impossible to begin to learn that which one thinks one already knows.”

Epictetus - Greek Stoic Philosopher

Contents

List of Figures	xiv
List of Tables	xv
List of Acronyms	xix
1 Introduction	1
1.1 Background and Context	1
1.2 Problem Statement, Hypothesis, and Research Questions	5
1.3 Research Objectives	7
1.4 Research Methodology	8
1.5 Document Organization	9
2 State of the Art in Autonomous Mobile Robots' Localization Systems	11
2.1 Overview of Autonomous Mobile Robots	11
2.2 Localization Systems in Autonomous Mobile Robotics	14
2.2.1 Technologies	16
2.2.2 Fundamental Problems of AMRs' Localization	20
2.2.3 Sensor Fusion Techniques	20
2.2.4 Mapping	22
2.2.5 Simultaneous Localization and Mapping	23
2.2.6 Visual SLAM	24
2.2.7 Map Matching Algorithms	25
2.2.8 Other Approaches	26
2.3 State of the Art in Object Detection Algorithms	28
2.4 AI in Robotics Applied to Localization Systems	33
2.5 Localization in Indoor Environments	36
2.6 PhD Motivation & Future Directions	37
3 Design and Development of the Robotic Platform and Localization Systems	39
3.1 Robot's Architecture & Design	41
3.1.1 Mechanical Design	41
3.1.2 Kinematics	43
3.1.3 Hardware Architecture	44
3.1.4 Software Architecture	48
3.1.5 Localization System	59
3.1.6 Path Planning Module	64
3.1.7 Trajectory Planning	65
3.2 Development of the Ground Truth System	67

3.2.1	System Architecture	68
3.2.2	The Local Marker Estimation Algorithm	70
3.2.3	The Kabsch Marker Estimation Algorithm	72
3.2.4	Theoretical Advantages of the KMEA Over the LMEA	75
3.2.5	Integration with AMRs	76
3.2.6	Improvements to the Ground Truth System	77
3.3	AI-Enhanced Indoor Localization System	78
3.3.1	System Architecture	80
3.3.2	Perfect Match	90
3.3.3	Localization System	93
3.3.4	Outlier Rejection Approach	96
4	Experimental Setup and Validation Methodology	98
4.1	Simulation Environment	98
4.2	Real-world Industry 4.0 Factory Mockup	102
4.3	Data Collection and Preprocessing	104
4.3.1	Statistical Metrics for System Validation	104
4.3.2	Ground Truth Validation Methodology	106
4.3.3	AI-Enhanced Indoor Localization System Validation Methodology	109
5	Results and Discussion	113
5.1	Ground Truth Results	113
5.1.1	Stage I. Several Markers Estimated Under Normal Conditions	113
5.1.2	Stage II. Several Markers Estimated With a 20 cm Height Support	116
5.1.3	Stage III. Several Markers Estimated With Orthonormal Basis Markers Distorted	117
5.1.4	Stage IV. Several Markers Estimated With Orthonormal Basis Markers Distorted, and the Camera's Pose Changed	121
5.1.5	Z-Correction Algorithm Results	124
5.2	AI-Enhanced Indoor Localization System Results	127
5.2.1	Simulation	127
5.2.2	Real-world	135
6	Conclusion and Future Work	144
6.1	Principal Outcomes	144
6.2	Development and Validation of the Robotic Platform	145
6.3	Development and Validation of Ground Truth System	146
6.4	Study Case and Testing Scenario	147
6.5	System Testing and Hypothesis Validation	148
6.6	Contributions to Indoor Localization Systems	148
6.7	Limitations and Challenges	149
6.8	Future Directions	149
	Bibliography	151

List of Figures

1.1	Design Science Research Methodology process model	8
2.1	Four major challenges in mobile robotics	12
2.2	Simplified AMR system block diagram	12
2.3	Block diagram of a hypothetical wheeled AMR system	13
2.4	Classification of object detection algorithms	29
2.5	Milestones of object detection evolution	30
3.1	Overall system architecture	40
3.2	Operational modes of the RoPM application showing data flow from different sources	40
3.3	Isometric views of the AMR 's design and its prototype	42
3.4	The prototype with design improvements for the localization systems' validation	43
3.5	Robot's kinematic architecture	44
3.6	Robot's hardware architecture diagram	45
3.7	Robot's power architecture diagram	48
3.8	Software architecture data flow diagram	49
3.9	Speed closed-loop step response measurement, following a method for bypassing motor dead zone	58
3.10	Speed closed-loop step response of one of the motors with the PI-Feedforward controller	58
3.11	DFD for the EKF-based localization using ArUco markers as absolute reference sensors in the AMR	61
3.12	GotoXY trajectory controller	66
3.13	ArUco markers detection software's interface	69
3.14	Data flow diagram describing the system's data flow during the pose estimation process	70
3.15	Flowchart of the LMEA process	71
3.16	Flowchart of the KMEA process	74
3.17	Parallax's effect on the elevated marker perceived pose	77
3.18	Effects of dynamic entities in the perception field of the LiDAR scanner	79
3.19	Simple test showing the robot observing with the camera the outliers	79
3.20	Localization system's visual representation of the detected objects	80
3.21	Communication diagram between system devices	81
3.22	System architecture's DFD in "Online" mode.	82
3.23	System architecture - threads interaction diagram	83
3.24	Four straight views of the real robot with the camera modules	85
3.25	System architecture DFD in "Offline" mode	85

3.26	GUI interface of the AMR application	88
3.27	Localization application running in “Online” mode without personnel	90
3.28	Outlier detection and rejection process	90
3.29	Map (a) and distance map (b) of the environment	91
3.30	Curve of the M-estimator with c_{err} of 70	92
3.31	Gradients applied in x and y direction of the distance map before and after applying the M-estimator	93
3.32	DFD for EKF-based localization using PM/RoPM as absolute reference sensors in the AMR	95
3.33	Sensor coordinate frames transformations for the AMR	96
4.1	Simulated environment and the real scenario	99
4.2	Distance trend of mean residuals and standard deviations - real LIDAR	100
4.3	Linear model of the standard deviation of the YDLIDAR X4 scanner	101
4.4	Distance trend of mean errors and standard deviations - Simulated LIDAR	101
4.5	Linear model of the standard deviation of the simulated YDLIDAR X4 scanner	102
4.6	RAF factory floor mockup	103
4.7	RAF structure	103
4.8	ArUco detection software - Stage I validation	107
4.9	Stage II validation	108
4.10	ArUco detection software - Stage III validation	109
4.11	ArUco detection software - Stage IV validation	109
4.12	Simulation - Case Study I conditions	110
4.13	Simulation - Case Study II conditions	110
4.14	Real - Baseline scenario	111
4.15	Real - Case Study I scenario	111
5.1	Stage I - Comparative scatter plots of markers’ position estimations by KMEA and LMEA	114
5.2	Stage I - Bar plots of error metrics of both algorithms described in the test methodology	115
5.3	Stage I - Scatter plot of both algorithms and the ground truth position of marker ID 12	116
5.4	Stage II - Comparative scatter plots of markers’ position estimations by KMEA and LMEA	117
5.5	Stage II - Bar plots of error metrics of both algorithms described in the test methodology	118
5.6	Stage III - Comparative scatter plots of markers’ position estimations by KMEA and LMEA	119
5.7	Stage III - Bar plots of error metrics of both algorithms described in the test methodology	120
5.8	Stage III - Scatter plot of both algorithms and the ground truth position of marker ID 34	121
5.9	Stage IV - Comparative scatter plots of markers’ position estimations by KMEA and LMEA	122
5.10	Stage IV - Bar plots of error metrics of both algorithms described in the test methodology	123
5.11	Stage IV - Scatter plot of both algorithms and the ground truth position of marker ID 30	124

5.12	Illustration of the parallax effect on KMEA pose estimation without compensation	125
5.13	KMEA pose estimation after parallax compensation	126
5.14	Case Study I: Robot state variables' absolute errors comparison of the original and RoPM systems	128
5.15	Case Study I: PM loss of the original and RoPM systems	129
5.16	Case Study I: <i>XY</i> plot of the ground truth, the original, and the RoPM systems . .	130
5.17	Case Study II: Trajectories plot of the ground truth, the original, and the RoPM systems	131
5.18	Case Study II: <i>XY</i> plot of the ground truth, the original, and the RoPM systems .	132
5.19	Case Study II: Robot state variables' absolute errors comparison of the original and the RoPM systems	132
5.20	Case Study II: PM loss of the original and the RoPM systems	133
5.21	Real Scenario Baseline Study: Trajectory plot comparison of the original and the ground truth systems	135
5.22	Real Scenario Baseline Study: Robot state variables' absolute errors of the original system	136
5.23	Real Scenario Baseline Study: PM loss of the original system	136
5.24	Real Scenario Baseline Study: <i>XY</i> plot of the ground truth and original systems .	137
5.25	Real Scenario Case Study I: Trajectory comparison of the ground truth, the original, and the RoPM systems	138
5.26	Real Scenario Case Study I: Robot state variables' absolute errors comparison of the original and the RoPM systems	138
5.27	Real Scenario Case Study I: PM loss of the original and the RoPM systems . . .	139
5.28	Real Scenario Case Study I. <i>XY</i> plot of the ground truth, the original, and the RoPM systems	140
5.29	Real Scenario Case Study II: Robot state variables' absolute errors comparison of the original and the RoPM systems	141
5.30	Real Scenario Case Study II. PM loss comparison of the original and the RoPM systems	141
5.31	Real Scenario Case Study II. <i>XY</i> plot of the ground truth, the original, and the RoPM systems	143

List of Tables

2.1	Summary of the analyzed review papers on indoor localization systems for AMRs	15
2.2	Classification of sensors used in mobile robotics by usage type and functional axes	17
2.3	Object detection review papers from the last five years	33
2.4	State of the art in Dynamic/Robust/Semantic SLAM approaches	35
2.5	SWOT analysis of the PhD research on AI-enhanced indoor localization	38
3.1	Robot's hardware components	47
5.1	Stage I - Comparison of estimation metrics	116
5.2	Stage II - Comparison of estimation metrics	117
5.3	Stage III - Comparison of estimation metrics	119
5.4	Stage IV - Comparison of estimation metrics	122
5.5	Statistical metrics for the normal and Z-corrected KMEA systems	126
5.6	Metrics' average percentage reduction for the Z-Corrected KMEA system compared to the normal	126
5.7	Case Study I error metrics. Original system.	129
5.8	Case Study I error metrics. RoPM system.	129
5.9	Case Study I: Percentage difference of the RoPM relative to the original system	130
5.10	Case Study II error metrics. Original system	134
5.11	Case Study II error metrics. RoPM system.	134
5.12	Case Study II. Percentage difference of the RoPM relative to the original system.	134
5.13	Real Scenario Baseline Study error metrics. Original system.	137
5.14	Real Scenario Case Study I error metrics. Original system.	139
5.15	Real Scenario Case Study I error metrics. RoPM system.	140
5.16	Real Scenario Case Study I. Percentage difference of the RoPM relative to the original system.	140
5.17	Real Scenario Case Study II. Original system.	142
5.18	Real Scenario Case Study II. RoPM system.	142
5.19	Real Scenario Case Study II. Percentage difference of the RoPM relative to the original system.	142

List of Acronyms

- AGV** Automatic Guided Vehicle. [1](#), [3](#)
- AI** Artificial Intelligence. [1](#), [2](#), [4–8](#), [10](#), [11](#), [14](#), [15](#), [33](#), [34](#), [37](#), [38](#), [78](#), [79](#), [89](#), [93](#), [97](#), [109](#), [112](#), [127–129](#), [132](#), [144](#), [145](#)
- AMR** Autonomous Mobile Robot. [xii](#), [xiii](#), [1–15](#), [19](#), [22](#), [24](#), [25](#), [27](#), [33](#), [35](#), [39](#), [41–43](#), [59](#), [61](#), [64](#), [66–68](#), [72](#), [76](#), [77](#), [80](#), [81](#), [88](#), [89](#), [95–99](#), [102](#), [111](#), [116](#), [124](#), [131](#), [144](#)
- CCD** Charged-Coupled Device. [26](#)
- CCW** Counter Clockwise. [90](#)
- CNN** Convolutional Neural Network. [4](#), [6](#), [15](#), [29–35](#)
- CPR** Counts per revolution. [45](#)
- CPS** Cyber-physical Systems. [2](#)
- CRC** Cyclic Redundancy Check. [51](#), [52](#)
- CTC** Clear Timer on Compare Match. [55](#)
- DFD** Data Flow Diagram. [xii](#), [xiii](#), [48](#), [60](#), [69](#), [78](#), [80](#), [82](#), [85](#), [93](#), [95](#), [112](#)
- DSRM** Design Science Research Methodology. [1](#), [8](#), [9](#)
- EKF** Extended Kalman Filter. [xiii](#), [21](#), [23–27](#), [59](#), [60](#), [62](#), [63](#), [81](#), [82](#), [89](#), [92–95](#), [110](#), [113](#), [127](#), [131](#), [135](#), [137](#), [145](#), [146](#), [149](#), [150](#)
- FM** Fiducial Markers. [18](#)
- FoV** Field of View. [68](#), [69](#), [90](#), [150](#)
- FPN** Feature Pyramid Networks. [31](#), [32](#)
- FPS** frames per second. [32](#)
- GPS** Global Positioning System. [16–18](#), [33](#), [67](#)
- GUI** Graphical User Interface. [xiii](#), [82](#), [88](#), [89](#), [96](#)
- HOG** Histograms of Oriented Gradient. [30](#)

- I4.0** Industry 4.0. [1](#), [2](#), [5–10](#), [27](#), [37](#), [38](#), [41](#), [50](#), [67](#), [68](#), [77](#), [91](#), [98](#), [102](#), [124](#), [144–149](#)
- ICP** Iterative Closest Point. [26](#)
- IFR** International Federation of Robotics. [2](#)
- IMC** Internal Model Control. [57](#)
- IMU** Inertial Measurement Unit. [3](#), [6](#), [15](#), [18](#)
- IoT** Internet of Things. [2](#), [3](#)
- IoU** Intersection over Union. [31](#)
- ISR** Interrupt Service Routine. [54](#), [56](#)
- KF** Kalman Filter. [21](#)
- KMEA** Kabsch Marker Estimation Algorithm. [xii–xv](#), [67](#), [69](#), [72–75](#), [107–109](#), [113–126](#), [144](#), [147](#)
- LED** Light-emitting diode. [19](#)
- LiDAR** Light Detection and Ranging. [xii](#), [4](#), [6](#), [14](#), [15](#), [24](#), [26](#), [34](#), [42](#), [43](#), [46](#), [47](#), [50–52](#), [78–82](#), [86](#), [88–99](#), [111](#), [112](#), [127](#), [128](#), [132](#), [133](#), [136](#), [150](#)
- LMEA** Local Marker Estimation Algorithm. [xii](#), [xiii](#), [67](#), [69–71](#), [75](#), [107](#), [108](#), [113–124](#)
- LSVM** Latent Support Vector Machines. [30](#)
- MAE** Mean Absolute Error. [105](#), [113–115](#), [118–121](#), [123](#), [130](#), [133](#), [134](#), [136](#), [137](#), [139](#), [140](#), [142](#), [145](#)
- mAP** mean average precision. [32](#)
- MaxAE** Maximum Absolute Error. [105](#), [113–115](#), [118–121](#), [123](#), [129](#), [139](#), [140](#)
- MinAE** Minimum Absolute Error. [105](#)
- ML** Machine Learning. [3](#)
- MLP** Multilayer Perceptron. [34](#)
- MRE** Mean Relative Error. [67](#), [106](#), [113–115](#), [118–120](#), [122](#), [123](#)
- MTU** Maximum Transmission Unit. [51](#), [86](#), [87](#)
- NDT** Normal Distribution Transform. [26](#)
- ORB** Oriented FAST and Rotated BRIEF. [25](#)
- PF** Particle Filter. [23](#), [26](#)
- PGO** Pose Graph Optimization. [24](#)

- PI** Proportional-Integral Controller. 57, 58
- PM** Perfect Match. xiii, xiv, 25, 26, 37, 78, 81, 82, 88–96, 110, 113, 127–129, 133–137, 139, 141, 142, 149
- PSO** Particle Swarm Optimization. 25, 26
- PWM** Pulse-width Modulation. 55, 57, 59
- R-CNN** Region-Based Convolutional Neural Network. 29–33
- RaaS** Robot-as-a-Service. 2
- RAF** RobotAtFactory 4.0. xiii, 27, 41, 50, 60–62, 64, 66, 98, 103, 124, 146–148
- RANSAC** Random Sample Consensus. 100
- RFID** Radio Frequency Identification Device. 15, 18, 28
- RGB-D** Red Green Blue - Depth. 18, 25, 34, 35
- RMSE** Root Mean Squared Error. 104, 105, 113–115, 118–120, 122, 123, 130, 133, 134, 136, 137, 140, 145
- RNN** Recurrent Neural Network. 34
- RoPM** Robust Perfect Match. xii–xv, 37–41, 78, 80, 81, 95, 110, 112, 113, 127–135, 137–150
- ROS** Robot Operating System. 15, 25
- RPROP** Resilient Backpropagation. 26
- RSSI** Received Signal Strength Indicator. 15, 28
- SDL** Simple DirectMedia. 54
- SfM** Structure from Motion. 15
- SIFT** Scale-Invariant Feature Transform. 19, 25
- SLAM** Simultaneous Localization and Mapping. xv, 4, 5, 14, 15, 18, 19, 23, 24, 26, 27, 34–37
- SOTA** State of the Art. 14, 20, 37, 146, 147, 149
- SSD** Single Shot Detector. 32
- STD** Standard Deviation. 101, 104, 113–116, 118–123, 125, 126, 131, 133–135, 139, 145
- SURF** Speeded Up Robust Features. 19, 25
- SVM** Support Vector Machines. 29, 31
- TCP** Transmission Control Protocol. 52
- ToF** Time of Flight. 16
- UAS** Unmanned Aircraft Systems. 15

UDP User Datagram Protocol. [46](#), [48–54](#), [82](#), [86](#), [87](#)

UI User Interface. [84](#)

UKF Unscented Kalman Filter. [21](#), [23](#), [24](#)

UWB Ultra-wideband. [33](#)

vSLAM Visual Simultaneous Localization and Mapping. [4](#), [15](#), [25](#), [34](#), [36](#), [37](#)

WMS Warehouse Management System. [3](#)

WSL2 Windows Subsystem for Linux 2. [86](#)

XOR Exclusive OR. [48](#), [52](#)

YOLO You Only Look Once. [31–33](#), [50](#), [53](#), [54](#), [81](#), [88](#), [128](#), [133](#), [134](#), [149](#)

Chapter 1

Introduction

In this era of rapid technological change, using [Autonomous Mobile Robots \(AMRs\)](#) in [Industry 4.0 \(I4.0\)](#) warehouses transforms logistics and manufacturing operations. These intelligent systems are becoming essential for increasing automation, accuracy, and efficiency. As [AMRs](#) develop, they depend heavily on advanced localization systems to navigate and execute tasks in dynamic and unpredictable environments. This research focuses on enhancing these systems' robustness, accuracy, precision, and cost-effectiveness by applying advanced object detection [Artificial Intelligence \(AI\)](#). The rise of [I4.0](#) has significantly increased the role of [AMRs](#) in optimizing warehouse workflows, reducing labor costs, and improving overall productivity. However, the shift to fully autonomous operations in dynamic settings comes with challenges, including the need for better localization techniques that can adapt to changes in real time without high costs. This thesis proposes a new [AI](#)-enhanced approach to [AMR](#) localization to overcome these challenges, ensuring these robots can operate effectively in modern warehouses' complex and dynamic landscape. The rest of the chapter is organized as follows: Section [1.1](#) introduces foundational concepts of [AMRs](#) and [I4.0](#) warehouses, highlighting their impact on warehouse operations and the associated localization challenges. It also discusses how advancements in object detection [AI](#) can address these challenges. Section [1.2](#) defines the main research problem and outlines the hypothesis and guiding research questions. The details of the goals of the research, focusing on developing a robust, cost-effective localization system for dynamic environments, are presented in Section [1.3](#). Section [1.4](#) summarizes the methodological framework, emphasizing the use of the [Design Science Research Methodology \(DSRM\)](#). Finally, Section [1.5](#) briefly describes the thesis's overall structure and subsequent chapters' content.

1.1 Background and Context

Introduction to Autonomous Mobile Robots and Industry 4.0 Warehouses

[AMRs](#) are self-navigating robots that can move materials and perform tasks without human intervention. Unlike [Automatic Guided Vehicles \(AGVs\)](#), which rely on predefined paths and require significant infrastructure for guidance, [AMRs](#) use advanced sensors, cameras, and onboard com-

puting power to interact with their environment. This capability allows them to navigate complex and changing environments, making them highly adaptable and efficient.

In modern industry, [AMRs](#) are crucial in optimizing workflows, reducing labor costs, and increasing productivity. According to a report by the [International Federation of Robotics \(IFR\)](#) in [1], the use of service robots, including [AMRs](#) [2], has been steadily increasing, with global sales of professional service robots growing by 48% in 2022. This growth is driven by the need for automation in several sectors, including manufacturing, logistics, agriculture, cleaning duties, inspection and maintenance, search and rescue, and healthcare, where [AMRs](#) enhance operational efficiency and provide flexibility in handling tasks. In particular, [I4.0](#) warehouses have significantly benefited from integrating mobile robot solutions, especially in transportation and logistics. In addition, according to [1], mobile robots for these applications saw a 44% increase in units sold in 2022, making up a substantial portion of the professional service robots market. Over half of the professional service robots sold that year were designed to transport goods or cargo, highlighting their critical role in streamlining logistics operations. In addition, still in [1], [IFR](#) states that although traditional sales remain the primary method of business model, the trend towards [Robot-as-a-Service \(RaaS\)](#) business models is growing in popularity, with the [RaaS](#) fleet growing by 67% in 2022, indicating a shift towards more flexible and scalable automation solutions.

These statistics show the importance of research in this area as the demand for advanced solutions snowballs. However, implementing [I4.0](#) solutions, including localization systems of [AMRs](#), faces significant challenges. These include skepticism about the benefits of digitalization, lack of commitment within companies, high implementation costs, and a shortage of skilled workforce and technical support, which are critical barriers to the successful deployment of these technologies [3]. Therefore, as evidenced by authors in [3], the research on smart warehouses in [I4.0](#) is comprehensive. However, with some critical gaps, it has several different categories and subcategories that aim to improve all the interconnected technologies that form the concept of a smart [I4.0](#) warehouse.

[I4.0](#), recognized as the fourth industrial revolution, combines [Cyber-physical Systems \(CPS\)](#), [Internet of Things \(IoT\)](#), edge computing, fog computing, cloud computing, deep learning, digital twin, and advanced data analytics to transform manufacturing and logistics processes [4, 5]. This integration aims to develop intelligent factories where machines and systems interact seamlessly, facilitating highly automated and optimized operations.

Automated warehouses are cornerstones of [I4.0](#), employing cutting-edge technologies such as robotics, [IoT](#) sensors, [AI](#), and big data analytics to improve inventory management, order fulfillment, and material handling. Key elements of an automated warehouse include:

- **Robotic Systems:** [AMRs](#) and robotic arms handle tasks such as picking, packing, and transporting goods. Equipped with [AI](#) algorithms, they autonomously navigate warehouse environments, optimizing pick routes and adapting to changes in warehouse layout [6].
- **[IoT](#) Devices:** These devices are embedded in the equipment of warehouses to gather real-time data, enabling monitoring and tracking of products, crucial for optimizing warehouse

operations and ensuring efficient decision-making processes [3].

- **Edge Computing:** Edge computing processes data near where it is generated, minimizing the need to relay it to central servers. In warehouses, this approach allows edge devices like robots and sensors to manage data on-site, thereby reducing latency and enhancing response times to improve operational efficiency [6]. Advances in the miniaturization and energy efficiency of processing units have led to more powerful IoTs devices capable of robust data processing and analysis. This enhancement supports applications requiring immediate responses and is critical for data-sensitive, mobile platforms such as smartphones, AMRs, and vehicles, including drones, self-driving cars, and AGVs [7].
- **Warehouse Management System (WMS):** Advanced software platforms streamline and enhance warehouse processes, integrating several data sources to improve operational efficiency [6].
- **Machine Learning (ML) Algorithms:** These algorithms are essential for forecasting demand, optimizing storage layouts, and enhancing decision-making processes by analyzing historical data and identifying patterns and trends [6].

In this advanced setup, AMRs enhances automation and efficiency. They navigate across warehouse floors, facilitating the movement of goods between several points and coordinating with other automated systems to maintain seamless and efficient operations [8].

Localization Systems in AMRs

Autonomous Mobile Robotics represents an expanding sub-field within robotics, characterized by vigorous research activity and increasing potential. Despite their increasing deployment, AMRs have yet to peak performance due to several significant challenges prominently discussed in the literature, such as [9]. A critical issue concerns localization—the system’s capability to continuously and accurately determine its position (“Where am I?”) within static and dynamic settings. The challenge nowadays is to focus on the latter since the former is already an extensively researched topic. Some authors consider them a solved topic, at least theoretically [10].

Many different systems and methodologies have been devised in the literature to address this localization challenge. Although these methods vary, they commonly leverage sensors, sensor fusion algorithms, and sophisticated localization techniques to enhance the accuracy of the robot’s position estimation—referred to as “belief” to underscore the probabilistic nature of these techniques. Technology advances have made these systems more reliable and increasingly viable across several societal applications. Alatise and Hancke [9] described that localization strategies are divided into relative and absolute categories. Relative localization integrates data from multiple sensors starting from a known position, typically employing encoders and Inertial Measurement Units (IMUs). In contrast, absolute localization determines the robot’s position and orientation

through systems deployed in the environment, such as navigation beacons, landmarks, or systems that identify environmental features. Robust data fusion algorithms facilitate the integration of these diverse data sources.

Localization systems are indispensable to [AMRs](#), enabling precise navigation and effective operation within their environments. Accurate localization is fundamental, influencing critical operational functions such as path planning, obstacle avoidance, and task execution. Traditional techniques, including [Simultaneous Localization and Mapping \(SLAM\)](#), while foundational, exhibit limitations, particularly in dynamic, cluttered environments like automated warehouses. The primary shortfall of traditional localization frameworks lies in their static environment assumption, often failing in dynamic contexts where the frequent movement of objects and people introduces errors in mapping and pose estimation [11]. This limitation is particularly evident in the domain of [SLAM](#), a cornerstone in robotics, automation, and computer vision for over three decades [12]. For example, sensors like [Light Detection and Ranging \(LiDAR\)](#) and ultrasonic, which mainly provide distance measurements, may inadequately capture the complexities of dynamic settings.

[Visual Simultaneous Localization and Mapping \(vSLAM\)](#) systems represent a significant advancement by using cameras to discern between static and dynamic elements, yet they, too, encounter challenges in distinguishing these features accurately [12]. Furthermore, current dynamic localization methods can be prohibitively expensive and computationally demanding. Integrating deep learning for enhanced obstacle detection and tracking presents a promising avenue but historically requires substantial processing capabilities, posing challenges for real-time applications on embedded systems. Recent strides in hardware and algorithmic efficiency, such as the advent of [Convolutional Neural Networks \(CNNs\)](#), see [13], are beginning to address these limitations, facilitating the deployment of more sophisticated solutions in practical settings. The integration of advanced deep learning techniques in obstacle detection has enhanced [vSLAM](#) algorithms, enabling their operation in dynamic environments by either ignoring or tracking dynamic entities, where they are often classified as “Robust”, “Dynamic” or “Semantic” [SLAM](#) [11, 14, 15]. This development has been bolstered by recent advancements in one-shot and two-shot detectors, which now allow for real-time processing in embedded systems [16], making deploying sophisticated algorithms in practical applications feasible.

Role of Object Detection AI

Object detection [AI](#) is the branch of [AI](#) that focuses on identifying and locating objects within digital images or videos. This technology leverages deep learning models, particularly [CNNs](#), to process visual data, distinguish between object classes, and pinpoint their locations within an image. Ongoing progress has significantly improved accuracy and processing speeds, enabling broader applications like robotics, autonomous vehicles, and surveillance [17]. In this reasoning, these advancements in object detection [AI](#) include the development of more efficient techniques, such as one-stage and two-stage detectors [13, 18], which have reduced computational costs while increasing detection accuracy. These innovations have found extensive applications in robotics,

particularly in enabling robots to interact more effectively with their environments by recognizing objects and obstacles. In autonomous vehicles, object detection AI is critical for real-time decision-making and navigation [17]. Enhanced visual perception capabilities, such as those enabled by Fast R-CNN, further refine these interactions, leading to improved spatial understanding and accurate environment mapping, essential for precise localization in robotic systems [13].

Integrating object detection AI with localization systems, such as SLAM, enhances robotic navigation and environmental interaction capabilities. This integration allows robots to perceive their surroundings and understand and map them accurately, essential for tasks like autonomous navigation and targeted object manipulation. Combining cognitive augmentation and object detection, AI enables robots to process complex sequences and adapt to changes dynamically, ensuring more robust localization and navigation capabilities [17].

Integrating object detection AI with localization systems can significantly improve robotic efficiency and adaptability. For instance, robots equipped with these technologies in industrial settings can autonomously navigate complex environments, identify and pick specific objects, and even collaborate with humans. Such capabilities lead to increased productivity, reduced operational costs, and enhanced safety in dynamic settings [17]. Applying sensors and effectors with object detection AI provides richer data for localization tasks, leading to more dynamic and responsive robotic behavior and enhancing overall system performance and reliability [17].

1.2 Problem Statement, Hypothesis, and Research Questions

Current localization systems for AMRs in I4.0 warehouses have significantly advanced, yet they predominantly operate under the static environment assumption. This limitation leads to sub-optimal performance in dynamic settings characterized by unpredictable elements, frequent changes, and outliers. The traditional systems' lack of semantic understanding and adaptability exacerbates these issues, decreasing accuracy and reliability in real-time operations. Moreover, solutions that adapt to dynamic changes, i.e., do not operate under this assumption, are often prohibitively expensive, relying on costly and power-hungry hardware and extensive processing capabilities, which hampers autonomy, scalability, and feasibility for widespread implementation.

To address these challenges, there is an urgent need for a robust, accurate, and cost-effective indoor localization system tailored for the dynamic environments typical of I4.0 warehouses. Thus, this research proposes developing an AI-enhanced localization system that leverages advanced object detection algorithms to interpret imagery data semantically. This approach allows the system to dynamically adapt to environmental changes and efficiently reject outliers, ensuring high robustness, accuracy, and precision in real-time operations without incurring excessive costs.

This system's enhancements are crucial for significantly improving the operational capabilities of AMRs in automated warehouses. They align with the broader goals of I4.0 by offering a scalable, energy-efficient, and economically viable localization solution suitable for dynamic scenarios.

Building on this problem statement and informed by the background and motivation sections, the following research questions and hypotheses have been formulated:

RQ 1 - *How can object detection AI be integrated into AMRs' localization systems to improve robustness, accuracy, and precision in dynamic and unpredictable I4.0 warehouse environments?*

Object detection AI can be integrated into AMRs' localization systems to enhance their performance in dynamic environments through a variety of approaches, each contributing to a comprehensive solution:

- **Semantic Interpretation:** Using object detection AI, AMRs can distinguish between static and dynamic objects, updating their navigation paths and internal maps in real time to avoid obstacles that have newly appeared or shifted positions.
- **Outlier Rejection:** Advanced AI models enable AMRs to identify and discard anomalies in sensory data, ensuring high accuracy and precision and minimizing the impact of transient disruptions on the system's performance.
- **Data Fusion:** This strategy involves merging and analyzing data from diverse sensors, including LiDARs, cameras, and IMUs, through deep learning models like CNNs. The result is a comprehensive, multidimensional view of the environment that enhances overall decision-making and path planning. This integrated approach allows the AMRs to use a richer data set for more precise localization and navigation, improving robustness and precision in navigating complex warehouse layouts.

RQ 2 - *What factors influence the cost-effectiveness and scalability of AI-enhanced localization systems in dynamic warehouse settings, and how can these factors be optimized?*

Several critical factors impact the cost-effectiveness and scalability of AI-enhanced localization systems in dynamic warehouses, and they can be optimized in the following ways:

- **Resource Optimization:** It is essential to design systems that maximize computational resource efficiency. Developing algorithms that minimize high-frequency updates and data bandwidth reduces operational costs, allowing the system to operate on more affordable hardware.
- **Scalable Deployment Models:** Cloud-based or edge computing architectures distribute computational tasks based on the complexity and immediacy of operations. This strategy lessens the need for high-end computing on each robot, facilitating cost-effective scaling.
- **Software and Algorithm Licensing:** Using open-source tools and developing proprietary algorithms decreases reliance on expensive commercial software, lowering upfront costs and enhancing the system's customization to meet specific warehouse needs.

Based on the discussions of this chapter and insights, this thesis posits the following hypothesis:

Integrating object detection AI with AMRs' localization systems will enhance robustness, accuracy, and precision in dynamic indoor Industry 4.0 environments by employing outlier rejection and data fusion, optimizing computational resources, and using scalable, cost-effective models with a blend of open-source and custom-implemented algorithms.

1.3 Research Objectives

Building upon the identified challenges and the hypotheses formulated in the previous sections, this research aims to develop a solution that enhances the operational capabilities of AMRs in dynamic I4.0 warehouses. The objectives outlined below address the core issues of robustness, accuracy, and cost-effectiveness in AMR localization systems. By focusing on these objectives, the study seeks to connect theoretical advancements with practical applications, ensuring that the developed solutions are effective in simulated environments and scalable and economically viable in real-world settings.

1. **Develop a Simulation Environment:** Create a simulation scenario within an I4.0 warehouse with a simulated robotic platform to initially test and refine the localization system.
2. **Design an AI-Enhanced Localization System:** Develop and implement a cost-effective, adaptive indoor localization system for AMRs using object detection algorithms integrated with AMRs' existing navigation systems. Focus on semantic interpretation of imagery data to adapt to environmental changes and efficiently reject outliers, ensuring system robustness, accuracy, and precision in dynamic I4.0 warehouse environments.
3. **Construct and Test a Real Robotic Platform:** Build a real robotic platform to implement and test the developed localization algorithms. Detail the use of cost-effective components and scalable architectures to validate the performance and adaptability of the localization system in a physical setting.
4. **Develop and Use a Testbed System:** Construct a testbed that mimics a real I4.0 automated warehouse at scale, including developing a ground truth system to validate the localization system's performance. Use this testbed to assess the system's adaptability and effectiveness under laboratory conditions.

By achieving these objectives, the research aims to significantly enhance the operational capabilities of AMRs in automated warehouses, contributing to the goals of I4.0 by providing an accurate, robust, precise, efficient, and economically viable localization solution in dynamic scenarios.

1.4 Research Methodology

The methodology for this research is guided by the **DSRM** model proposed by [19], which is particularly suited for developing and evaluating artifacts designed to solve identified problems. This research aims to develop a robust, cost-effective **AI**-enhanced localization system for **AMRs** in **I4.0** warehouses, leveraging advanced object detection technologies. Figure 1.1 illustrates the **DSRM** process adopted for this thesis.

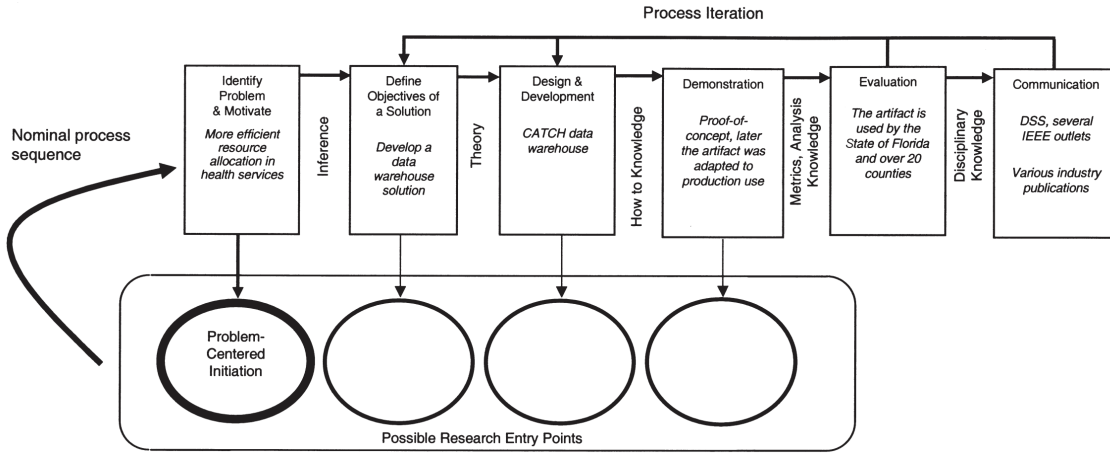


Figure 1.1: Design Science Research Methodology process model [19]

Step 1: Problem Identification and Motivation

The primary challenge addressed in this research is the inefficiency of current localization systems for **AMR** in dynamic environments typical of **I4.0** warehouses. These systems often fail to adapt to environmental changes, decreasing accuracy and reliability. Furthermore, the systems capable of handling dynamic environments without failing are often prohibitively expensive, making them impractical for widespread adoption in the industry. This research aims to develop a cost-effective solution that maintains high performance in dynamic settings without the substantial energy consumption burden typically associated with advanced localization systems.

Step 2: Define the Objectives of a Solution

The objective is to design a localization system that enhances the robustness, accuracy, and precision of **AMRs** in dynamic settings. Incorporate cost-effective technologies to ensure scalability and efficiency.

Step 3: Design and Development

This step involves creating an **AI**-enhanced localization system using object detection algorithms. The system will be designed to interpret complex dynamic environments and adjust the localization process in real time.

Step 4: Demonstration

The system will be tested in two distinct settings to demonstrate its functionality and validate its effectiveness in dynamic environments typical of (I4.0) warehouses. The localization system will initially be evaluated through simulations replicating real-world dynamics and several operational conditions. Following successful simulation trials, the system will undergo further testing in a factory I4.0 mockup developed at scale within a laboratory setting. This two-tier demonstration approach ensures comprehensive system performance validation, allowing adjustments and refinements in a controlled yet realistic environment before real-world application.

Step 5: Evaluation

The localization system will be evaluated based on its accuracy, robustness, and adaptability in simulated and real-world environments. This evaluation methodology will assess the system under several operational conditions to ensure robust performance across different scenarios. Performance metrics from both environments will be compared against the system without AI enhancement for consistency and to validate improvements, gauge cost-effectiveness, and identify any necessary refinements. This thorough evaluation will highlight the system's ability to handle dynamic changes typical of I4.0 warehouses and demonstrate its practical viability for real-world applications.

Step 6: Communication

Findings from this research will be documented in scholarly papers and presentations at scientific conferences to disseminate the results to academic and industrial communities.

Process Iteration

The DSRM model emphasizes iterative refinement. Based on evaluation results, the localization system will undergo iterative improvements to enhance its performance and reliability.

Entry Points

This research enters the DSRM cycle at the “Problem-Centered Initiation”, starting with the specific challenge of improving localization in dynamic warehouse environments. As the research progresses, other entry points, such as “Objective-Centered Solution” and “Design and Development-Centered Initiation” may also become relevant, reflecting the iterative nature of design science research.

1.5 Document Organization

This thesis is structured to comprehensively explore developing and implementing an advanced localization system for AMRs in I4.0 warehouses. The chapters are organized as follows:

- **Chapter 2 - *Related Work*:** This chapter provides an overview of the scientific literature on [AMRs](#), focusing on their localization systems, the role of artificial intelligence in robotics for localization, and the specific challenges they face in indoor environments. This background sets the stage for understanding the gaps and opportunities for innovation in the field.
- **Chapter 3 - *Methodology*:** This chapter details the design and development process of the robotic platform used in this research. It includes creating the ground truth system essential for validating the localization accuracy and developing the localization system itself. Additionally, this chapter discusses the [AI](#) model designed to enhance localization performance.
- **Chapter 4 - *Experimental Setup*:** This chapter describes the simulation environment and the real-world [I4.0](#) factory mockup used for testing. It also outlines the data collection and preprocessing methods essential for ensuring the robustness and reliability of the experimental results.
- **Chapter 5 - *Results and Discussion*:** This chapter presents the results from the developed ground truth system and the localization system's results from simulations and real-world testing.
- **Chapter 6 - *Conclusion and Future Work*:** The final chapter summarizes the essential research and contributions to the field. It discusses the limitations and challenges encountered during the research. It outlines future directions to enhance further the capabilities and applications of the developed research and [AMRs](#) in [I4.0](#) environments. It also states some of the decisions during the development of the system.

By following this structure, the thesis aims to provide a clear and logical progression through the research process, from exploring related work to presenting results and prospects.

Chapter 2

State of the Art in Autonomous Mobile Robots' Localization Systems

This chapter reviews the critical technologies and methodologies employed in robotics for localization, emphasizing the role of [AI](#). Understanding the operational frameworks of [AMRs](#) is imperative and hinges on the interaction of several subprocesses to achieve autonomy. After a brief overview, the discussion extends to critical topics such as localization methods and sensor fusion techniques. Additionally, this review explores deep learning strategies for real-time object detection, a foundational element of this thesis. The focus then shifts to how contemporary research integrates these strategies for effective robot localization.

The literature review aims to delineate current indoor localization methods, [AI](#)-enhanced object detection, and associated technologies. These discussions are crucial for setting the stage for the thesis and aligning with the rigorous research methodology introduced in Chapter 1. Therefore, the organization of this chapter is as follows: Section 2.1 offers a concise overview of [AMRs](#) and their operational processes. Section 2.2 reviews current technologies, sensor fusion techniques, and localization methods used in [AMRs](#). A thorough review of the latest object detection algorithms is presented in Section 2.3. Section 2.4 deepens the discussion on object detection by examining its application to localization in autonomous mobile robotics. Soon after, Section 2.5 narrows the discussion to [AMRs'](#) indoor localization and the current challenges and prospects. Finally, Section 2.6 concludes the chapter by presenting the significance of this PhD thesis. It addresses open issues, challenges, and future directions in the field, comparing these with ongoing research.

2.1 Overview of Autonomous Mobile Robots

[AMRs](#) represent a rapidly evolving class of robots designed to operate unsupervised in dynamic and partially unknown environments. Adopting the definition from [9], we understand that:

“An autonomous mobile robot operates in unpredictable and partially unknown environments. It must navigate seamlessly and avoid obstacles within its movement

confines. An **AMR** functions with minimal human intervention, following a predefined path in indoor and outdoor settings [9].”

This definition encapsulates the complex dynamics of **AMRs**, which face significant challenges as they autonomously navigate real-world environments, illustrated in Figure 2.1.

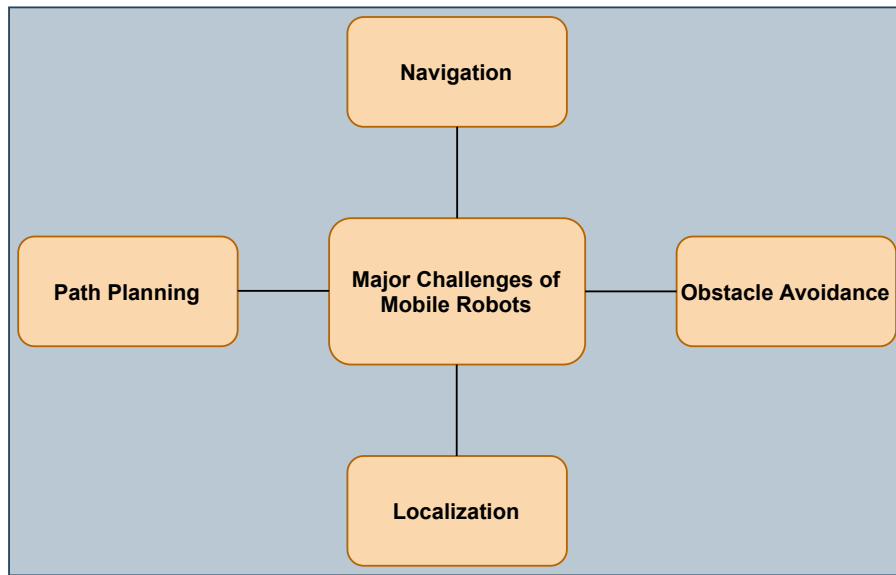


Figure 2.1: Four major challenges in mobile robotics, adapted from [9]

These challenges are inherently interconnected; a robot’s ability to autonomously navigate depends critically on its ability to identify and traverse a collision-free path. Each challenge represents a distinct area of academic research because of this complexity. This interconnection is better illustrated in Figure 2.2.

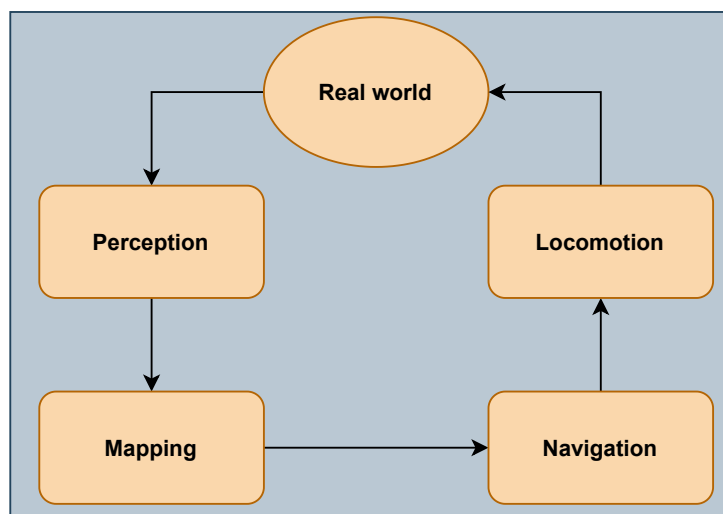


Figure 2.2: Simplified **AMR** system block diagram, adapted from [20]

In this simplified diagram, the sequential process highlights how perception is crucial for the robot to navigate and respond to its environment effectively. The robot can perceive and map its surroundings by acquiring and processing data from several sensors, a foundational step in autonomous operation. Thus, perception is essential in mobile robotics as it allows robots to become aware of their surrounding environment and internal states. High-resolution sensors and sophisticated algorithms enable perception, directly impacting a robot's ability to operate autonomously [20].

In this reasoning, a fundamental understanding of mobile robotics can be gained from the system block diagram of a hypothetical wheeled mobile robot shown in Figure 2.3, simplifying the discussion while remaining applicable to other robot architectures such as the one presented in [20].

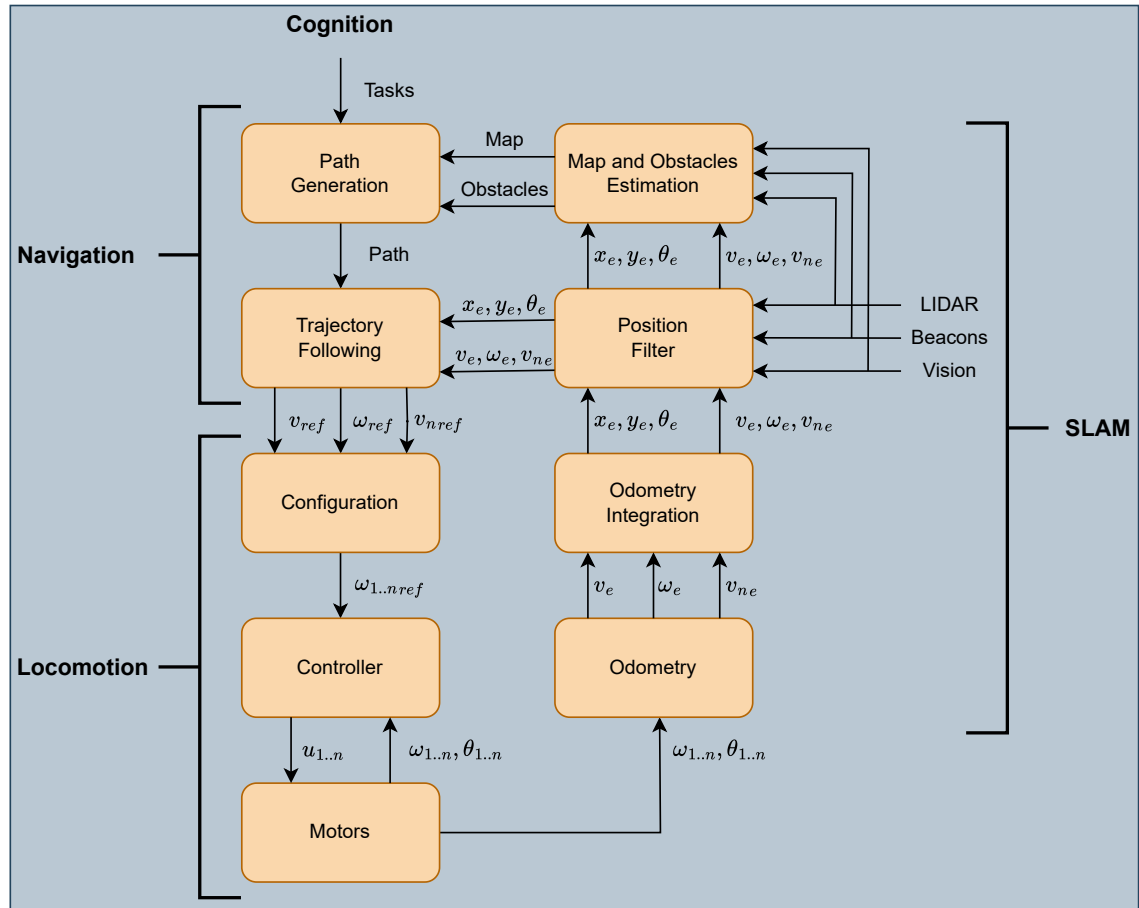


Figure 2.3: Block diagram of a hypothetical wheeled AMR system

Where in the illustration, the following variables are:

- $[v_{ref}, \omega_{ref}, v_{nref}]$: Omnidirectional robot's reference speed states.
- $\omega_{1..nref}$: Wheels' reference speeds.
- $u_{1..n}$: Controller's outputs.

- $\theta_{1..n}$: Orientations of the motors' joints.
- $[v_e, \omega_e, v_{ne}]$: Estimated robot's states speeds.
- $[x_e, y_e, \theta_e]$: Estimated robot's pose.

The diagram begins with cognition, where the robot's **AI** determines the necessary tasks. This leads to path generation for task execution, such as moving a box in a warehouse and using several path-planning techniques. While sometimes sacrificing path optimality for computation speed, these techniques ensure the robot can compute paths efficiently. For a comprehensive review of these techniques, see [21–24]. Once the path is defined, trajectory planning specifies how the robot should follow it, a distinct process often confused with path planning. Subsequent blocks convert the planned trajectory into specific commands for the robot's motors, considering its architecture, whether a differential or omnidirectional drive or other architectures. Finally, the **SLAM** integrates several subsystems, including perception and sensor fusion. It is worth noting that while robots can operate with preloaded maps, genuine autonomy is demonstrated by a robot's ability to perform **SLAM** effectively. This process involves estimating the robot's position using odometry—a technique prone to cumulative errors from factors like wheel slippage. To compensate for these errors, position filters that integrate data from multiple sensors (e.g., vision, beacons, **LiDAR**) refine the robot's state estimates, thus enhancing the accuracy of the trajectory planning and path generation processes in real time.

2.2 Localization Systems in Autonomous Mobile Robotics

Indoor localization is a broad and quickly changing topic in **AMRs**, exploring different methods and technologies. Since plenty of recent reviews and surveys are already available, doing another in-depth review would not be wise and would cost energy without beneficial contributions if the objective was to produce original work. Such an effort would likely repeat what is already known instead of adding new ideas to the discussion. Therefore, the research presented in this work is based on these detailed reviews to provide a solid understanding of the current knowledge. The analysis relies on these reviews, but it is not limited to them. This approach avoids unnecessary repetition and keeps up-to-date research on the **State of the Art (SOTA)**, enhancing the originality of the work. Thus, Table 2.1 summarizes the recent review papers mentioned. It lists their references, topics, technologies covered and the limitations they note on **AMRs**. The research papers were searched using relevant keywords and their combinations, as shown below. The documents were searched across multiple academic databases, and those published over 6 years ago were excluded from the review.

- “Indoor Localization System”
- “Autonomous Mobile Robots”
- “Survey”

- “Review”
- “Robot Indoor Localization”

Table 2.1: Summary of the analyzed review papers on indoor localization systems for AMRs

Reference	Year	Topics and Technologies	Limitations
Saputra et al. [14]	2018	Structure from Motion (SfM), vSLAM, operation in dynamic environments, robust visual SLAM reconstruction, dynamic object segmentation and tracking, joint motion segmentation and reconstruction, taxonomy of existing approaches, practical and robust solutions	Assumption of static environments, handling temporary stopping, computational expense of fitting a motion model, handling noise and outliers, among others.
Panchpor et al. [31]	2018	Localization and mapping in dynamic environments, obstacle detection and avoidance, moving obstacles, AI and reinforcement learning, Radio Frequency Identification Device (RFID) tags	Inaccurate localization due to moving landmarks, exclusion of dynamic features in vSLAM, handling dynamic changes, computational load, sensor limitations, outlier detection, environmental assumptions
Rubio et al. [32]	2019	Locomotion, perception and sensors, cognition and control systems, navigation, obstacle avoidance, positioning techniques, impact on several sectors, emerging technologies	Stability issues, sensor limitations, navigation challenges, complexity in control systems, technological limitations, cost and complexity
Morar et al. [34]	2020	Computer vision-based indoor localization, static and mobile cameras, configuration stage, sensing devices, detection elements, localization methods, use of AI, benchmarks and datasets	Requirement of large training data, hardware demands, manual annotation, orientation information accuracy, environmental changes, complexity of data fusion, high computational cost
Alatise and Hancke [9]	2020	Sensor fusion, challenges in mobile robots, devices/sensors, estimation, navigation, object recognition, feature matching	Low stability of natural landmarks, navigation and path planning, localization issues, obstacle avoidance, sensor limitations, absolute localization techniques limitations
Zghair and Al-Araji [33]	2021	Control systems, navigation, localization and sensor fusion, energy-efficient path planning, heuristic and optimization algorithms, multi-robot systems	Issues in navigation and obstacle avoidance, sensor limitations, navigation in dynamic environments, energy efficiency, localization and mapping, multi-robot coordination
Tee and Han [28]	2021	LiDAR-based 2D SLAM, Robot Operating System (ROS), 2D LiDAR modules, IMUs, wheel encoders	GMapping limitations, issues with Rao-Blackwellized particle filter SLAM, Hector-SLAM's data issues, high computational cost for large environments, sensor and environment errors
Kalaitzakis et al. [29]	2021	Fiducial markers for pose estimation, ARTag, AprilTag, ArUco, STag markers, detection and computational performance, marker configurations, applications in Unmanned Aircraft Systems (UAS) and mobile robotics	Marker size performance, motion blur impact, distance sensitivity, computational cost of AprilTag, lighting conditions, colored markers at steep angles and distances
Niloy et al. [20]	2021	Indoor AMRs, design issues, control, SLAM, navigation, path planning, decision making, sensors, locomotion	Kalman filter with non-linear systems, high memory usage of semantic maps, sensor aliasing, long-term consistency, challenges of dynamic environments, computational cost and efficiency
Oliveira et al. [25]	2022	Computer vision, vision-based localization, challenges in localization, prospects, statistical analysis of keywords, multi-sensor fusion, artificial markers, Kalman filter, fish eye cameras, Kinect sensor, Transfer Learning, CNNs	Challenges in dynamic environments, dependence on specific hardware, need for good sensor calibration, high computational cost, lack of standardized accuracy measurement
Motroni et al. [26]	2022	Mobile robotics localization, RFID technology, algorithms and tools for localization, multi-sensor data fusion	RFID limitations, Received Signal Strength Indicator (RSSI) challenges
Panigrahi and Bisoy [27]	2022	Localization strategies, probabilistic localization, autonomous map building, RFID-based localization, SLAM, position and orientation errors analysis, evolutionary approaches	Low frequency and accuracy of absolute measurements, poor odometry from encoders, high computational cost of absolute measurements, challenges in dynamic environments, limitations of RFID, complexity of SLAM in unknown environments
Nahavandi et al. [35]	2022	Sensor types, mobile robot platforms, simulation tools, path planning, sensor fusion, obstacle avoidance, SLAM, navigation techniques, deep learning methods	Sensor efficiency issues, weather impact on LiDAR, limitations of aerial vehicles, computational constraints
Ding et al. [36]	2022	Applications of AMRs, perception technologies, robot communications, control mechanisms, localization systems, collaborative control, object detection	Perception accuracy, sensor fusion, intermittent communication, multi-robot issues, inaccurate maps, calibration of cameras, cost and dependency on 3D maps
Sousa et al. [30]	2023	Long-term SLAM, appearance invariance, environment dynamics, map size management, multi session strategies, parallel computing, timing efficiency	Limited in-depth analysis, exclusion of long-term autonomy challenges, bias in methodology

2.2.1 Technologies

A standard sensor definition is: “A sensor is a device that receives and responds to a signal or a stimulus” [37]. This definition is generic, as it can describe electronic and electromechanical devices, such as a current sensor or a gas pedal in a car, and biological devices, like the human ear. Therefore, a more specific definition for sensors in a robotics scope is “a device that produces an electrical signal in response to a stimulus from a physical system.” This electrical signal can be encoded in amplitude, polarity, frequency, phase, or digital code [37]. The data produced by the sensor represents information about the physical system.

Sensors are classified in two functional axes [32]:

- Proprioceptive or Exteroceptive.
- Active or passive.

On the one hand, proprioceptive sensors read internal changes in the robot, such as motors' speeds, acceleration, temperature, and battery level, among others. On the other hand, exteroceptive sensors read data external to the robot, i.e., the environment. Some examples are distance measurements, thermal readings, and light intensity, among others [9, 20, 32].

Regarding the second category, active sensors emit energy into the environment and measure its physical response. Some examples are distance sensors via triangulation or [Time of Flight \(ToF\)](#). A passive sensor can perceive energy when triggered by certain stimuli from either the internal or external surroundings, like the motion of a gyroscope [20].

In this reasoning, Table 2.2 classifies the most used sensors in mobile robotics according to their usage type and functional axes.

Building on the broad overview of sensors used in autonomous mobile robotics illustrated in Table 2.2, this section delves into the specific technologies most commonly used in robotics applications. This detailed exploration complements the general sensor classifications previously discussed, focusing specifically on their applications.

Tactile Sensors: These sensors detect objects at close range with or without direct contact. Capable of sensing forces and temperature changes, they enable precise robot interactions with objects, such as in manipulation tasks, and are crucial for determining the exact position of an object or detecting heat variations [20, 32, 33].

Force Torque Sensors: Employed to regulate the force applied during tasks like assembly, these sensors provide enhanced interaction capabilities [20, 32].

Pressure and Force Sensors: These convert the force exerted on materials into measurable electrical signals, proper in detailed interaction tasks [20].

Global Positioning System: [Global Positioning System \(GPS\)](#) receivers are indispensable for mobile robots that perform tasks in several locations, such as a rescue robot navigating toward a victim. These receivers calculate their precise longitude, latitude, and time by interpreting the time difference between the reception and broadcast of satellite signals. This capability is crucial for outdoor navigation and can achieve positioning accuracy within a few meters if the receiver

Table 2.2: Classification of sensors used in mobile robotics by usage type and functional axes [9, 38]

Usage classification	Sensors	PC or EC	A or P
Tactile/Proximity sensors	Contact switches, bumpers	EC	P
	Optical barriers	EC	A
	Proximity sensors	EC	A
Wheel/motor sensors	Brush encoders	PC	P
	Potentiometers	PC	P
	Synchros, resolvers	PC	A
	Optical encoders	PC	A
	Magnetic encoders	PC	A
	Inductive encoders	PC	A
	Capacitive encoders	PC	A
Heading sensors	Compass	EC	P
	Gyroscopes	PC	P
	Inclinometers	EC	A/P
Acceleration sensor	Accelerometer	PC	P
Ground beacons	GPS	EC	A
	Active optical or RF beacons	EC	A
	Active ultrasonic beacons	EC	A
	Reflective beacons	EC	A
Active ranging	Reflectivity sensors	EC	A
	Ultrasonic sensor	EC	A
	Laser rangefinder	EC	A
	Optical triangulation (1D)	EC	A
	Structured light (2D)	EC	A
Motion sensors	Doppler radar	EC	A
	Doppler sound	EC	A
Vision sensors	CCD/CMOS cameras	EC	P

PC: Proprioceptive; EC: Exteroceptive; A: Active; P: Passive;

can access signals from at least four satellites. However, [GPS](#) effectiveness is limited in environments without a clear line of sight to these satellites, such as in urban areas with tall buildings or mountainous regions, where signal blockage can occur. Additionally, [GPS](#) signals are unavailable indoors [39].

Differential GPS can enhance accuracy, which reduces localization errors by using correction signals from fixed beacons known for their precise locations on Earth. Despite its advantages, the quality of DGPS corrections diminishes as the distance from these reference stations increases [35].

Encoders: Including both linear and rotary types, encoders offer vital data on the position and speed of the robot's moving parts, essential for detailed motion control. They are used to measure distances or the speed the robot has driven, as well as counting the revolutions of each wheel. However, they are subject to errors such as wheel slippage, leading to inaccurate displacement estimates [20, 32, 33, 35].

Infrared and Ultrasonic Sensors: Infrared sensors detect heat energy emitted from objects and assist in motion detection and navigation. Ultrasonic sensors measure distances using high-frequency sound waves, commonly used in navigation and obstacle detection. These are active sensors emitting energy and then measuring the environmental response [20, 33, 35].

Sonar: Primarily used for object detection, sonar sensors are effective both on land and underwater, using changes in sound to determine distances [20, 32, 33].

Passive beacons: This type of beacon does not have its power source and can only have information coded or stored into it. Some examples are passive **RFID** tags, where the chip draws power from the **RFID** reader with an embedded antenna on the chip [40].

Fiducial Markers (FM) are unique patterns used to help robots and other systems determine their position and orientation in space, especially in situations where other localization methods like **GPS** or **SLAM** might not be available or accurate enough [29].

Active beacons: Active beacons are essential devices that enhance robot localization and navigation capabilities, particularly in environments where traditional methods like **GPS** are ineffective [41]. These devices emit detectable signals that enable robots with sensors to determine their spatial orientation and position accurately. Active beacons are especially crucial in indoor settings or areas with obstructions that disrupt satellite signals, making them indispensable in scenarios such as automated warehouses for robots navigating complex spaces. For example, in swarm robotics, a robotic navigator may project these beacons onto surfaces like ceilings to guide other robots [42], facilitating adaptable and efficient navigation and supporting tasks such as coordinated movements or precise positioning in cluttered or enclosed spaces.

RFID Tags: **RFID** tags are known for their effectiveness in context-aware applications and are classified as passive and active. The former, powered by the **RFID** reader's signal, contrasts with active tags that require batteries and can emit signals over longer ranges. Localization with **RFID** tags typically involves arranging them in a grid pattern to enable precise location detection by the robot equipped with a reader as it passes by [27].

IMUs: These inertial sensors are ubiquitous in mobile devices and navigation systems. They provide essential data for calculating the position and orientation of the robot without external references, often necessitating the fusion of data from multiple sensors to correct for drift and noise [9, 20, 32, 33, 35].

Laser Range Finders and Vision-Based Sensors: These sensors offer high-precision measurements and image processing capabilities vital for object detection and navigation. Stabilization technologies further enhance these systems to ensure accurate tracking and operation in dynamic environments [9, 20, 32, 33, 35].

Optical Flow Sensors: Employed in environments where weight and size are limiting factors, these sensors determine ground velocity by making surface textures and features visible [20].

RGB-D Sensors: Combining color and depth detection, **Red Green Blue - Depth (RGB-D)** sensors provide a comprehensive view of the environment, significantly enhancing object detection and navigation capabilities [9, 20, 33, 35].

Color-tracking sensors: Color-tracking sensors enable the detection and monitoring of color within a given environment [32].

Thermal Sensors: Thermal sensors enhance robotic navigation by overcoming the limitations of vision-based sensors in challenging environments. Advanced SLAM systems now integrate thermal imaging with inertial sensors to navigate effectively in low-light or obscured conditions, such as smoke or dust [43]. These systems employ deep learning to process thermal data, improving localization accuracy and the ability to detect environmental features.

Additionally, new exploration algorithms allow robots to simultaneously map environments and collect thermal data in a single pass. This integrated approach optimizes navigation paths, focusing on unvisited areas and significantly enhancing data collection efficiency in complex settings like buildings [44]. These innovations are crucial for increasing the operational capabilities of robots across diverse and challenging environments.

Landmarks: An effective strategy for enhancing the localization accuracy of autonomous mobile robots involves the use of landmarks, which can be broadly categorized into artificial and natural types, each serving unique purposes within the robot's navigation system [9]. In particular, the differences between artificial and natural landmarks are:

- **Artificial Landmarks:** These are specially designed objects or markers placed in the environment that robots detect using a variety of sensors, including laser, infrared, sonar, and vision systems. The uniqueness of these markers is critical, ensuring quick recognition and high reliability under different visual conditions. Artificial landmarks, such as stickers and Light-emitting diode (LED) arrays, are handy because they are easy to detect and describe, allowing for high-speed and precise localization. The simplicity and ease of setup with artificial landmarks make them widely used, though their deployment is generally limited to environments where the number of markers remains manageable [9].
- **Natural Landmarks:** Unlike artificial landmarks, natural landmarks consist of existing environmental features, such as corridors, doors, and ceiling lights, which are not explicitly installed for navigation but can be used by robots. Natural landmarks often require more sophisticated detection and matching algorithms, with commonly employed feature extraction techniques like Scale-Invariant Feature Transform (SIFT) and Speeded Up Robust Features (SURF). Despite their potential, natural landmarks are less frequently used in practical applications due to their lower stability and adaptability in changing environments [9].

Both categories of landmarks play a crucial role in AMRs' localization and navigation systems. By incorporating these features, robots can achieve more accurate and reliable navigation, especially in complex environments where other forms of sensory data might be less effective [9].

Integrating these varied sensor technologies enables autonomous mobile robots to perform complex tasks with high autonomy. Using sensors that measure internal states and external conditions, these robots can effectively navigate and interact with their environments, adapting to indoor and outdoor tasks.

2.2.2 Fundamental Problems of AMRs' Localization

Localization remains a critical challenge in mobile robotics, requiring accurate determination of the robot's position, with approaches varying based on the availability of environmental maps. Three primary types of localization problems have been identified [20]:

1. **Local Pose Tracking:** This requires the robot's previous position to monitor its current location, typically addressed using uni modal distribution such as a normal distribution to model the robot's belief.
2. **Global Localization:** The robot determines its position relative to a map, starting with a uniform distribution as its initial belief.
3. **Kidnapped Robot Problem:** Occurs when a robot is moved unknowingly to a new location on the map. The resolution of this problem is similar to that of global localization but depends on the robot recognizing its kidnapped state. The issue becomes complex when the robot is mistaken about its actual position.

To solve these problems, one of the most critical aspects that localization systems must have is integrating valid data that often comes in different formats to locate itself. For this reason, localization systems often employ sensor fusion techniques that aim to maximize the usability of the data over time and mitigate the noise. In this way, the next section will describe the most popular sensor fusion techniques in the [SOTA](#).

2.2.3 Sensor Fusion Techniques

Sensor fusion techniques are crucial for enhancing the performance of systems that depend on integrating multiple sensors. Combining data from several sources helps mitigate the inherent uncertainties and discrepancies associated with data from individual sensors. Adopting sensor fusion offers numerous benefits, as outlined in [9].

One significant advantage of sensor fusion is the reduction in uncertainty. This reduction is achieved by integrating data from several sources, compensating for individual sensors' limitations. This integration is particularly critical in applications such as robot localization, where errors in odometric measurements can be corrected using absolute positioning techniques. Additionally, sensor fusion enhances accuracy and reliability. The system maintains robustness using multiple sensor sources, providing reliable information even when partial sensor failure occurs. This multi-sensor approach ensures the system's performance remains high under several conditions. Another benefit is the extended spatial and temporal coverage. Sensor fusion enables broader coverage in both dimensions. For example, combining inertial sensors, such as accelerometers, with visual sensors, like cameras, can achieve comprehensive environmental monitoring that exceeds the capabilities of individual sensors.

Furthermore, sensor fusion leads to improved resolution. Combining independent measurements from different sensors results in higher resolution of computed values, surpassing the outcomes achievable with single sensor data. Finally, sensor fusion can reduce system complexity.

Preprocessing sensor data through fusion algorithms standardizes inputs to the controlling systems, simplifies the development of applications, and facilitates easier integration or replacement of sensors.

Thus, three fundamental approaches to integrating sensor data in systems, as described by [9], are outlined below:

- **Competitive/Redundant:** Sensors independently measure the same attribute, increasing system reliability and fault tolerance, thereby ensuring robustness in the face of individual sensor failures.
- **Complementary:** Different sensors provide unique, non-overlapping data, which, when combined, offer a complete view of the environment, particularly useful in complex or visually obscured settings.
- **Cooperative:** Multiple sensors contribute data to generate information that no single sensor could, such as creating three-dimensional images from two-dimensional inputs (stereoscopic vision) and enhancing depth perception but sensitive to individual sensor errors.

The limitations of single sensors have led to the development of various sensor fusion techniques, which can be combined to optimize accuracy and coverage. The primary approaches used in robotics and autonomous systems—Kalman Filter Localization, Markov Localization, and Monte-Carlo Localization—each have unique operational principles and are best suited for different scenarios:

- **Kalman Filter Localization:** The [Kalman Filter \(KF\)](#) is ideal for systems where the model dynamics and noise are well understood and can be accurately modeled as linear and Gaussian. This filter continuously updates its estimates by predicting the next state and then updating this prediction based on new measurements. Its effectiveness lies in its computational efficiency, making it suitable for real-time applications in linear settings. However, its limitation is in handling non-linear systems, which can be partly mitigated by its variants like the [Extended Kalman Filter \(EKF\)](#) and [Unscented Kalman Filter \(UKF\)](#) that handle non-linearities through linear approximations and advanced sampling techniques, respectively [9, 20].
- **Markov Localization:** This method uses a probabilistic framework to handle the localization problem under conditions of uncertainty and ambiguity. Unlike the [KF](#), Markov Localization does not require a prior understanding of the system's dynamics, making it more flexible. It is particularly effective in environments where the robot's initial position is unknown or if the robot needs to recover from being “kidnapped”. The technique updates its “belief” about the robot's position by integrating continuous sensor inputs, making it robust against dynamic environmental changes and sensor inaccuracies [20, 27].
- **Monte-Carlo Localization (Particle Filter):** This approach is distinct in using particles to represent the probability density function of the robot's state. Each particle represents a

possible state, with a weight that indicates its likelihood based on how well it predicts the observed measurements. This method is highly adaptable to non-linear and non-Gaussian problems, making it powerful in complex and dynamic environments where the assumptions required for Kalman filters cannot be satisfied. Monte-Carlo Localization can handle huge state spaces but at the cost of increased computational demand, especially as the number of particles grows [9, 20].

Each technique provides a different approach to the localization problem, with its applicability dependent on the specific requirements of the case being analyzed and the resources available.

2.2.4 Mapping

While mapping is not the primary focus of this work, understanding its basic concepts is crucial for foundational knowledge in this PhD research. Maps are integral in mobile robotics, aiding robots in navigating and interacting with their environments [45].

Map-based localization systems are generally preferred over map-less systems due to their robustness and reliability [20]. While map-less systems exist, they often come with significant limitations that can affect the performance and effectiveness of localization. Map-based approaches can represent static and dynamic features of the environment and enhance the robot's ability to operate effectively in complex settings.

Recent advancements have led to the development of sophisticated map structures, such as the 3D-Multi-Layer-Multi-Representation map proposed in [45]. These maps are designed with separate layers for static objects (using octomap techniques), dynamic objects (using bounding boxes), and even specific layers for human interactions, significantly improving navigation and task performance in environments with static and dynamic elements. Additionally, [46] introduced a novel multi-layer semantic map that facilitates object-oriented navigation. This map incorporates layers for occupancy, localization, and object pose estimation, enabling precise docking with specific objects. Such maps use advanced techniques like RTAB-Map and deep-learning models for real-time 3D object pose estimation, enhancing the robot's interaction capabilities within indoor environments.

Choosing the proper map representation involves considering several factors as stated by [20]:

1. The complexity of the map impacts the computational costs of localization, mapping, and navigation.
2. The map's precision must align with the required accuracy for AMRs.
3. The congruence of map features with the real-world dimensions of objects they represent.

Different map representations, such as metric, topological, semi-dense, dense, sparse, and semantic maps, each offer unique advantages and trade-offs tailored to specific applications and objectives as described by [20]. Current state-of-the-art mapping systems typically employ grid

maps for basic navigation tasks. However, advanced systems now integrate semantic information into multi-group maps, which allow for richer and more context-aware navigation [45, 46]. These innovations mark a significant step towards more adaptive and intelligent robotic navigation systems capable of handling the complexities of real-world environments.

2.2.5 Simultaneous Localization and Mapping

The **SLAM** problem formulates the issue of placing a mobile robot at an unknown location in an unknown environment and, from there, tests if it can build a reliable map of this environment while simultaneously estimating its position. A solution to this problem has been an aspiration to the robotics community as it would provide the tools to make a robot truthfully autonomous [47].

SLAM uses data from the robot's proprioceptive and exteroceptive sensors to estimate the robot's map and path. Some examples of data used, but not limited to, are odometry and environmental features detected by laser range finders, ultrasonic sensors, and cameras, among others [27]. In this way, **SLAM** can be categorized into two types: online **SLAM**, which only estimates the current pose and builds the map, and full slam, where it estimates the entire path alongside building the map [20].

Probabilistic **SLAM** was first introduced in the 90s, and according to [47], the **SLAM** problem is solved theoretically and conceptually. However, important issues, such as building and using perceptually rich maps as part of the **SLAM** algorithm, are to be addressed. In fact, since then, extensive research has been made in this area, especially in improving computational efficiency without harming accuracy, non-linearity, data association, and landmark characterization [48].

As stated by [20], the majority of **SLAM** algorithms are inspired by three models, **EKF-SLAM**, **Particle Filter (PF)-SLAM**, and **Graph-Based SLAM**. In this reasoning, Some recent work on **SLAM** algorithms is described below.

- **EKF-SLAM:**

According to [27], the **EKF-SLAM** was the first formulation to solve the **SLAM** problem, which was addressed by several papers [49–59]. Authors in [58], compared **EKF-SLAM** and Sigma Point Kalman Filter-**SLAM** (**SPKF-SLAM**) in simulation. The latter, also known as Unscented Kalman Filter **SLAM** (**UKF-SLAM**), had better results in state pose estimation and state pose covariance of the robot. According to the authors, since **SLAM** is a non-linear problem, **EKF-SLAM** introduces errors by using first-order Taylor expansion to approximate these non-linear models. This drawback was also studied and confirmed by [48, 60, 61]. In this way, researchers proposed other approaches to describe better the non-linear models [62–64].

- **UKF-SLAM:**

UKF-SLAM uses the **UKF** to address the shortcomings of **EKF-SLAM** by improving the estimation of non-linear mean and variance without relying on linear approximations. The unscented transformation, a core component of **UKF**, selects optimal sigma points around the mean state to represent the state distribution, thereby achieving more accurate state

predictions and corrections [27]. This technique provides enhanced accuracy with tighter uncertainty bounds and reduces errors compared to **EKF**-based methods. The approach is beneficial in complex environments where precision in state estimation is critical. Furthermore, the flexibility of the **UKF** approach in handling non-linear system dynamics is highlighted by its capability to maintain accurate state estimations even in challenging scenarios, as depicted in predictive comparisons with **EKF**, demonstrating its robustness [35]. Some examples of works using **UKF-SLAM** are [65–67].

- Particle filter **SLAM**:

According to [20], particle filters are acceptable for localization; however, as **SLAM** problems often have many features, the system becomes complex to handle. Thus, the Fast-SLAM approach addressed this issue with an adapted number of particles. Even so, this approach opened the path to grid-based improved versions of it, such as GridSLAM [68], and GMapping [69–72].

- Graph-based **SLAM**:

Pose Graph Optimization (PGO) has become increasingly popular over the last decade, primarily within the domain of **SLAM** [20]. **PGO** aims to estimate the robot's trajectory through relative pose measurements, constructing a pose graph that interlinks robot poses with line segments. This optimization challenge is generally treated as a non-convex problem, where the goal is to find a maximum likelihood approximation of relative poses [20]. Iterative methods such as Gradient Descent [73, 74] and Gauss-Newton [75, 76] are commonly used to converge on a solution. Additionally, some strategies employ duality theory to assess whether **SLAM** estimates are globally optimal [77, 78], and convex relaxations have been developed to handle **PGO** with outliers [79]. Furthermore, a novel multi-robot localization technique that uses a pose graph and does not require prior knowledge of initial relative poses has been introduced [80]. In addition, Hector-SLAM implements scan-matching with **LiDAR** data to facilitate odometry-free localization and mapping, optimizing robot poses using the Gauss-Newton method to reduce scan discrepancies with existing maps [28]. Ideal for environments where traditional odometry is infeasible, Hector-SLAM demands high-quality **LiDAR** to maximize performance and minimize mapping errors [28]. Additionally, developed by Google in 2016, Google Cartographer is a robust graph-based **SLAM** technique that efficiently processes extensive robot trajectory data through sophisticated graph optimization. It operates in two phases: Local **SLAM**, which performs immediate scan-matching and pose estimation, and Global **SLAM**, which optimizes these poses across the map to ensure consistency and accuracy, crucial for dynamic environments [28].

2.2.6 Visual SLAM

SLAM research initially focused on 2D mapping, but with technological advancements, the need for more complex 3D mapping increased, especially for **AMRs** tasked with complex operations

[20]. **vSLAM** uses cameras exclusively as its sensory method and includes variants such as monocular, stereo, and **RGB-D** cameras [81].

One common approach in **vSLAM** is feature-based methods, using image features to ascertain scene geometry and camera pose. These methods, including **EKF**-based visual-inertial odometry and key frame-based systems, intuitively extract features using methods like **Oriented FAST and Rotated BRIEF (ORB)** [82], **SURF** [83], and **SIFT** [84]. For instance, ORB-SLAM, which uses **ORB** features, has evolved through several versions to support different camera models and configurations, operating in diverse environments without requiring a GPU [35]. However, feature-based approaches can be limited by the specific features they can detect and often struggle with man-made environments [35].

Direct methods represent another **vSLAM** strategy, focusing on using all available photometric data from images, thus avoiding the potential information loss inherent in feature-based methods. Large-Scale Direct Monocular SLAM exemplifies this approach by efficiently creating large-scale 3D maps in real-time [35]. Direct Sparse Mapping, another example, leverages photo metric bundle adjustment to enhance map reuse and data association over more extended periods [35].

Moreover, the semi-direct method combines the strengths of both feature-based and direct methods. It aims to optimize initial motion estimation by minimizing photometric errors non-linearly, providing a balanced solution that leverages the best aspects of both methodologies [85].

Overall, **vSLAM** techniques are crucial for developing 3D mapping capabilities **AMRs** required for navigating and performing tasks in complex environments [20]. Finally, authors in [14] conducted an extensive survey.

2.2.7 Map Matching Algorithms

There are works in mobile robotics for localization that use map-matching algorithms to perform the update stage in sensor fusion algorithms. In simple terms, the robot has a pre-built map of the environment, and it tries to estimate its position on the map by making real-time scans of the environment and building another map where it tries, in some way, to get the best match between the maps. There are several algorithms in this regard, some of which will be described here.

Perfect Match Algorithm and Enhancements

The **Perfect Match (PM)** algorithm, initially developed for the RoboCup Midsized league, represents a substantial advancement in localization technologies. It was demonstrated to be more precise and efficient than traditional particle filtering by optimizing error terms and integrating sensor data more effectively [86]. It has outperformed other map-matching algorithms' accuracy and computational speed, as evidenced in several comparative studies [87]. **PM** has been successfully extended for industrial uses and integrated with the **ROS**, proving its utility in environments without beacons. It incorporates an **EKF** for improved sensor fusion [88] and has been enhanced with **Particle Swarm Optimization (PSO)** to address the challenges of dynamic environments, enhancing both robustness and precision under uncertainty [89]. These improvements address the

demand for high accuracy, robustness, and computational efficiency in static and dynamic settings. Initially described in [90], **PM** computes vehicle pose using 2D distance data from onboard **LiDARs** or **Charged-Coupled Device (CCD)** cameras. It involves three key processes: matching error and gradient computation, optimization via **Resilient Backpropagation (RPROP)**, and covariance estimation [91]. Subsequent research introduced a 3D adaptation of **PM**, offering faster and more computationally efficient localization compared to 2D and 3D **SLAM** and **Iterative Closest Point (ICP)** algorithms [92].

Further robustness and precision tests with real robots confirmed these findings [93]. An additional variant for small robots, integrating **PM** with **EKF** and **PSO**, demonstrated enhanced performance in the Robot@Factory competition [94]. Finally, comparative analyses affirm **PM**'s computational superiority over other leading algorithms like **ICP** and **Normal Distribution Transform (NDT)**, although all showed comparable overall performance [91].

Iterative Closest Point

The **ICP** algorithm is a widely used map-matching method that optimizes the Euclidean distance between sensor data (input) and a predefined reference map (model). This algorithm operates by alternating between two primary phases: the matching stage, where data points are aligned with the nearest points on the model, and the transformation stage, which adjusts the data to fit the model best. This iterative process continues until the algorithm converges to a minimum error solution [91].

Normal Distributions Transform

Initially developed for 2D scan registration [95], the **NDT** was later adapted for 3D scan matching applications [96, 97]. The **NDT** operates by converting the scan data into a probabilistic representation using local probability density functions derived from a voxel grid. Each voxel grid is formed from sets of reference points that define multiple cells. The scan transformation is then refined through an optimization process where a scoring mechanism is calculated, and Newton's Method is applied to minimize this score, enhancing the alignment accuracy [91].

2.2.8 Other Approaches

This section was reserved for popular localization techniques that are not exactly considered a category of localization itself, but, however, absolute positioning techniques applied in the greater scope of sensor fusion and **SLAM**.

Evolutionary Methods

Authors in [27] researched evolutionary methods applied in localization. According to them, several authors applied evolutionary techniques to classic localization methods (**EKF**, **PF**, among others) to perform the robot's localization. Approaches such as **PSO** were used by [94, 98–103].

A few works were proposed using Genetic Algorithm [104–106], and Differential Evolution [100, 107–109].

Fiducial Markers

Fiducial markers, with their distinctive visual features and specific patterns, are emerging in robotics, serving as reliable landmarks for pose estimation across several applications. Their well-characterized size and shape and patterns that embed specific encoding avoid misdetections and enhance the accuracy and robustness of the localization system of a robot [29]. Especially in tasks such as localization and mapping, fiducial markers have shown their worth by significantly enhancing the precision and adaptability of robotic systems in diverse settings, including complex environments like automated warehouses and factory floors. This wide usage underscores their contribution to the progression of robotic technologies, highlighting their broad applications and the importance of integrating these visual markers with inertial sensors to improve the performance and reliability of [AMRs](#).

The ArTuga project introduces advancements in aerial robotics, specifically for precise unmanned aerial vehicles' vertical takeoffs and landings in challenging environments. It features an advanced multi-modal fiducial marker, improving landing precision under diverse lighting and weather conditions, essential for continuous operations [110]. In addition, developing an object pose estimation technique using pentagonal fiducial markers for enhanced detectability and accuracy presents forward-thinking to marker design [111]. MechaTag, a mechanical fiducial marker system for harsh industrial settings, is introduced by Digiacoio et al. [112], showcasing high reliability in marker recognition. Zhang et al. [113] introduced a visual-marker-inertial fusion system using sliding window optimization to tackle the challenges of visual-inertial odometry, demonstrating enhanced global localization accuracy, particularly in weak texture conditions and highlighting its application potential in [I4.0](#) settings. In addition, Popović et al. [114] and Alabbas et al. [115] have both made strides in human-robot collaboration in warehouse settings, focusing on reliable human localization for safety and operational efficiency and introducing innovative solutions such as wearable cameras, fiducial markers, and haptic feedback wearable robots.

Additionally, innovations in autonomous logistics, such as using fiducial markers for real-time pallet detection and tracking, highlight the practical implications of marker systems in dynamic industrial environments [116]. Furthermore, authors in [117] contributed to [AMR](#) localization in industrial settings through the [RobotAtFactory 4.0 \(RAF\)](#) competition by presenting an indoor localization system that uses an [EKF](#) and ArUco markers. Separately, Romero et al. [118] focused on a different aspect of [AMR](#) technology by developing a validation strategy for target-based vision tracking systems. Zhang et al. [119] developed a precise visual positioning method for agricultural robots in greenhouses, using fiducial markers to enhance accuracy and minimize re-projection errors. Their technique significantly improves positioning precision compared to traditional methods. Also, Zhang et al. [120] tackles autonomous navigation challenges in indoor environments with a graph-based [SLAM](#) method assisted by visual markers, achieving enhanced positioning accuracy and robustness. Huang et al. [121] and Yang and Yang [122] have explored

optimization techniques for visual localization and multi-sensor fusion positioning systems, showcasing significant improvements in localization accuracy and system robustness, even in dynamic environments. Also, authors in [123] introduced an optimized marker placement algorithm that significantly improves localization accuracy by up to 20 % through strategic marker placement. Analytical models for estimating pose estimate variances from fiducial markers are presented by Adámek et al. [124]. Huang et al. [125] provides a comprehensive review of indoor positioning systems for mobile robots, detailing current trends and developments where fiducial markers have their place.

Furthermore, a comprehensive evaluation of fiducial marker systems reveals their diverse applicability across robotics applications, shown in [126]. Machine learning approaches for robot localization using fiducial markers are introduced by authors in [127], achieving millimeter-scale error in pose estimation. Kim et al. [128] and Ekici et al. [129] present novel solutions for localization and object tracking in dynamic environments, using multi-layered 3D scan-matching, virtual fiducial markers, and robust object pose tracking algorithms. In Micro Aerial Vehicle localization, Kirsch et al. [130] uses fiducial markers for high-precision map creation and navigation in indoor environments.

RFID

A plethora of methodologies employing RFID technology for localization tasks have been reviewed and documented by Panigrahi and Bisoy [27]; some of the most recent ones are the research works [131–136]. These methods leverage several dimensions of RFID system outputs to pinpoint the precise location of mobile robots. The techniques for localization are the probability of detection, the RSSI, and the phase of tag replies [26]. Each approach offers unique advantages and suffers from distinct limitations.

For instance, the probability of detection method evaluates the likelihood of a reader detecting an RFID tag at several locations to estimate the robot's location [26]. Similarly, RSSI-based techniques measure the signal strength from multiple RFID tags to triangulate the robot's position [26]. In contrast, methods analyzing the phase of tag replies use phase differences to determine the robot's coordinates [26].

RFID technologies are praised for their low operational cost and ease of deployment, making them a practical choice for several environments and a promising area for continued research and development [26]. Recent studies have significantly expanded the understanding and application of RFID in localization, demonstrating its potential in several context-aware applications [27]. However, challenges such as detection accuracy, environmental sensitivities, and the necessity for meticulous deployment planning underscore the limitations inherent in these systems [26].

2.3 State of the Art in Object Detection Algorithms

The research landscape in object detection using deep learning techniques is extensive and intricate. Numerous review papers have explored this domain, each contributing unique perspectives

and insights. As highlighted by [137], within the context of mobile robotics, deep learning harnesses the power of large-scale neural networks to transform raw, high-dimensional data from diverse sources into formats that robotic systems can process and understand. Notably, real-time processing speed and accuracy are critical in this application.

CNNs, which mimic human visual processing, are important in advancing object detection technologies. **CNNs** analyses visual inputs, assigning weights and biases to different input aspects, enabling them to distinguish between distinct objects. These networks are structured similarly to the human visual cortex, with neurons that respond to specific visual field areas [138]. This structure includes several layers, such as convolutional layers, pooling layers, and fully connected layers, which collectively work to extract features, reduce data dimensionality, and perform classification tasks [139].

The evolution of **CNN** architectures has significantly boosted their efficiency and accuracy. Advances such as developing the **Region-Based Convolutional Neural Network (R-CNN)** and its faster variants have simplified the generation of region proposals and the refinement of bounding box coordinates, enhancing the speed of object detection [138]. These improvements marked a substantial progression. These traditional image processing techniques relied heavily on manual feature design and needed to be more capable of handling variations in object scale, orientation, and position [140].

The inception of the ImageNet competition and the introduction of models like AlexNet have been crucial in demonstrating the superior capabilities of **CNNs** over traditional methods, which often involved labor-intensive feature engineering and less effective classifiers such as **Support Vector Machine (SVMs)** [140]. Integrating classification and regression tasks within a single network by modern **CNNs** simplifies the process. It enhances overall performance by learning to identify and classify objects directly from images, eliminating the need for separate feature extraction and classification steps [139].

The state of the art of object detection can be separated into two periods, namely before and after the introduction of deep learning for object detection in 2013, where AlexNet served as the watershed between traditional and **CNNs**-based methods [141]. Figure 2.4 illustrates a classification diagram of object detection algorithms.

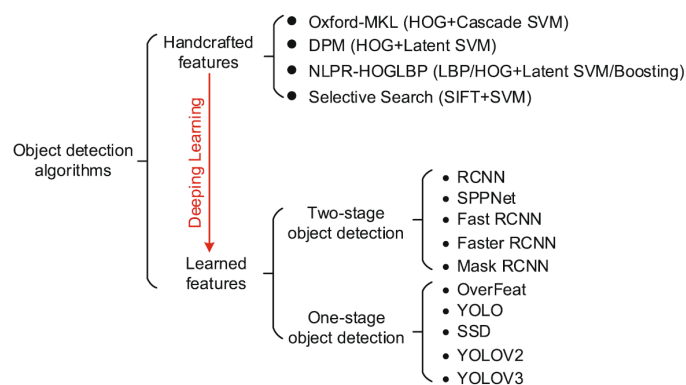


Figure 2.4: Classification of object detection algorithms [141]

A complete story of object detection's state of the art can be seen in [141]. In this reasoning, Viola and Jones [142] proposed a novel machine-learning approach for object detection capable of rapidly processing images, running at 15 frames per second. Also, Dalal and Triggs [143] showed that **Histograms of Oriented Gradient (HOG)** descriptors outperformed existing feature sets for human detection. They studied the influence of each phase of the computation on performance. In addition, Felzenszwalb et al. [144] introduced a general framework that trains **Latent Support Vector Machines (LSVMs)**, which is employed in their system and is based on multi-scale deformable models [144].

After the introduction of **CNNs**, the approaches for object detection can be divided into two categories, two-stage and one-stage [141]. According to [145], two-stage detectors are typically more accurate, and one-stage detectors are usually faster. In addition, for two-stage detectors, the first stage is usually used to generate proposals classified during the second stage [145]. According to [141], the papers considered milestones for two-stage detectors are displayed in Figure 2.5.

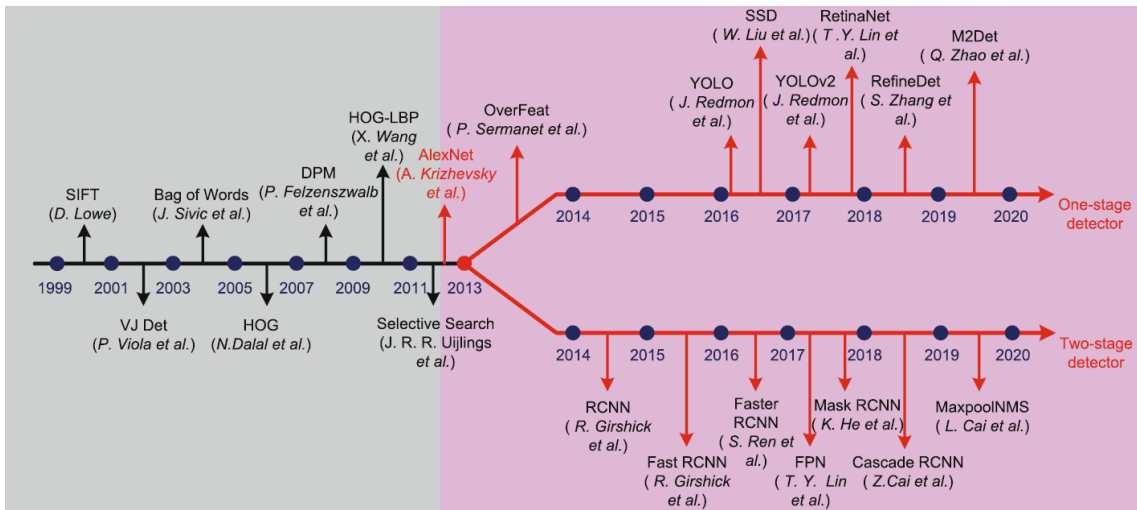


Figure 2.5: Milestones of object detection evolution [141]

The following list expands on these methods, incorporating updates for 2024 from [139]:

- Region-based Convolutional Neural Networks (**R-CNN**) [146].
- Fast **R-CNN** [147].
- Faster **R-CNN** [148].
- Feature Pyramid Networks (FPN) [149].
- Mask **R-CNN** [150].
- Cascade **R-CNN** [151].
- Maxpool Non-maximum Suppression (MaxpoolNMS) [152].
- Granulated Region-based Convolutional Neural Networks (G-RCNN) [153].

According to [145], **R-CNN** popularized the two-stage approach. The algorithm generates category-independent region proposals using Selective Search, extracting features using a large **CNN**. Finally, they are classified using a set of linear **SVMs**. Fast **R-CNN** extended **R-CNN** by using a convolutional implementation of the sliding window approach to classify the regions [145]. Faster **R-CNN** improved Fast **R-CNN** by employing a **CNN** to propose regions instead of selective search [145]. The authors in [149] proposed the architecture entitled **Feature Pyramid Networks (FPN)**, which implements feature pyramids in the Faster **R-CNN** system. They stated that they achieved state-of-the-art single-model results surpassing the ones at that time. Mask **R-CNN** expands Faster **R-CNN** by adding a branch for predicting an object mask in parallel with the existing branch for bounding box recognition [150]. Cascade **R-CNN** [151] addressed problems regarding the low and high **Intersection over Union (IoU)** threshold trade-offs. MaxpoolNMS was proposed in [152] to eliminate the bottlenecks in two-stage detectors, achieving a 20 times faster speed than GreedyNMS. Finally, by incorporating the concept of granulation in a deep **CNN**, G-RCNN improves Fast **R-CNN**, and Faster **R-CNN** [153].

The following compilation presents significant one-stage object detection algorithms, incorporating works cited in [141] and [139], with updates through 2024:

- OverFeat [154].
- **You Only Look Once (YOLO)** [155].
- Single Shot MultiBox Detector (SSD) [156].
- RetinaNet [157].
- RefineDet [158].
- M2Det [159].
- YOLOv3 [160].
- YOLOv4 [161].
- You Only Learn One Representation (YOLOR) [162].
- YOLOX ([163]).
- YOLOv7 ([164]).

Authors in [154] proposed the Overfeat architecture, which integrated object classification and location into one network architecture [141]. **YOLO** introduced a new approach to object detection in 2016. In contrast to previous works that use classifiers to perform detection, **YOLO** frames the object detection as a regression problem to spatially separated bounding boxes and associated class probabilities [155]. In other words, the algorithm can predict bounding boxes and class probabilities from full images in one stage. According to the authors, their architecture

is high-speed, reaching 45 **frames per second (FPS)** on a Titan X. In addition, a smaller version, called Fast **YOLO**, reaches 155 **FPS** while still achieving double **mean average precision (mAP)** of other real-time detectors [155]. In addition, the authors admit that even though their approach makes more localization errors, their approach is less likely to predict false positives. Moreover, it is possible to see that their approach outperforms other detection methods, such as **R-CNN**, in certain situations [155].

In comparison, **Single Shot Detector (SSD)** uses convolutional layers instead of fully connected layers that **YOLO** uses [145]. RetinaNet addressed dense detectors' class imbalance problem by reshaping the standard cross-entropy loss [157]. RefineDet, a single-stage detector with better accuracy than two-stage detectors that maintains the efficiency of single-stage approaches, was proposed by [158]. Authors in [159] proposed a multi-level **FPN** to construct more effective pyramids for object detection. Finally, some improvements were made in **YOLO** over the years [160–162]. According to the authors, YOLOv3 achieves a similar **mAP** as **SSD** three times faster. Finally, YOLOv4, compared with YOLOv3, achieved an improvement of 10% in the average precision and 12% in **FPS**. The **YOLO** series continues to advance with its several versions, each tailored to specific real-time object detection challenges. YOLOX emphasizes performance enhancements through new techniques and optimizations, aiming for versatility across different detection tasks, thereby suiting a range of applications [139]. Meanwhile, YOLO-R focuses on robustness, integrating advanced features to perform reliably under challenging conditions, such as low-light environments or when objects are occluded, proving essential for critical detection needs [139]. On another front, YOLOv7, as an enhancement over its predecessors, offers improved accuracy and speed, targeting applications needing swift and precise detections like autonomous driving and surveillance systems [139].

Overall, **CNNs** have revolutionized the field of computer vision by automating feature extraction and providing a robust framework for the dynamic and efficient recognition of objects in several applications [138]. To end this topic of the literature review, it was found that there is an abundance of review papers on the subject. Therefore, there is no purpose in delving more into the subject since several authors comprehensively address it. Table 2.3 displays review papers researched over the last five years and describes which topics each discusses.

Table 2.3: Object detection review papers from the last five years

Reference	Year	Topics Covered
Xiao et al. [141]	2020	Deep learning object detection methods, datasets, backbone networks, loss functions. Image datasets: ImageNet, PASCAL VOC, Open Images, MS COCO. Incremental learning, zero-shot learning, feature similarity, semantic information
Nazir and Wani [165]	2023	YOLO variants from V1 to V7, architectural modifications, evaluation metrics
Bhaidasna and Bhaidasna [166]	2023	Advancements using ML, datasets like COCO, PASCAL VOC
Amjoud and Amrouch [140]	2023	CNNs, anchor-based and anchor-free detectors, transformer-based detectors
Bhumbla et al. [139]	2023	Object detection techniques, evolution in deep learning, single and two-stage models
Abdul-Khalil et al. [167]	2023	AI and object detection for autonomous mobile robots
Setia et al. [138]	2023	CNNs, YOLO, R-CNN, limitations and improvements in detection
Kaur and Singh [168]	2023	Frameworks, architectures, evaluation metrics, applications in self-driving cars

2.4 AI in Robotics Applied to Localization Systems

Deep learning has been incorporated into several subprocesses within the context of AMRs, as highlighted in comprehensive surveys [137, 167]. These studies reveal that deep learning techniques enhance several aspects of mobile robotics, including perception, localization, control, exploration, navigation, and autonomous driving. Specifically, this review concentrates on the application of deep learning in localization. Recent breakthroughs in leveraging AI for autonomous mobile robotics localization systems have markedly enhanced indoor and outdoor navigation capabilities. Such progress is encapsulated in pioneering research efforts ranging from pure deep learning models to sophisticated hybrid learning systems. Starting with the work documented in [169], the researchers developed a robust system leveraging the ubiquitous Wi-Fi infrastructure for indoor localization. Using a robot-aided data collection strategy, they successfully addressed the scarcity of annotated data. Through a hybrid learning approach combining supervised, unsupervised, and semi-supervised learning strategies, their models could incrementally improve, enhancing localization and navigation capabilities in complex indoor settings. Similarly, [170] introduced a system that uses visual markers for localization, significantly enhancing accuracy by replacing traditional methods with deep learning techniques. This innovation proved particularly beneficial when GPS signals were unavailable, facilitating precise navigation over extended distances through strategically placed markers. Adding a different layer to localization methodologies, [171] introduced a self-supervised learning framework using Ultra-wideband (UWB) tech-

nology for the 3D localization of small flying robots. This approach was supported by a novel simulation pipeline that improved the model's generalizability in real-world scenarios, demonstrating its efficacy with Crazyflie2 quad rotors equipped with an AI deck. Further, [172] emphasized the fusion of sensory data from LiDAR and odometry through a neural network to achieve real-time and highly accurate indoor localization. This work highlighted the important role of deep learning in fusing several data types to reduce localization errors significantly. The research in [173] explored the concept of "localizability" using a neural network model comprising CNN, Recurrent Neural Network (RNN), and Multilayer Perceptron (MLP) layers. This model excelled in estimating robots' location, facilitating more reliable navigation by comparing real-time locations with a pre-existing map. Moreover, [174] developed a system that used visual information and sensor data fusion to enhance vehicle localization accuracy. Integrating a supervised neural network for map-matching exemplified the significant potential of deep learning in complex data association tasks, ensuring high precision in position estimation. Lastly, the focus on vSLAM in [175] and the three-dimensional indoor positioning system developed in [176] illustrate the broadening scope of deep learning applications in mobile robotics. These studies underscore the shift towards integrating deep learning into SLAM and the successful deployment of YOLOv5 for efficient object detection in real-time applications.

These papers reveal a trend towards more integrated, robust, and accurate localization systems in autonomous mobile robotics, driven by deep learning innovations that handle several environmental complexities and operational demands.

When deep learning is applied in SLAM/vSLAM methodologies, it is often characterized as "Robust", "Dynamic", or "Semantic" [11, 14, 15]. Given that SLAM and vSLAM fundamentally presupposes a static environment, their performance is typically limited in dynamic contexts. Consequently, methods incorporating AI to detect and either track or disregard dynamic entities are labeled accordingly, enhancing localization accuracy by identifying and excluding such variables. Innovations in this field include multi-task deep neural networks for outlier rejection [177], sophisticated geometric techniques in Accurate and Robust Dynamic SLAM (ARD-SLAM) [178], and the integration of semantic detection with depth segmentation in enhancements to ORB-SLAM2 [179]. Additionally, notable advancements are probabilistic models for dynamic detection [180], semantic RGB-D SLAM for high-precision localization [181], and semantic segmentation for motion estimation in rigid objects [11]. Such improvements demonstrate a trend toward more adaptive SLAM systems. Table 2.4 summarizes the primary references, outlining their contributions and the hardware used, thus defining the state-of-the-art in this domain. The summary was researched using the criteria of being recent (last 5 years, focusing on the most recent side of this range). The following keywords were searched in different academic research databases: "Robust", "Dynamic", "Semantic" with "Indoor Localization", "SLAM", "vSLAM" and "Robot Localization".

Table 2.4: State of the art in Dynamic/Robust/Semantic SLAM approaches

Reference	Year	Contributions	Hardware Used
Cebollada et al. [187]	2019	Examines global appearance descriptors for robot localization, comparing traditional analytic methods (HOG, gist) with deep learning techniques (auto-encoders, CNNs), highlighting the enhanced performance of the latter in visual tasks.	GTX 1080 Ti
Henein et al. [184]	2020	Introduces a novel dynamic SLAM algorithm using semantic segmentation for real-time velocity estimation without 3D model constraints, alongside a feature-based approach and joint optimization for accurate motion estimation and mapping.	Not specified
Rong et al. [190]	2020	Enhances RGB-D SLAM robustness in dynamic environments by integrating YOLOv4 with ORB-SLAM2 for dynamic object removal, improving place recognition and loop closing with semantic and depth data.	Intel Xeon CPU E5-1630v3, 3.7 GHz
Ran et al. [197]	2021	RS-SLAM uses semantic segmentation with Bayesian updates for precise dynamic object detection and movable object identification, contributing to high accuracy localization and real-time 3D semantic OctoMap construction in static environments.	i5-7500 CPU, GTX 1060
Li et al. [188]	2021	Proposes online image inpainting using deep neural networks to improve AMR visual localization by removing pedestrian features, supported by field tests validating over 50% correction in erroneous predictions using YOLOv4 and SN-PatchGAN.	i7 9700 3GHz, GTX 1080 Ti, 64 GB RAM
Yang et al. [189]	2021	Introduces a CNN-based image re-localization method for kidnapped indoor robots, featuring novel state parameterization for planar motion and lightweight residual blocks for efficient pose regression.	GTX 1080, Intel i7 7700 CPU
Cai and Wu [196]	2022	This paper leverages semantic segmentation and epipolar geometry to filter dynamic points based on prior knowledge, greatly improving localization accuracy by retaining potential dynamic points for enhanced tracking in highly dynamic environments.	Not specified
Cheng et al. [183]	2023	This paper presents a semantic RGB-D SLAM system that incorporates object detection, semantic processing, and geometry-based ground mask extraction to improve mapping accuracy in dynamic scenes.	i3 CPU, GTX 1660s GPU
Nam et al. [186]	2023	iNAV-drlSLAM integrates advanced path planning algorithms (ACO, DQN, TD3, PPO) with SLAM to address the challenges of dynamic robot navigation in varied environments.	Not specified
Zeng et al. [192]	2023	Introduces a visual SLAM system using multi-task deep neural networks focused on dynamic region detection and removal, which improves key point extraction and outlier rejection, incorporating semantic information for enhanced dynamic environment performance.	Not specified
Yin et al. [193]	2023	Dynam-SLAM presents a robust stereo visual-inertial system that uses loose and tight coupling of scene flow and IMU data for dynamic feature detection and optimization, significantly enhancing SLAM accuracy and robustness in dynamic environments.	AGX Xavier
He et al. [194]	2023	Describes OVD-SLAM, an online visual SLAM without predefined dynamic labels, combining semantic, depth, and optical flow to distinguish between foreground and background, and restoring static points on moving objects to improve SLAM accuracy.	Notebooks with i7 CPU, Tesla M40, RTX 2060, RTX 3060 GPUs
Zhang et al. [195]	2023	Proposes a visual SLAM system that integrates moving object trajectory tracking and prediction for accurate localization and mapping, supported by comprehensive evaluations on simulation and real-world KITTI datasets.	AMD 5600X CPU, 2080ti GPU
Zhang and Li [15]	2023	Offers a real-time semantic vSLAM system for dynamic environments, using object detection and motion consistency checks to pinpoint outliers, achieving superior performance on indoor and outdoor datasets.	Ryzen 7 5800h CPU, RTX 3070 GPU
Heiß et al. [198]	2023	PredictiveSLAM, an extension to ORB-SLAM2, improves SLAM by extracting features from regions of interest designated for detecting humans, applying static and moving object masking methods to enhance tracking accuracy.	Jetson Xavier NX
Mahmoud and Penkovskiy [199]	2023	Employs a geometric technique using KMeans clustering and SegNet for semantic segmentation to filter features on moving objects, enhancing SLAM accuracy for wheeled robots in dynamic environments with encoder measurements for bundle adjustment.	i7 CPU, RTX 2060
Zhang et al. [182]	2024	The study introduces a probability-based dynamic feature detection using Gaussian modeling and Kolmogorov-Smirnov tests, coupled with Bayesian model-based data association to enhance camera localization in dynamic environments.	Core i9-12900kf and NVIDIA RTX A5000
Li et al. [185]	2024	DDN-SLAM enhances dynamic SLAM accuracy by integrating semantic features, foreground segmentation, and a novel scene representation using NeRF. This method achieves a significant reduction in Average Trajectory Error.	i7 12700k CPU, RTX 3090ti GPU

Continued on next page

Table 2.4 – continued from previous page

Reference	Year	Contributions	Hardware Used
Ul Islam et al. [191]	2024	ARD-SLAM emphasizes trajectory error minimization in dynamic scenes, using instance segmentation networks for accurate dynamic element filtering and camera pose estimation in SLAM applications.	Consumer GPU

In conclusion, the preceding section has delineated how deep learning-driven localization methods have significantly enhanced autonomous mobile robotics navigation capabilities. While these advancements demonstrate considerable progress in accuracy and robustness, they remain dependent on high-cost, energy-consuming hardware platforms. This reliance on sophisticated systems creates barriers to widespread adoption, particularly in resource-constrained environments. Future research should focus on developing cost-effective, energy-efficient solutions that maintain current performance standards to enable broader implementation of these technologies.

2.5 Localization in Indoor Environments

In addressing the state-of-the-art localization in autonomous mobile robotics within indoor environments, significant challenges persist that complicate the practical implementation of current technologies. One of the most critical hurdles is the strong assumption of a static environment, a premise upon which many contemporary systems are built [14]. This assumption leads to challenges in dynamic environments where moving objects or changes in the environment can severely disrupt localization accuracy.

While research aims to mitigate these limitations by tracking and detecting outliers to refine localization [14], these solutions often rely on expensive, high-performance hardware that is not scalable or economically viable for widespread deployment [34, 36]. For instance, systems that use sophisticated sensors inherently drive up costs and complexity, limiting their accessibility and application to specialized or well-funded environments [32].

Further complicating these implementations are the computational demands these systems impose. Many discussed methods require extensive computational resources for operation and during the training phases of deep learning models [34]. For instance, the high memory usage of semantic maps, crucial for navigating complex environments dynamically, poses a substantial challenge [20]. These maps, which provide detailed environmental information, demand extensive memory, rendering them less effective for real-time applications and contributing significantly to the computational burden [20]. The mathematical modeling and control of mobile robots required to manage these maps and execute precise localization tasks further increase the overall system complexity and cost [20]. This is particularly challenging when deploying vSLAM methods on embedded platforms with constrained processing capabilities, which requires balancing real-time execution with high accuracy [35].

Additionally, the lack of standardized, large-scale, and accurately annotated 3D training datasets hampers the development and performance of these systems [34]. The requirement for manual annotation of training data is both labor-intensive and costly, further constraining the practical deployment of sophisticated models. Moreover, the calibration of camera parameters and the fusion

of data from multiple sensors add layers of complexity and potential points of failure, which can degrade system performance in unpredictable ways [25, 36].

The state-of-the-art, as presented, demonstrates a trend toward more integrated and robust solutions for indoor localization. However, the reliance on high-power and processing-intensive hardware revealed in the survey underlines the dichotomy between technological advancement and practical implementation. Thus, the challenge lies in developing methods that break away from the static environment assumption and in making these solutions economically accessible and energy-efficient to ensure they are not confined to niche applications but can be scaled for broader, real-world applications.

2.6 PhD Motivation & Future Directions

Robust indoor localization is critical for mobile robots' autonomy as they navigate complex, dynamic environments. Despite significant advancements in the field, the predominant models, particularly within **SLAM** technologies, have traditionally hinged on the assumption of static environments. This assumption significantly curtails their effectiveness in real-world applications where environments are inherently dynamic [11, 12].

While advancing in technological sophistication, the existing systems often require expensive, high-performance, and high-energy-consuming hardware to achieve the desired accuracy and robustness, which is not feasible for scalable, economically viable applications due to costs and autonomy. Particularly, the **vSLAM** that employ extensive sensor suites and computational resources exemplify this challenge. They struggle in dynamically changing settings due to the difficulty distinguishing static from dynamic elements [14].

This work aims to address these shortcomings by advancing the **PM** algorithm, known for being the best among others in the **SOTA** in map-matching tasks, to operate effectively under the dynamic conditions frequently encountered in practical scenarios. The proposed **Robust Perfect Match (RoPM)** enhances the **PM** by incorporating **AI**-enhanced techniques to mitigate the impacts of the static world assumption, thereby improving both the accuracy and robustness of localization in dynamic environments.

The **RoPM** aims to address the gap in the literature and limitations of high-cost, high-power localization systems by adopting a more cost-effective, low-resource approach. This involves integrating lightweight, efficient deep-learning models capable of real-time object detection. These are crucial for maintaining high-performance localization without the extensive computational overhead [16].

This innovative approach aligns with broader real-world applications' economic and practical demands. It establishes a new benchmark for indoor localization systems that effectively balance cost-efficiency with technological advancement. By addressing the gap in current literature regarding dynamic adaptation of map-matching algorithms and reducing dependency on high-cost hardware, this work propels the development of more adaptable, economically accessible, and energy-efficient mobile robotic systems that align well with the **I4.0**.

A SWOT analysis was conducted to elucidate this research's strategic implications further. This analysis helps evaluate the strengths, weaknesses, opportunities, and threats associated with the AI-enhanced indoor localization system, providing insights into the potential impact and challenges the PhD thesis faces. Table 2.5 presents this analysis in detail.

Table 2.5: SWOT analysis of the PhD research on AI-enhanced indoor localization

Aspect	Details
Strengths	- Advanced algorithmic approach using the RoPM method for superior map-matching. - Integration of AI-enhanced techniques to overcome static world assumptions. - Economic efficiency through cost-effective, low-resource solutions.
Weaknesses	- Dependency on technological advancements in AI and deep learning. - Complexity in implementing efficient deep-learning models for real-time dynamics. - Potential unforeseen technological limitations. - Technical complexity of implementing the AI system with real-time requirements and low-cost embedded systems.
Opportunities	- Increasing demand for dynamic localization solutions in logistics and autonomous navigation. - Alignment with I4.0 trends towards automation. - Potential for interdisciplinary applications in automated vehicles, drones, and other options. - The current dynamic localization solutions are high-cost, energy-consuming, and not scalable.
Threats	- Economic constraints impacting research scope or technology deployment. - Potential global pandemics hindering the progress and implementation of the research. - Emergence of new technological advancements that could supersede current methodologies.

Chapter 3

Design and Development of the Robotic Platform and Localization Systems

This chapter presents the development of three key components essential to this research: the autonomous mobile robot platform and two distinct localization systems. Beginning with an overview of the system architecture, the chapter then details the development of the [AMR](#) platform in Section [3.1](#), covering its mechanical design, kinematics, hardware and software architecture, localization system, and path and trajectory planning modules. Section [3.2](#) describes the development of a ground truth localization system, which serves as a reliable reference for validating the main contribution of this thesis. Finally, Section [3.3](#) introduces the primary contribution - an AI-enhanced indoor localization system that leverages object detection techniques. This novel system incorporates an outlier rejection strategy and builds upon existing methodologies to improve localization accuracy and robustness.

System Overview

Before delving into the specific components of this research, it is important to understand the overall system architecture and its operational modes. The system comprises several interconnected components that work together to achieve robust localization in dynamic environments.

Figure [3.1](#) illustrates the entire system's architecture, comprising the robotic platform components, the ground truth system, and the edge device. The dashed rectangle labeled SimTwo simulator indicates that all components within can be simulated. The system integrates various sensors (shown in green), processing devices (in blue), and actuators (in yellow). The ground truth system runs independently on a Raspberry Pi 5, providing reference pose data for system validation. The edge device handles AI-based image processing or posterior analysis of logged data via the developed [RoPM](#) application.

The developed [RoPM](#) application, which implements the AI-enhanced localization system, runs on the edge device and demonstrates significant versatility in its operational modes. As shown in Figure [3.2](#), the application can process data from four different sources: real-time data

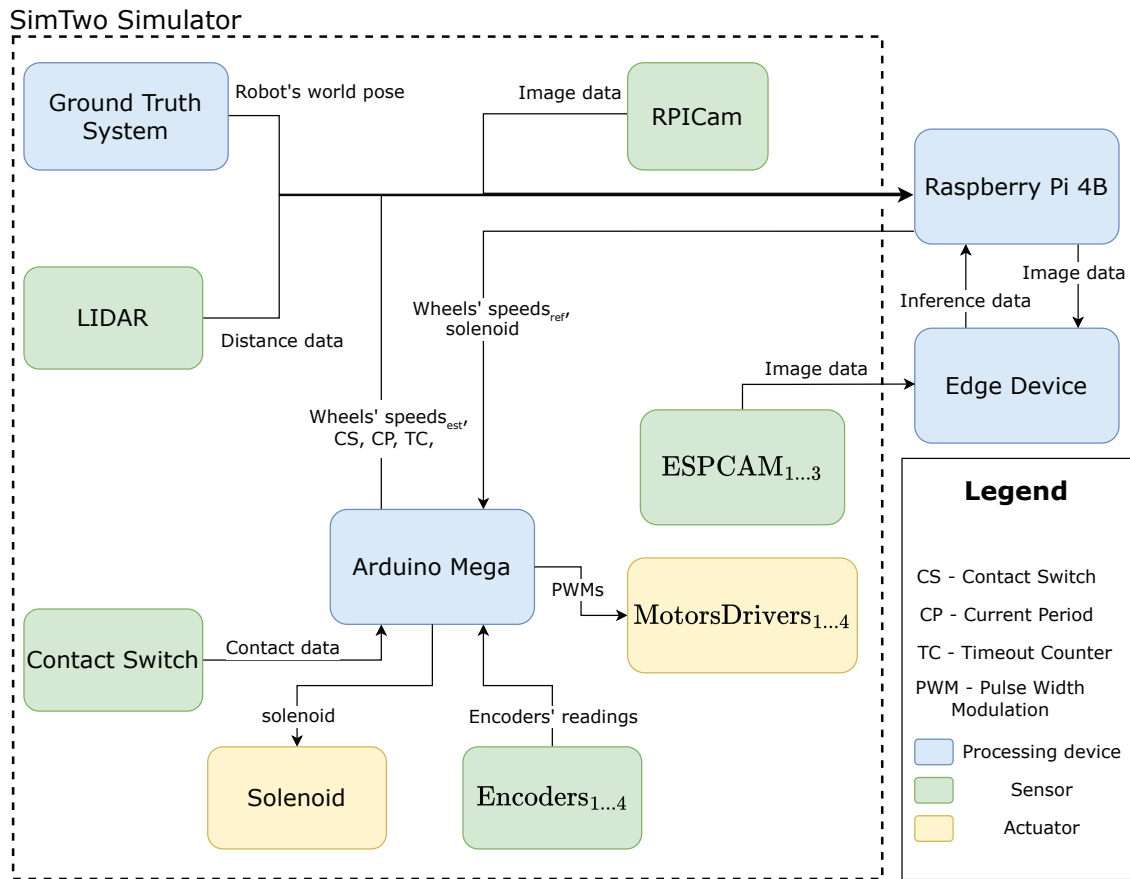
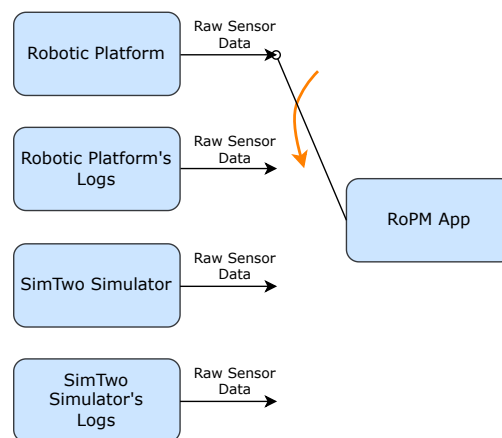


Figure 3.1: Overall system architecture

from either the physical robotic platform or the SimTwo simulator and logged data from either source. This flexibility enables comprehensive testing and validation across different scenarios while maintaining consistent performance regardless of the data source.

Figure 3.2: Operational modes of the **RoPM** application showing data flow from different sources

The subsequent sections provide detailed explanations of each system component, beginning with the robotic platform's design and implementation in Section 3.1, followed by the ground truth system development in Section 3.2, and concluding with the AI-enhanced indoor localization system, RoPM, in Section 3.3.

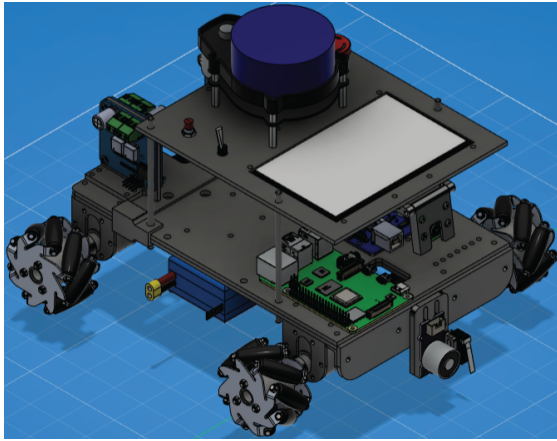
3.1 Robot's Architecture & Design

This section presents the design of the AMR, structured around its primary components and operational functionalities. The development of this robotic platform served multiple objectives: first, as a testbed for developing localization expertise that would later inform the main contribution of this thesis; second, as a competitive platform for participation in a national robotic competition; and third, as the foundation for components that would be adapted into the AI-enhanced localization system. The discussion begins with an in-depth look at the mechanical design, contrasting initial design drafts with the implemented version and highlighting key improvements made for system validation. Following this, the kinematics of the AMR are described, underscoring the capabilities of its 4-wheeled mecanum drive system. The hardware architecture subsection describes the hardware components used in the robot. The software architecture is then detailed, highlighting the software frameworks and structures that enable the robot's operation. Finally, the subsections on the localization system, path planning, and trajectory planning discuss the methods used to navigate and operate within the robot's workspace environment efficiently and autonomously.

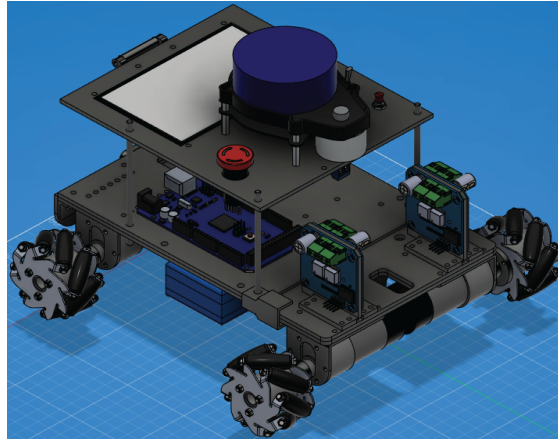
3.1.1 Mechanical Design

The mechanical design of the AMR was primarily developed as a master thesis project by Kais Baidi [200]. He contributed to the robotic platform's design, tailored for participation in the RAF competition—a national autonomous robotics contest emphasizing logistics tasks within an automated I4.0 factory warehouse setting. Figure 3.3 shows two perspectives of the AMR's design and assembled prototype in a top-bottom comparison. The upper row displays the modeled design in Figures 3.3a and 3.3b, while the bottom row shows the assembled prototype in Figures 3.3c and 3.3d.

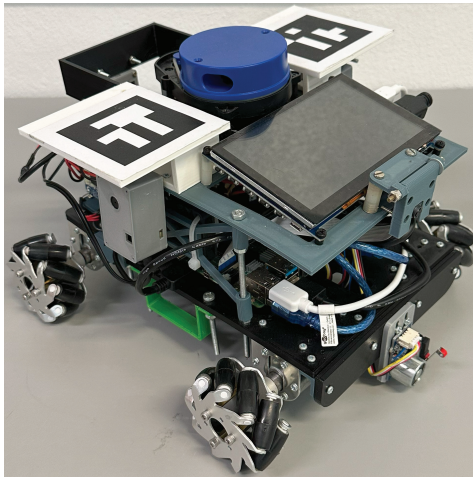
The architecture of the AMR was designed to support omnidirectional movement, facilitated by a 4-wheeled mecanum wheel system. The mechanical design strategically accommodated space for four motor drivers capable of handling the motor reversals, inertia, and power demands typical of omnidirectional movement. This design of four individual drivers considered cost and reliability, ensuring that the torque and current spikes do not overwhelm the drivers. Also, they include reverse polarity and other protections for added reliability. Additionally, the design incorporated a Raspberry Pi 4B and an Arduino Mega 2560 on the first level of the robot, alongside essential components such as a contact switch and an electromagnet. The upper level of the robot hosts a touchscreen display, a Raspberry Pi Camera Module v2 (RpiCam V2), and a YDLIDAR X4. It also includes two power switches—one for activating the entire system and another to cut off



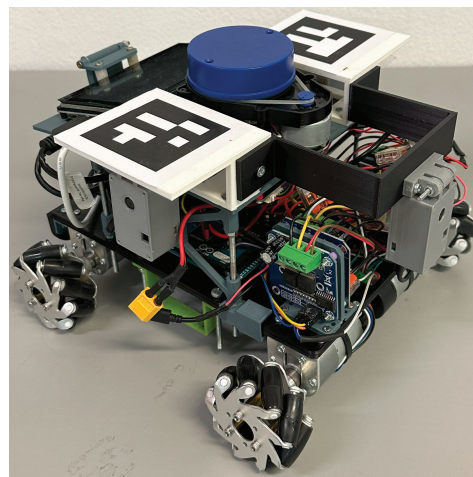
(a) Isometric-front-top-left view of the AMR's design



(b) Isometric-back-top-right view of the AMR's design



(c) Isometric-front-top-left view of the AMR prototype



(d) Isometric-back-top-right view of the AMR prototype

Figure 3.3: Isometric views of the AMR's design and its prototype

power to the motors, enhancing safety during operation. Later enhancements added three ESP32-CAMs mounted on the second level, though these modifications will be elaborated upon later in the chapter. A distinctive feature of the AMR's design is dividing the first level into front and back sections, connected only by a movable axis equipped with a bearing. This arrangement allows for 4-point contact with the ground at all times, ensuring stable movement across different directions, which is important during lateral or complex maneuvers.

3.1.1.1 Improvements to the Robotic Platform

Specific design improvements were implemented on the robotic platform to allow the validation of the localization systems. These improvements included the addition of two support structures attached to the second level of the robot, situated below the z-level of the LiDAR scanner and aligned along the x-axis of the robot frame. Each support is a square with sides measuring 8 cm,

designed to host ArUco markers that measure 6 cm on each side. This arrangement leaves a 1 cm border of white paper around each marker, enhancing the contrast and enabling their detection by the camera. These supports are essential for the ground truth localization system, described later in this chapter, which relies on these markers to accurately locate the robot and ensure the effectiveness of the localization systems' validation. Figure 3.4 shows the AMR with these improvements, illustrating the placement of the ArUco markers.

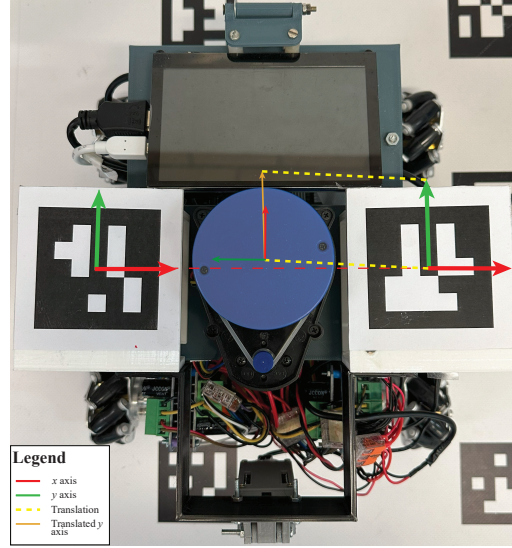


Figure 3.4: The AMR prototype with design improvements for localization validation, showing the ArUco markers and LIDAR reference frames, and their offset (orange arrow).

As illustrated in Figure 3.4, each ArUco marker has its coordinate frame. The ground truth system estimates the 2D pose at the midpoint between these markers (indicated by the dashed red line), with its reference frame centered on the LiDAR sensor. The orange axis represents the translated y-axis of one marker, highlighting the angular offset between the ground truth and LiDAR reference frames. These offsets, resulting from assembly non-idealities, can be addressed through extrinsic calibration, which will be detailed later in this thesis.

3.1.2 Kinematics

The robot's kinematics can be divided into forward and inverse kinematics models. The forward kinematics model allows the computation of the robot frame's speed states, $[v, v_n, \omega]$, based on the angular speed of each wheel $[\omega_{1..4}]$. The inverse kinematics model, on the other hand, computes the wheels' angular speeds from the respective robot's speed states. Figure 3.5 depicts the kinematic architecture of the robot, and Equations 3.1 and 3.2 represent the forward and inverse kinematics model of the 4-wheeled mecanum mobile robot respectively [201].

Where:

- $[v, v_n, \omega]$: Robot's speed states, linear, orthogonal, and angular speeds, respectively.
- $[\omega_i]$: Robot wheels' speeds.

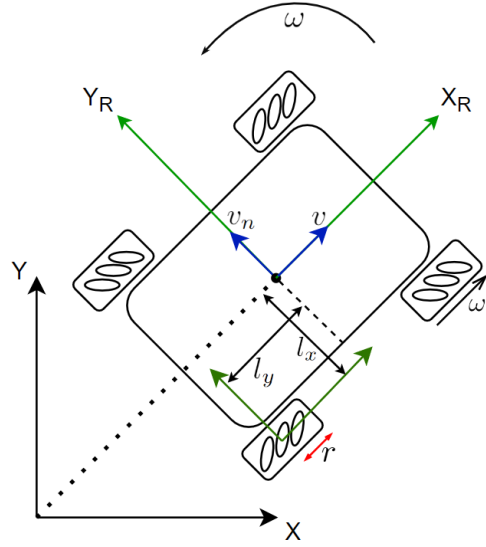


Figure 3.5: Robot's kinematic architecture

- $[l_x, l_y]$: Distance of the robot's frame to the midpoint of the wheels.
- $[X_R, Y_R]$: Robot's body frame.
- $[X, Y]$: Space frame.
- r : wheels' radius.

Thus, the following equations can express the forward kinematics model:

$$\begin{bmatrix} v \\ v_n \\ \omega \end{bmatrix} = \frac{r}{4} \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 1 & 1 & -1 \\ -\frac{1}{l_x+l_y} & \frac{1}{l_x+l_y} & -\frac{1}{l_x+l_y} & \frac{1}{l_x+l_y} \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix} \quad (3.1)$$

In contrast, the inverse kinematics model can be expressed by the following equation:

$$\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix} = \frac{1}{r} \begin{bmatrix} 1 & -1 & -(l_x+l_y) \\ 1 & 1 & (l_x+l_y) \\ 1 & 1 & -(l_x+l_y) \\ 1 & -1 & (l_x+l_y) \end{bmatrix} \begin{bmatrix} v \\ v_n \\ \omega \end{bmatrix} \quad (3.2)$$

3.1.3 Hardware Architecture

The hardware architecture of the robot, as illustrated in Figure 3.6, incorporates a range of input devices, control devices, power management components, and output devices, enabling sophisticated functionalities.

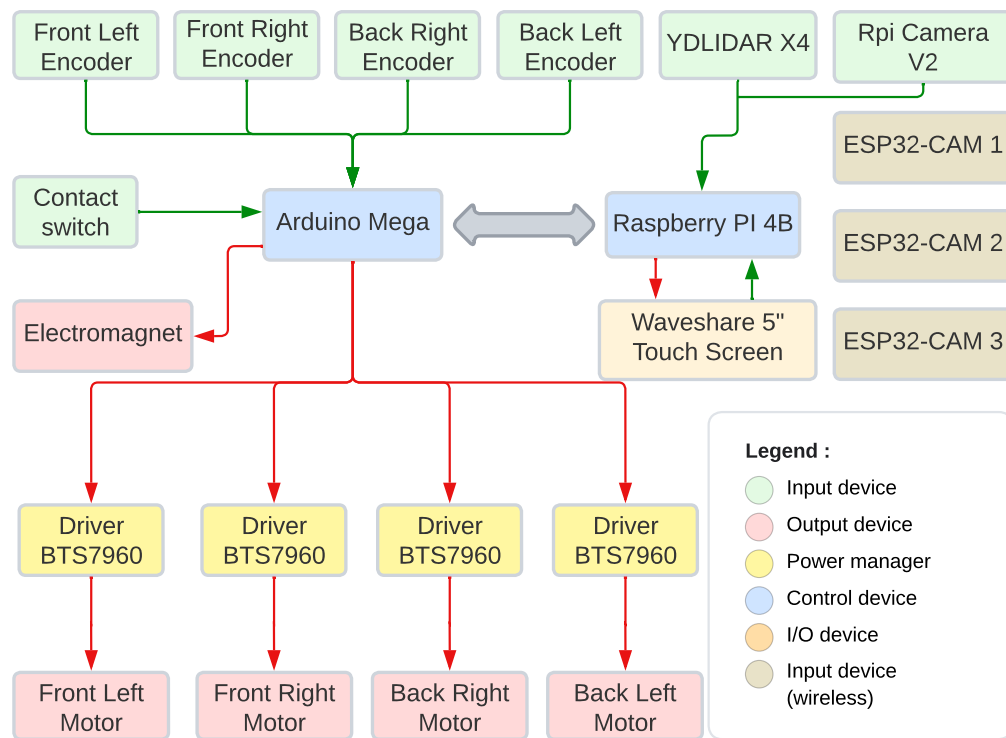


Figure 3.6: Robot's hardware architecture diagram

Controllers

1. **Raspberry Pi 4B:** Serves as the central processing unit of the robotic system, handling high-level decision-making and data processing. It is connected to several peripherals, enabling user interaction and data acquisition.
2. **Arduino Mega:** Functions as a subordinate microcontroller that handles the low-level firmware of the platform. Its task is managing the control cycle of the robotic platform, interfacing with multiple encoders and other sensors and actuators, and managing real-time control for the motors.

Sensory Inputs

1. **Quadrature encoders (Front Left, Front Right, Back Right, Back Left):** Each motor is equipped with a 64 **Counts per revolution (CPR)** encoder that delivers feedback on rotational speed, position, and direction. This data is needed for precise navigation and motion control. These encoders are paired with the motors' 30:1 gearbox, enhancing their performance and achieving an effective resolution of 3840 **CPR** on the motor's output shaft.

2. **YDLIDAR X4:** A 360-degree [LiDAR](#) laser scanner connected to the Raspberry Pi, used for localization purposes.
3. **Rpi Camera V2:** A vision sensor providing visual data to the Raspberry Pi 4B for localization purposes. It was calibrated using OpenCV library [202].
4. **ESP32-CAM Modules (CAM_{1...3}):** Equipped with Wi-Fi capabilities, these modules send imagery data via [User Datagram Protocol \(UDP\)](#) protocol to an edge device for further semantic localization computation. The selection of ESP32-CAMs over alternatives such as omnidirectional cameras, fish eye lenses, or concave mirror setups was primarily motivated by simplicity, cost, and ease of calibration. Using ESP32-CAMs simplifies the integration and image multiplexing processes due to their straightforward handling of distortions, as opposed to the complex distortion correction required by other camera types. Each ESP32-CAM module costs approximately €2, making it a cost-effective choice, especially when including the inexpensive designed and printed supports. Furthermore, calibrating these cameras using the OpenCV library to obtain intrinsic parameters and distortion coefficients is more direct than the intricate calibration required for other cameras involving detailed adjustments to projection matrices and panoramic distortions.
5. **Contact switch:** Industrial-grade contact switch to detect collision/contact with external entities.

User Interface

1. **Waveshare 5" Touch Screen:** Connected to the Raspberry Pi, it is a direct human-machine interface, allowing on-site adjustments, status displays, and manual control or overrides.

Actuators

1. **BTS7960 H-Bridge Drivers (Front Left, Front Right, Back Right, Back Left):** These motor drivers control the respective motors based on commands issued by the Arduino Mega 2560, facilitated by the feedback from the motor encoders for precise speed and directional control.
2. **D37 12VDC Pololu Motors (Front Left, Front Right, Back Right, Back Left):** The selected motors for this project are the 12VDC D37 30:1 Pololu motors, which are outfitted with integrated quadrature encoders.
3. **Electromagnet:** Operated via the Arduino Mega for box handling tasks on the factory floor.
4. **Mecanum wheels 65 mm:** Allows omnidirectional movement.

Power Management

1. **Battery (14.8V, 5200mAh):** Supplies power to the entire system, ensuring operation without external power sources.
2. **5V Voltage Regulator:** A buck converter that steps down and regulates the battery voltage to the 5V bus, necessary for powering the Raspberry Pi, the touch screen, the microcontroller, and the peripherals as shown in Figure 3.7.

Table 3.1 summarizes the main hardware components of the robot and their functions. In addition, to source all these components, the power architecture diagram in Figure 3.7 illustrates the electronic system design for the robotic platform.

Table 3.1: Robot's hardware components

Component Type	Component	Description
<i>Input Devices</i>		
	Motors' Integrated Quadrature En-	Measure wheel rotation speeds and direction
	coders (4)	
	Contact Switch	Detects collisions
	YDLIDAR X4	360-degree LiDAR sensor
	Raspicam Module V2	Captures images
	ESP32 Cams (3)	Provides additional visual data
<i>Control Devices</i>		
	Arduino Mega 2560	Manages low-level firmware: control cycle, motors, and sensors
	Raspberry Pi 4B 4 GB RAM	Processes sensory data
<i>Output Devices</i>		
	Electromagnet	Allows object manipulation
	12 VDC Pololu D37 motor (4)	Drives each wheel
<i>Power Management</i>		
	BTS7960 H-bridges (4)	Controls motor speed and direction
	Buck Converter	Regulates voltage to 5 VDC
	LiPo Battery (4-cell, 16.8 VDC, 5200 mAh)	Powers the system
<i>I/O Device</i>		
	WaveShare 5" Touch Screen	Displays system statuses and controls

Wiring and Connectivity

The diagram uses a color-coded wiring scheme to denote several connections:

- **Red wires:** +5V power supply.
- **Black wires:** Ground connections.
- **Blue wires:** +16.8V power from the battery.

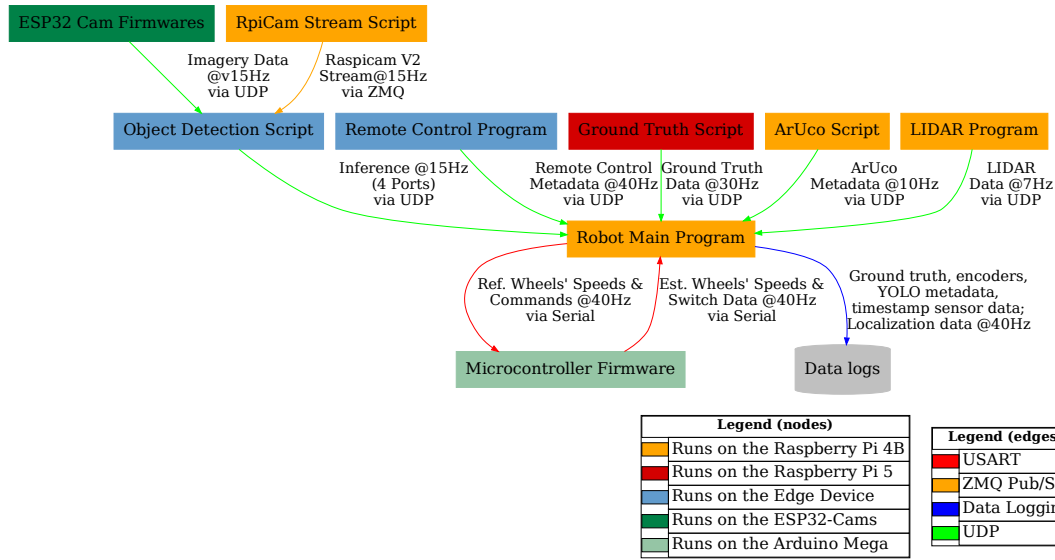


Figure 3.8: Software architecture data flow diagram

- **Data logs:** The C++ program also logs the raw sensory and localization data for posterior analysis.

3.1.4.1 High-Level Control

The high-level control uses an event-driven programming model that efficiently manages system resources. The main loop monitors file descriptors for available data and triggers appropriate handlers, preventing CPU throttling. This design ensures responsive and efficient operation by minimizing idle time and quickly processing new information. This paradigm was chosen since the complexity and scalability of the project did not require using threads.

Algorithm 1 represents the main control loop of the platform, and it reflects the robot's ability to handle multiple data streams simultaneously and decide which tasks to prioritize based on the data's readiness. This setup maximizes resource usage efficiency across the robot's hardware components and ensures timely responses to robot environment changes. Following the order of the file descriptors and their handlers' calls, below is described their functionality:

- **Serial descriptor:** Serial communication between the Raspberry Pi 4B and the Arduino Mega. As shown in Figure 3.8 and described in this subsection, both exchange sensory and command information regarding motors, encoders, the contact switch, control cycle metadata, and the electromagnet.
- **Ground truth descriptor:** UDP communication from the Raspberry Pi 5, which is installed on the test-bed environment, better described later in the thesis, to the Raspberry Pi 4B for validation of the localization systems.

Algorithm 1 Robot’s control loop

```

1: procedure MECANUMROBOTLOOP
2:   while robot is operational do
3:     Check for available data from all inputs           ▷ Serial and UDPs comms. There is timeout handling
4:     if new serial data or remote control input then
5:       if in remote control mode then
6:         Process remote control input
7:       else
8:         Process serial data                           ▷ Autonomous mode
9:       end if
10:    Perform EKF prediction stage
11:    Execute other subsystems                           ▷ Logging, path and trajectory planning, low-level commands
12:  end if
13:  if new ground truth data then
14:    Update ground truth information
15:    Check if sufficient observations for operation
16:  end if
17:  if new LIDAR data then
18:    Process LIDAR information
19:  end if
20:  for all YOLO detectors do
21:    if new YOLO data then
22:      Process YOLO detection information
23:    end if
24:  end for
25:  if new ArUco data then
26:    if ArUco data is within valid bounds then
27:      Process ArUco data
28:    end if
29:  end if
30:  if new box-related data then
31:    Handle box treatment operations
32:  end if
33: end while
34: end procedure

```

- **Remote control descriptor:** [UDP](#) communication from the Edge Device to the Raspberry Pi 4B for remote robot control. This C++ program encodes a game pad’s joystick data to the robot’s speed states $[v, v_n, \omega]$.
- **LIDAR descriptor:** It is a standalone C++ program that decodes the [LiDAR](#) scans via serial communication, sent via [UDP](#) communication to the robot’s main C++ program.
- **YOLO Metadata₁₋₄:** Object detection inference data sent via [UDP](#) protocol from the Edge Device back to the robot’s main C++ program for decision-making in the localization system.
- **ArUco descriptor:** The primary localization system made for the robot for the national robotic competition [RAF](#). When the ArUco script is run, the “CamStream” script cannot run since they both use the “Rpi Cam v2”.
- **Box descriptor:** This descriptor was created because this robot was also tailored to compete in the [RAF](#) about [I4.0](#) automated warehouses. Therefore, the robot needed to organize boxes on a factory floor autonomously.

Adjustments to LIDAR's payload UDP communication

The largest sensor payload comes from the **LiDAR** scans, and since both C++ programs run on the same machine, the Raspberry Pi 4B, there is no concern regarding **Maximum Transmission Unit (MTU)** size and data fragmentation. Thus, it was decided to send **LiDAR** data in its entirety; however, computing **Cyclic Redundancy Check (CRC)** checksum to ensure data integrity. Using the Boost library, Algorithms 2 and 3 display the serialization and deserialization process.

Algorithm 2 Send serialized LIDAR data

```

1: procedure SENDSERIALIZEDLIDARDATA
2:   Prepare LIDAR distance data                                ▷ Collect and process raw LIDAR readings
3:   Create payload with processed distance data
4:   Serialize payload                                          ▷ Convert payload to binary format
5:   Prepare header                                            ▷ Include data type and payload size information
6:   Serialize header
7:   Combine serialized header and payload                    ▷ Construct complete message
8:   Calculate and append error checking data                  ▷ e.g., CRC checksum
9:   Transmit data over network                                ▷ Send via UDP protocol
10: end procedure

```

The encoding process in the robot regarding the **LiDAR** scan payload begins by preparing it, which is stored directly in a buffer. This payload is serialized with a message header that indicates the type *LASER_DATA* and the size of the message. Both elements are serialized using the Boost library in binary format. The **CRC** checksum is calculated and appended to the serialized data. This complete package, comprising the header, payload, and checksum, is transmitted over **UDP**.

Algorithm 3 Handle and deserialize received LIDAR data

```

1: procedure HANDLELIDARDATA
2:   Receive serialized message                                ▷ Incoming data from network
3:   Verify data integrity                                    ▷ Extract and verify CRC Checksum
4:   if data integrity verified then
5:     Extract and deserialize header                          ▷ Retrieve message metadata
6:     if message type is LIDAR data then
7:       Extract and deserialize payload                        ▷ Retrieve actual LIDAR readings
8:       Process LIDAR data                                    ▷ Store data
9:     else
10:      Handle incorrect message type                          ▷ Log error
11:    end if
12:  else
13:    Handle data corruption                                    ▷ Log error
14:  end if
15: end procedure

```

At the destination, the integrity of the received data is verified using the extracted **CRC** checksum. The message is deemed intact if the checksum matches and the header and payload are deserialized. The message type is checked to confirm it pertains to *LASER_DATA*. Upon successful validation, the payload is either processed further or stored.

Adjustments to LIDAR's real-world characteristics

Discrepancies in the extrinsic pose of the **LiDAR** sensor between the real and simulated robots necessitated specific adjustments. The sensor was installed inversely (backward), generating a

180-degree offset, and its firmware output measurements were in a clockwise direction, contrary to the simulated model's counterclockwise expectations. These discrepancies could not be rectified through firmware configuration changes, including an unexpected 11-degree offset of the first **LiDAR** measurement from the datasheet's origin.

To address these issues and align the real sensor data with the simulated model, a method was implemented within the robot's application software to reorganize the **LiDAR**'s scan data before sending it to the robot's main application, where the localization system runs. Algorithm 4 was employed to adjust the data container during the **LiDAR** serialization process, ensuring the physical **LiDAR**'s readings accurately reflected the simulated model.

Algorithm 4 Reorganize LIDAR Scan Data Container

```

1: Input: lidar_scan
2: Output: lidar_scan
3: procedure ORGANIZELIDARCONTAINER
4:   if lidar_scan is empty then
5:     Handle empty scan case                                     ▷ Log error
6:   return
7:   end if
8:   Initialize temporary storage for reorganized data
9:   Calculate new reference angle                                   ▷ Adjust for LIDAR offset
10:  Determine the angular resolution of the LIDAR scan
11:  for each data point in lidar_scan do
12:    Calculate adjusted angle                                     ▷ Apply offset and normalize to [0, 360)
13:    Determine new index based on the adjusted angle
14:    Store data point at new index in temporary storage
15:  end for
16:  Update LIDAR scan with reorganized data
17: end procedure
  
```

ESP32-CAM image stream transmission

The approach to communicating the image stream via the network was also more complex than the traditional **UDP** datagram, and this is because the packets can arrive out of order, become lost, and cause other problems that corrupt the reconstructed image on the receiver end. Even with all these problems, the **UDP** protocol is still much more reliable than the **Transmission Control Protocol (TCP)** protocol because if one of these problems occurs, both systems will try to correct these lost packets, and this would overload the socket. Therefore, **UDP** was chosen, and if a particular packet gets corrupted, it is simply discarded on the receiver end, and then the next is read. However, since imagery data are extensive, the probability of these problems occurring is higher. Therefore, to avoid image misassembly or reconstructing incomplete data, a **XOR** checksum approach was chosen for the ESP32-CAMs for being more computationally efficient than the de facto **CRC**. Algorithm 5 displays the encoding script for the ESP32-Cameras, whereas Algorithm 6 shows the Python script that handles video transmissions from all four cameras.

As can be seen in Algorithm 5, it describes how the ESP32-CAM firmware captures an image, computes its checksum for error detection, and sends both the image and its checksum to a receiver over **UDP**.

Algorithm 5 ESP32 camera system - sending images

```

1: procedure ESP32CAMERAOPERATION
2:   Initialize camera system                                ▷ Set up camera with required configuration
3:   Establish network connection                            ▷ Connect to WiFi network
4:   Set up communication protocol                          ▷ Initialize UDP on specified port
5:   while camera system is active do
6:     Attempt to capture an image frame
7:     if frame captured successfully then
8:       Prepare image for transmission                    ▷ Compute XOR checksum for error detection
9:       Transmit image data                               ▷ Send image and checksum via UDP
10:    else
11:      Handle capture failure                                ▷ Log error
12:    end if
13:    Manage frame rate                                      ▷ Wait without blocking execution
14:  end while
15: end procedure

```

Algorithm 6 Python receiver system - processing images

```

1: procedure IMAGERECEIVERSYSTEM
2:   Initialize system components                            ▷ Set up CLI arguments, signal handlers, UDP socket
3:   Load object detection model                          ▷ Prepare YOLO model for image processing
4:   Configure output settings                             ▷ Set up display and/or saving options
5:   Prepare multi-camera reception                       ▷ Create a thread pool
6:   Initialize a thread queue                             ▷ To communicate to the main thread
7:   for each configured camera do
8:     Start image reception thread                        ▷ ImageReceptionThread(port, camera)
9:   end for
10:  while system is active do
11:    if frame available in queue then
12:      Retrieve processed frame from the queue
13:      Display frame                                       ▷ If display option is enabled
14:      Save frame                                         ▷ If save option is enabled
15:    end if
16:    Check for system exit conditions
17:  end while
18:  Signal all threads to terminate
19:  Perform system cleanup                                ▷ Close threads, release resources
20: end procedure

21: Input: port, camera_label
22: procedure IMAGERECEPTIONTHREAD
23:  while thread is active do
24:    Receive image data on assigned port
25:    if complete frame received then
26:      Verify frame integrity                                ▷ Check XOR checksum
27:      if frame is valid then
28:        Process image                                    ▷ Perform object detection with YOLO
29:        Add processed frame to the thread's specific queue
30:        Transmit detection results                        ▷ If required by configuration
31:      else
32:        Handle corrupted frame                            ▷ Log error and discard
33:      end if
34:      Reset frame buffer                                ▷ For next image
35:    end if
36:  end while
37: end procedure

```

As shown on Algorithm 6, the receiver system is designed to process and multiplex image data transmitted from ESP32-based cameras. It initiates by parsing command-line arguments and setting up a signal handler for a clean shutdown. Upon configuration, a **UDP** socket is prepared to receive incoming images, and the **YOLO** model is loaded with pre-trained weights to enable real-time object detection. A thread pool is created to manage multiple image sources, with each thread

dedicated to handling data from a specific camera port. These threads continuously listen for incoming UDP packets, identifying the start and end of JPEG frames to assemble complete images. Once an image is fully received, its integrity is verified through a checksum. Valid images are then decoded and processed through the YOLO model, which detects and classifies objects within the frame. Based on the user's preferences specified through command-line options, the system may display the processed images in real-time or save them for further analysis. Additionally, detection results can be transmitted to the robot for decision-making. The system remains in operation until a shutdown signal is received, ensuring all resources are properly released and all active threads are closed, thereby maintaining system stability and data integrity.

Remote control

For testing purposes, it is essential to have the ability to control the robot remotely. For this reason, the approach was to implement a C++ application that maps a gamepad's joystick movements to the robot's speed states $[v, v_n, \omega]$ and buttons to the robot's commands. Therefore, the Simple DirectMedia (SDL)¹ library was chosen to perform this for its reliability and minimalist approach to low-level access to hardware. Algorithm 7 shows the high-level algorithm to perform this.

Algorithm 7 Gamepad to robot communication

```

1: procedure GAMEPADTOROBOTCOMMUNICATION
2:   Initialize gamepad input system
3:   Open gamepad
4:   if gamepad initialization fails then
5:     Log error and exit
6:   end if
7:   Initialize UDP connection to the robot
8:   while system is active do
9:     Read gamepad input
10:    Calculate robot speeds                                ▷ Encode
11:    Serialize speed data
12:    Send data to the robot via UDP
13:    Regulate communication rate                          ▷ Sleep to maintain 40 Hz of operation frequency
14:  end while
15:  Cleanup and release resources
16: end procedure

```

3.1.4.2 Low-Level Firmware

The low-level firmware of the microcontroller was designed to configure and manage the robot's operational hardware components. This includes initializing motor controllers, the control cycle, the electromagnet, the contact switch, the Interrupt Service Routine (ISR), and the serial communication.

Setup

¹<https://www.libsdl.org/>

Algorithm 8 details the setup procedure, emphasizing each element's configuration to ensure optimal performance and responsiveness in real-time operations. Each motor (front left, front right, back left, and back right) is configured with two pins for quadrature encoder inputs and two for **Pulse-width Modulation (PWM)** outputs, enabling control over motor speed and direction. The encoder inputs are set with internal pull-up resistors to read changes in the encoder states, and they are used for distance measurement and motor speed regulation. The setup includes configuration for an electromagnetic pin and a contact switch. The electromagnetic pin is configured as an output to control an electromagnet, which is used for tasks requiring object manipulation. The contact switch is set with an internal pull-up resistor to detect physical contact reliably, serving as an input for interaction handling. Also, to enable the robot to react to time-sensitive events, the firmware configures timer interrupts for precise timing operations. The interrupt setup involves disabling global interrupts during configuration (with “cli()”), setting the “Timer One” control register to **Clear Timer on Compare Match (CTC)** mode, and defining the timer compare match value to establish a consistent interrupt interval of 80 μ s. This configuration is vital to read reliably the four motor encoders. Serial communication is initialized with a Baud rate of 115200 bps, providing a fast channel for data exchange between the microcontroller and the Raspberry Pi 4B. The firmware initializes serial communication routines, setting up handlers for incoming packets and defining routines for outbound communication. This allows the microcontroller to send and receive commands and data efficiently, facilitating robust and responsive control.

Algorithm 8 Low-level firmware - setup

1: procedure LOWLEVELFIRMWARESETUP	
2: Configure motor pins for all motors	▷ Set encoder and PWM pins for front/back and left/right motors
3: Set up electromagnet and contact switch pins	▷ Configure electromagnet as output and contact switch as input pull-up
4: Configure timer interrupt	▷ Set up 80 μ s interrupt using Timer1 in CTC mode
5: Initialize serial communication	▷ Set the baud rate to 115200 and configure handler method
6: end procedure	

This detailed setup ensures that each hardware component is configured for its role in the robotic system.

Main loop

The core functionality of the robot is driven by the main loop, represented in Algorithm 9 in the low-level firmware, which handles tasks at defined intervals, ensuring timely responses and updates to control commands. The loop continuously checks for serial communication data, processes encoder readings, manages motor control, and handles safety mechanisms.

The loop begins by checking for available serial data. When available, it is read and processed through a state machine, ensuring robust data integrity and operational synchronization. The loop runs a control interval every 25 milliseconds, within which all operational checks and updates are performed. This includes updating the robot's timeout counter to monitor responsiveness, reading encoder values for movement control, adjusting motor speeds based on the controllers' demands, and verifying the status of the touch switch. If the robot becomes unresponsive (detected by the timeout counter reaching a threshold), a cutoff routine is initiated to halt all motors, ensuring

Algorithm 9 Low-level firmware - main loop

```

1: procedure LOWLEVELFIRMWAREMAINLOOP
2:   if serial data available then
3:     Process incoming serial data
4:   end if
5:   if control interval elapsed then ▷ 25 ms control loop
6:     Update robot timeout counter
7:     if robot unresponsive then
8:       Disable robot
9:     end if
10:    Update last execution timestamp
11:    Read and process sensor data ▷ Includes encoders, touch switch, and motor speed estimation
12:    Perform motor control
13:    Update electromagnet state
14:    Send telemetry data ▷ Includes encoder values, estimated speeds, switch state, timing info
15:    Send acknowledgment
16:  end if
17: end procedure

```

safety. Additionally, the loop handles actuator control for tasks requiring physical interactions, like using an electromagnet for object manipulation. Extensive data regarding the robot's state, such as encoder readings, estimated wheel speeds, and other critical telemetry, are regularly sent back to the Raspberry Pi 4B, which also sets the control cycle of the robot to 40 Hz.

Interrupt service routine

The firmware incorporates a specialized [ISR](#) to handle encoder updates for all four motors, essential for maintaining precise control over motor position and speed. The [ISR](#) is displayed in Algorithm 10 and is triggered by Timer 1's compare match A interrupt, which is set to fire at regular intervals, ensuring that encoder readings are updated in real-time to reflect motor movements accurately.

Algorithm 10 ISR for Timer 1 compare match A interrupt

```

1: Initialize encoder state transition table ▷ Quadrature encoders state transition table
2: procedure TIMERINTERRUPTHANDLER
3:   for each encoder do
4:     Read new encoder state (A and B pins) ▷ Bitwise operation for fast execution
5:     Compute state transition (previous state + new state) ▷ Bitwise operation
6:     if encoder is odd-numbered then
7:       Increment position based on the transition table
8:     else
9:       Decrement position based on the transition table
10:    end if
11:    Update encoder total count
12:    Store new state for next iteration
13:  end for
14: end procedure

```

Each motor encoder's output is read and interpreted through a state machine approach, using a predefined table (*encoder_table*) that maps encoder state transitions to positional increments or decrements. This table-driven method allows for fast and efficient processing of encoder signals, translating them into positional adjustments. For each motor, the [ISR](#) reads the current state of the encoder's two output pins, combines them to form a current state, and computes the change

from the previous state. Depending on the motor (odd or even), the change is either added to or subtracted from the current position count. This is important for determining the direction and extent of motor rotation. It is also important to note that although the encoders are read in intervals of $80\ \mu\text{s}$, they are sampled by the main loop at a 40 Hz rate. These updates to encoder positions are critical for the motor controllers to adjust speeds and positions accurately based on the robot's movement commands and navigational needs. This real-time tracking ensures the robot can adjust its speed dynamically, responding adeptly to control inputs.

Motors speeds' controllers

The Arduino Mega controls the motor speed control system within the robot to manage four independent motors, each equipped with its own control and feedback mechanisms. The speed control is implemented using [Proportional-Integral Controller \(PI\)](#) controllers, which adjust the [PWM](#) signals based on encoder feedback to control motor speed and direction accurately.

The [PI](#) controllers were tuned using the [Internal Model Control \(IMC\)](#) methodology, which provides a structured approach to determine the controller gains based on a predictive model of the motor dynamics. The tuning sequence involved first setting the feedforward gain (K_f), which was derived based on the inverse dynamics of the plant. Subsequently, the proportional (K_p) and integral (K_i) gains were adjusted to enhance the stability and responsiveness of the system. The [PI](#) controller gains incorporate the inverse transformation from the controller output range (0-16.4V) to the PWM range (0-255), ensuring proper scaling between the designed voltage-based control signal and the microcontroller's PWM output capabilities.

In the digital implementation of the [PI](#) controllers, quantization and discretization can pose challenges. Quantization error arises from the finite resolution of the digital representation of the [PWM](#) signals, potentially leading to minor steady-state errors or oscillations around the set point. Due to the fixed sampling frequency of 40 Hz, discretization introduces a delay between the measurement of the speed and the adjustment of the [PWM](#) signals, which could affect the controller's ability to react instantaneously to changes in motor speed. Despite these potential problems, the controllers were designed carefully, and the sampling and control frequency were chosen to meet the requirements of the robotic platform and the required tasks. A closed-loop step response was generated to show the behavior of the controlled system. However, a methodology was followed to avoid the potential dead zone of the motor affecting the controller's ability. First, the tested motor was commanded to follow a reference signal and only then was another step response generated. Figure 3.9 illustrates this methodology and the system's responsiveness. It is possible to see that the closed-loop step responses do not overshoot and achieve steady-state quickly (few samples).

To show more technical details of the controlled system, Figure 3.10 illustrates only the second closed-loop step response of the motor, considering the previous reference signal as the new baseline. The response characteristics, with a rise time of 0.05 seconds and a settling time of 0.1 seconds, display the responsiveness and stability of the control system for the required reference

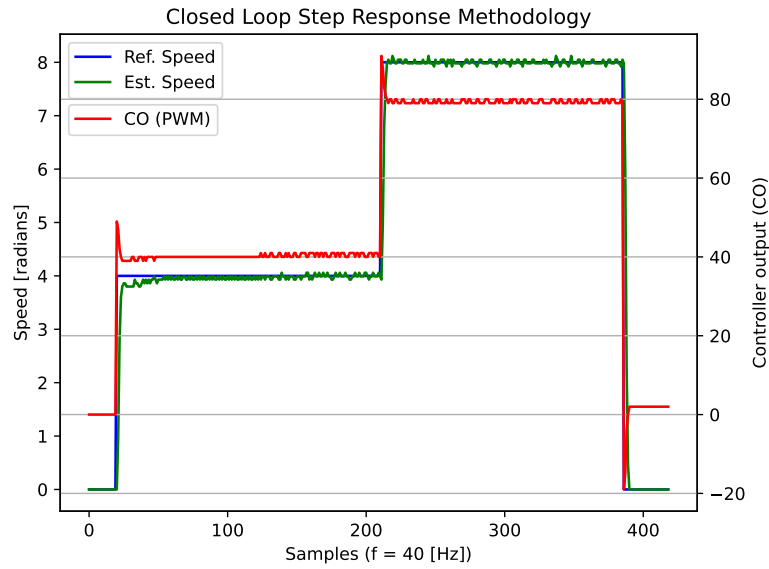


Figure 3.9: Speed closed-loop step response measurement, following a method for bypassing motor dead zone

signals (from 0 to 8 [rad/sec]). There was no need to design a controller to operate stably beyond the robot's required speed range. Therefore, extensive tests ensured stability until 12 [rad/s] in the motors.

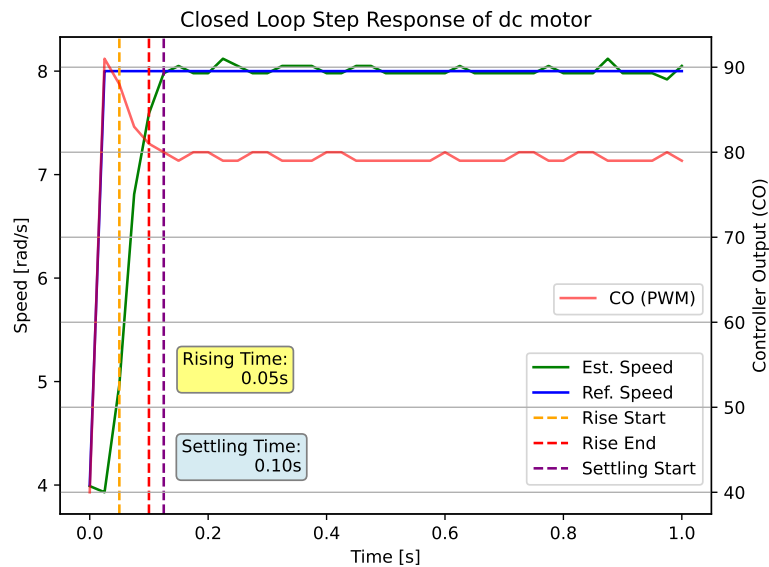


Figure 3.10: Speed closed-loop step response of one of the motors with the PI-Feedforward controller

This analysis presents one motor's behavior since all motors exhibit insignificant differences with identical controller characteristics. The **PI** controllers incorporate an anti-windup mechanism to prevent the integral term from accumulating excessive error when the actuator outputs are

saturated. Additionally, the outputs are constrained within the operational limits of the microcontroller's PWM module, ranging from -255 to 255. These two strategies effectively address the integral windup. Uniform controller parameters ($K_p = 3$, $K_i = 9$, $K_f = 9.22$) are used across all motors, reflecting the robustness of the controller for minor mechanical variations.

3.1.5 Localization System

The localization system uses an EKF, which functions in two main phases: prediction and update. The prediction phase estimates the state and covariance of the filter based on a state transition model. In contrast, the update phase involves assimilating new camera sensor data to refine these estimates.

- **Prediction Stage:**

1. State estimation via the state transition model using previous states and control inputs:

$$\hat{\mathbf{x}}_{k|k-1} = f(\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}_{k-1})$$

where $\hat{\mathbf{x}}_{k|k-1}$ is the filter's state variables, $[\hat{x}_k, \hat{y}_k, \hat{\theta}_k]$, $\mathbf{u}_{k-1}$ the control vector, and the state transition model is:

$$\begin{cases} \hat{x}_k = x_{k-1} + v_{k-1} \cos(\theta_{k-1}) \Delta t \\ \hat{y}_k = y_{k-1} + v_{k-1} \sin(\theta_{k-1}) \Delta t \\ \hat{\theta}_k = \theta_{k-1} + \omega_{k-1} \Delta t \end{cases}$$

which is the odometry model of the AMR. The estimated robot's speed state, $[v_{k-1}, v_{n_{k-1}}, \omega_{k-1}]$, are the control inputs, and Δt is the control period.

2. Estimation of predicted filter's covariance, $\mathbf{P}_{k|k-1}$, projecting the uncertainty of the state estimation forward in time:

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

where $\mathbf{F}_k$ is the Jacobian of the state transition model,

$$\mathbf{F}_k = \left. \frac{\partial f}{\partial \mathbf{x}} \right|_{\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}_k},$$

$\mathbf{P}_{k-1|k-1}$ the filter's covariance matrix from the previous state, and $\mathbf{Q}_k$ the covariance of the state transition model's noise.

- **Update Stage:**

1. Computation of innovation, $\tilde{\mathbf{y}}_k$, by comparing actual sensor measurements against predicted states:

$$\tilde{\mathbf{y}}_k = \mathbf{z}_k - h(\hat{\mathbf{x}}_{k|k-1})$$

where $\mathbf{z}_k$ is the observation and $h(\hat{\mathbf{x}}_{k|k-1})$ the observation model. The latter maps the measurements from the state space to the observation space.

2. Update of innovation covariance, $\mathbf{S}_k$, which represents the combined uncertainty of the predictions and measurements:

$$\mathbf{S}_k = \mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^\top + \mathbf{R}_k$$

where $\mathbf{H}_k$ is the observation matrix, the Jacobian of the observation model, and the covariance of the observation model's noise is $\mathbf{R}_k$.

3. Calculation of Kalman Gain, $\mathbf{K}_k$, to optimize the weight given to the innovation during state correction:

$$\mathbf{K}_k = \mathbf{P}_{k|k-1} \mathbf{H}_k^\top \mathbf{S}_k^{-1}$$

4. Update of state estimates, $\hat{\mathbf{x}}_{k|k}$, (called correction as well):

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \tilde{\mathbf{y}}_k$$

5. Update of filter's covariance, $\mathbf{P}_{k|k}$, using the innovation and Kalman Gain:

$$\mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_{k|k-1}$$

The prediction and update cycles are designed to function asynchronously, ensuring that the localization system can adapt to the dynamics of the robot's sensors effectively. The system employs a non-linear model due to the robot's kinematic constraints, necessitating an [EKF](#) over a standard Kalman Filter. The frequency of the prediction stage is set at 40 Hz, while the update stage operates at 10 Hz, allowing for real-time corrections based on external sensor data.

A [DFD](#) outlined in Figure 3.11 visually demonstrates these operations where $\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_e$ which is the robot's 2D pose.

Finally, given the circumstances when the robot and the localization system were developed, the covariance of the process model and the sensor measurements (camera with ArUco markers) were subjected to a sensitivity analysis checking the convergence and stability of the filter. This adjustment aimed to achieve a balance where the [EKF](#) was not overly sensitive to measurement noise, thereby avoiding erratic behavior while also ensuring the state estimates did not lag significantly behind the true states of the robot considering a fixed nominal speed of the robot to perform in the competition, [RAF](#). The caveat of this method is that the covariance of the [EKF](#), $\mathbf{P}$ does not represent the uncertainty of the states estimates as a physical characteristic of the system, and instead a threshold that will define how the filter will react to new state transitions and measurements. If both covariances noise matrices are correctly estimated, [EKF](#), $\mathbf{P}_{k|k-1}$ would represent the squared uncertainty of the filter's states estimates $\hat{\mathbf{x}}_k$.

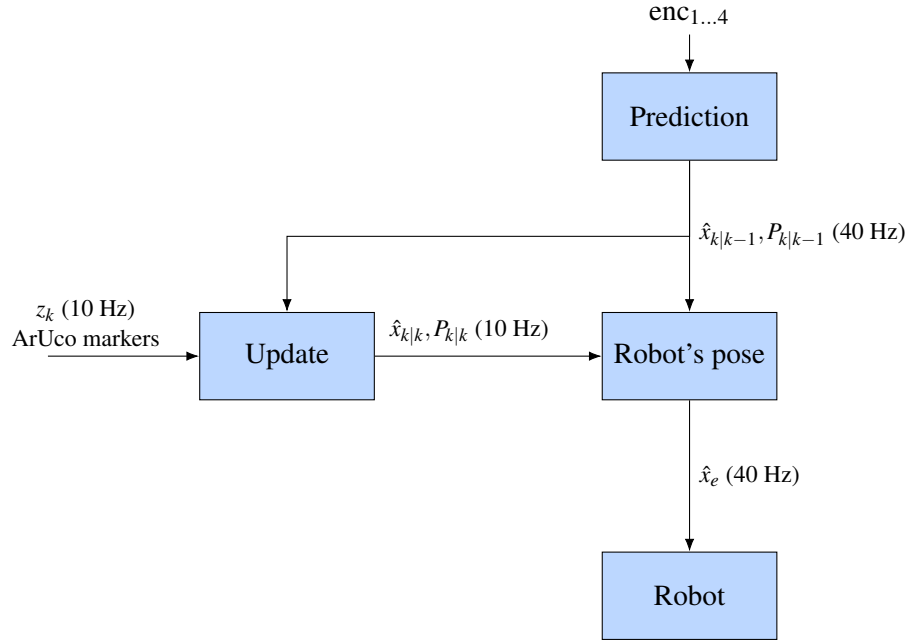


Figure 3.11: DFD for the EKF-based localization using ArUco markers as absolute reference sensors in the AMR

Camera measurements for EKF innovation

For the innovation, the robot's frontal camera and the ArUco Markers on the factory floor were chosen as the primary source of absolute localization. In this reasoning, as discussed in Chapter 2, fiducial markers are widely present in localization approaches for mobile robots, especially in indoor environments. In the context of the RAF, the ArUco fiducial markers displaced across the factory floor and on the walls of the machines are the primary references for localization in the scenario.

In the image processing stage for the pose estimation of the detected fiducial markers relative to the camera, the ArUco module from OpenCV [202] was used.

The used ArUco markers have known sizes. Thus, the OpenCV library can determine the distance and orientation between the camera and the tag according to the camera's intrinsic parameters and the position occupied by the fiducial marker's centroid in its 2D projection on the image, represented by a point estimated in pixels. Each fiducial marker has a unique identification ID, which the library can also decode. In addition, a database in the AMR has stored the absolute pose of each ArUco tag on the competition scene floor, i.e., its position and orientation relative to the center of the factory floor, the space frame. Whenever a marker is detected and has its ID decoded, the algorithm derives the translational and rotational vectors of the marker relative to the camera. Afterward, the camera's pose relative to the space frame is computed, as exemplified by Equation 3.3.

$$T_c^s = T_m^s \cdot (T_m^c)^{-1}, \quad (3.3)$$

where T_m^c represents the homogeneous transformation of the fiducial marker frame relative to the camera frame. T_m^s represents the absolute position of the fiducial marker relative to the space frame. Finally, T_c^s represents the camera pose relative to the space frame. In addition, another homogeneous transformation is necessary to displace the estimated camera's absolute pose to the robot's centroid, which is simple after the external calibration between the frames in the robotic platform is determined. Equation 3.4 shows the computations.

$$T_r^s = T_c^s \cdot T_r^c, \quad (3.4)$$

where T_r^s is the homogeneous transformation matrix of the robot frame to the space frame, T_c^s , the transformation of the camera frame to the space frame, and T_r^c , the transformation matrix from the robot frame to the camera frame.

Update Stage Filters

The Kalman Filter and the [EKF](#) use a series of measurements affected by statistical noise to generate improved estimates compared to those derived from a single measurement. In the update stage, additional filtering techniques can be introduced to enhance performance by managing observational errors effectively. This is crucial as the [EKF](#)'s covariance may diverge during the prediction phase but should converge upon observation integration, assuming the observations are accurate. Wrong observations, however, can degrade system performance significantly by misleading the filter's estimation convergence. In this reasoning, two non-linear filters were integrated into the camera measurements before being integrated as innovations in the [EKF](#).

Theoretically, it is reasonable to hypothesize a covariance model that states the accuracy of the ArUco measurements is non-linearly correlated with the depth and the angular offset from the camera's optical axis. Also, it is possible to derive an analytical model [203]. However, this would imply extensive testing to validate the model numerically for the former approach where the latter was not feasible in the time window when the system was developed, i.e., time was short for the [RAF](#) competition. Thus, the threshold for the following filters was found via sensitive analyses.

Euclidean Distance Filter

Measurements derived from ArUco tags are scrutinized based on their Euclidean distance from the robot's center, with a threshold set at 0.71 meters, to determine their acceptability for innovation insertion in the filter. In this reasoning, measurements farther than this threshold were deemed inappropriate to be used as observations in the filter. This distance threshold helps manage the non-linear and non-monotonic deterioration in pose estimation accuracy, particularly affecting the orientation estimation (θ) at varying distances and angles relative to the tags.

Mahalanobis Distance Filter

The Mahalanobis distance is used to measure the deviation of new measurements from the state estimates distribution, considering the correlations within the dataset. This filter identifies outliers

by considering how observations fit into the overall distribution shaped by the [EKF](#)'s current covariance matrix ($\mathbf{P}_{k|k-1}$). Observations with a Mahalanobis distance exceeding 0.08 are deemed outliers and are excluded from the update process. The distance is computed as

$$d_M(\mathbf{y}, P) = \sqrt{(\mathbf{y} - \boldsymbol{\mu})^T \mathbf{S}^{-1} (\mathbf{y} - \boldsymbol{\mu})}$$

where d_M is the mahalanobis distance for a given point $\mathbf{y}$ in the probability distribution P , with mean $\boldsymbol{\mu}$, and covariance matrix $\mathbf{S}$ which in this context is $\mathbf{P}_{k|k-1}$. Measurements are processed through both filters to determine their validity before being included in the update stage as observations. Algorithm 11 displays this procedure as a pseudocode.

Algorithm 11 Observations filtering - EKF's update stage

```

1: constant PLANAR_THRESHOLD  $\leftarrow$  0.71
2: constant MAHALANOBIS_THRESHOLD  $\leftarrow$  0.08
3: procedure FILTERUPDATECALLBACK(observations)
4:   for each observation in observations do
5:     Compute planar distance to robot center
6:     Compute Mahalanobis distance to filter state
7:     if planar distance  $\leq$  PLANAR_THRESHOLD and Mahalanobis distance  $\leq$  MAHALANOBIS_THRESHOLD then
8:       Update filter with observation
9:     end if
10:   end for
11: end procedure

```

These approaches maintain system reliability and stability by preventing erroneous states from misleading the filter during critical operational phases. In this reasoning, several strategies have been explored to optimize the integration of observations into the [EKF](#) [117]. The sequence and method by which observations are processed, specifically when multiple measurements occur within the same image frame, may impact the filter's performance. This is especially critical in scenarios involving precise and less accurate observations. One approach processes observations sequentially as they are detected, whereas another strategy sorts them by increasing Euclidean distance, which could influence how measurements affect the state estimation of the system. In addition, alternative methods such as integrating observations using a multivariate Gaussian [203] or employing an arithmetic mean might also be considered.

The integration of incorrect observations, particularly in a diverged filter state, can cause the [EKF](#) to erroneously converge to these incorrect states, potentially leading to prolonged periods of inaccurate state estimation until the filter's prediction capabilities adjust the covariance to more realistic values, allowing for the acceptance of new observations. In this situation, the worst can happen where the Mahalanobis filter negatively impacts the filter's ability to recover from flawed estimations. New potential good estimates will not be accepted because the filter converges to flawed estimations.

Despite the different testing, it was concluded that it was not worth the effort to develop a complex measurement integration into the update stage for the competition scenario itself because the gains were insignificant, and time was of the essence. Therefore, the approach to integrate the measurements as innovations in the [EKF](#)'s update stage was to insert them as they were detected

as long as they passed through both filters. This was found to be adequate for the reliable operation of the filter and robot.

Since this localization system was explicitly designed for the competition and as a learning endeavor for the next steps of the PhD, the results of this localization system will not be displayed to avoid confusion about the message this thesis is trying to portray. Therefore, the reader is suggested to read the research works [117] and [204], which were published in an international indexed journal and conference, to check the results of this localization system, which achieved millimetric accuracy and precision.

3.1.6 Path Planning Module

The path planning module is an essential component of [AMRs](#), facilitating effective navigation and interaction within an operational environment. This subsection elaborates on the path planner's design developed for the [RAF](#) competition using high-level pseudocode as an auxiliary resource for describing the module.

Design overview

The path planning component is central to the robot's navigation capability and is designed to manage and execute movements and interactions based on initial configurations. Initialized with the robot's pose and task-related data of the competition, that is, the type and order of boxes, the module systematically generates waypoints that guide the robot to perform tasks associated with the color-coded boxes.

The path planning logic is embedded within a graph-based structure where nodes correspond to physical markers in the environment (ArUco markers), and edges represent feasible paths between these markers. Each edge is assigned a weight, the Euclidean distance between connected nodes, facilitating the application of shortest-path algorithms. Algorithm 12 provides an abstract representation of the path-planning process.

Algorithm 12 Path planning module

```

1: Input: initial_pose, boxes
2: Output: full_waypoints
3: function PATHPLANNER(initial_pose, boxes)
4:   Initialize path planner with robot pose and box positions
5:   Sort tasks based on box colors
6:   Initialize navigation graph and waypoints
7:   for each task in sorted_tasks do
8:     Determine box color and entry point
9:     Plan route to entry point
10:    Add box pickup sequence
11:    if box is blue then
12:      Plan direct route to drop-off point
13:    else if box is green then
14:      Plan route to processing station
15:      Plan route from processing station to drop-off point
16:    else if box is red then
17:      Plan route to first processing station
18:      Plan route to second processing station
19:      Plan route to the drop-off point
20:    end if
21:    Add box drop-off sequence
22:    if not last task then
23:      Plan route back to central point
24:    end if
25:  end for
26:  return full_waypoints
27: end function

```

▷ All planned waypoints for the robot's path
 ▷ Path planning algorithm is a modified Dijkstra's algorithm
 ▷ Blue, Green, Red order
 ▷ Entry point: ArUco marker near box
 ▷ Drop-off: ArUco 23-26
 ▷ Processing: ArUco 13
 ▷ First: ArUco 10
 ▷ Second: ArUco 13
 ▷ Central: ArUco 22

Implementation details

Within Algorithm 12, the robot is configured with its initial pose and tasks are associated with specific colors of boxes. The tasks are prioritized and processed in sequence, each involving navigation to a task location, manipulating objects, and proceeding to the following designated point.

The Dijkstra algorithm was selected for path planning due to its straightforward implementation and suitability for environments of constrained size, such as the factory floor considered in this study. Given the operational area's relatively modest scale and complexity, the more computationally intensive algorithms did not add value, and more optimized algorithms such as A* [205]. Consequently, Dijkstra's algorithm balances computational efficiency and implementation simplicity, adequately meeting the requirements of the specified application context.

3.1.7 Trajectory Planning

This section details the trajectory planning methods implemented in the robot for achieving precise positioning and rotation control. Two main procedures are described: Algorithm 13, which controls the robot to navigate to a target location with a desired orientation, and Algorithm 14, which controls the robot's orientation to a specific angle without its displacement.

Algorithm 13 is designed to enable an omnidirectional robot to navigate to a specific location defined by coordinates (x, y) and a desired orientation while it is performing this trajectory. The robot adjusts its linear and angular speeds based on the calculated distance and orientation difference to the target location.

Algorithm 13 Trajectory planning for position control

```

1: Input:  $target\_x, target\_y, max\_speed, target\_angle, max\_angular\_speed, Kp\_position, Kp\_angle$ 
2: Output: robot speed commands
3: procedure PLANTRAJECTORY
4:   Calculate distance to target
5:   Compute desired linear speed ▷ P controller:  $speed = Kp\_position * distance$ 
6:   Limit linear speed to maximum speed
7:   Calculate heading angle to target
8:   Decompose speed into x and y components
9:   Transform speed to robot's local coordinate system
10:  Calculate angle error ▷ Difference between target and current angle considering angle wrapping
11:  Compute angular speed ▷ P controller:  $angular\_speed = Kp\_angle * angle\_error$ 
12:  Limit angular speed to maximum
13:  return robot speed commands
14: end procedure

```

Figure 3.12 shows a sketch of the AMR kinematics and the pertinent variables for this trajectory controller. In the illustration, the heading angle is r_{Angle} where “front” is where the robot’s front part will face when performing the trajectory, $[v, v_n, \omega]$ are the robot’s speed states, $[X_R, Y_R]$ its frame, and θ the robot’s orientation.

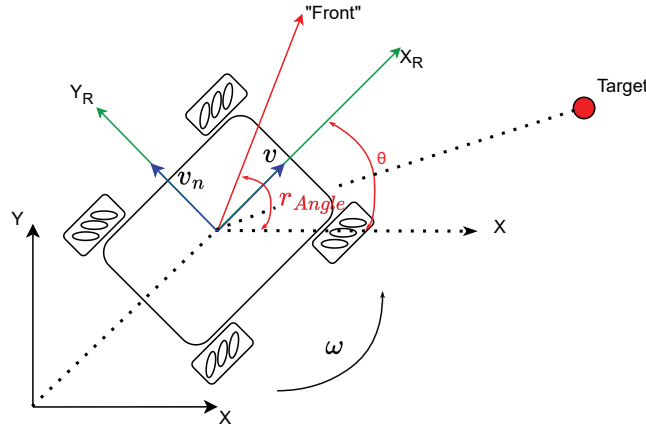


Figure 3.12: GotoXY trajectory controller

On the other hand, the Algorithm 14 procedure controls the rotational aspect of the robot, aligning it to a desired orientation. It adjusts the robot’s angular speed to smoothly rotate to the specified angle without affecting the translational movement. It is assumed that the midpoint of the wheels coincides with the robot’s centroid, so this method generates speed states for the robot to rotate along its axis of rotation.

The parameters for both controllers were adjusted through sensitive analysis because of the circumstances already mentioned in this chapter of the RAF competition. Therefore, a systematic controller tuning method was not employed due to the research’s focus not aligning with such methodologies and the limited time available before the competition.

Algorithm 14 Trajectory planning for rotation control

```

1: Input:  $Kp\_angle, target\_angle, max\_angular\_speed$ 
2: Output: robot speed commands
3: procedure PLANROTATION
4:   Calculate angle error                                ▷ Difference between target and current angle, handling wrap-around
5:   Set linear speeds to zero                             ▷ Robot should only rotate, not translate
6:   Compute angular speed                               ▷ P controller:  $angular\_speed = Kp\_angle * angle\_error$ 
7:   Limit angular speed to maximum
8:   return robot speed commands
9: end procedure

```

3.2 Development of the Ground Truth System

The indoor localization of AMRs in dynamic I4.0 environments presents significant challenges, especially in the absence of GPS signals [39] and several obstructions. A highly accurate and reliable ground truth system is essential to validate novel AI-enhanced localization approaches in such environments. A previous methodology, the Local Marker Estimation Algorithm (LMEA) introduced in [117], proved insufficient for this purpose due to its reliance on three fiducial markers to form an orthonormal basis for localization. Its effectiveness is compromised in dynamic scenarios with visual obstructions, and its dependence on a few anchor markers makes it vulnerable to installation errors.

Therefore, this section introduces the Kabsch Marker Estimation Algorithm (KMEA), a robust ground truth system designed to provide highly accurate reference measurements for validating the core AI-enhanced localization system proposed in this thesis. Using all visible markers within the environment, KMEA significantly improves robustness against visual obstructions. The algorithm operates without requiring camera extrinsic parameters, and it also estimates these parameters. It remains less sensitive to marker misplacement during workspace adaptation by seeking the optimal transformation matrix between real-world markers and their image plane counterparts.

The system incorporates the Kabsch Algorithm to align detected fiducial markers with their known positions, achieving the high precision necessary for a ground truth system. Rigorous testing in an I4.0 automated warehouse mockup demonstrates its superiority over LMEA, with a Mean Relative Error (MRE) of approximately less than 1.4% relative to the factory mockup's largest dimension. This accuracy, combined with the algorithm's flexibility and error resilience using cost-effective hardware, establishes KMEA as an ideal reference system while suggesting potential for broader applications in augmented reality and visual positioning systems.

The main contributions of this ground truth system are:

- Development of the KMEA, integrating the Kabsch-Umeyama Algorithm for high accuracy and precision, addressing the core limitations of LMEA and establishing a robust validation solution.
- Establishment of a solid theoretical framework with thorough testing methodology, ensuring reliability as a reference for subsequent validation of localization approaches.

- Demonstrated effectiveness in an [I4.0](#) automated warehouse mockup, establishing its potential usage as a standalone multi-marker localization system.

3.2.1 System Architecture

The research work's system architecture is designed around a Raspberry Pi 5, strategically positioned to monitor the factory floor, as illustrated in Figure [4.7](#). This device serves as the central node for the detection of ArUco markers. It has a Raspberry Pi Camera Module 2 that continuously records video frames. These frames are subsequently analyzed to identify ArUco markers within its [Field of View \(FoV\)](#) and to estimate their relative poses. This pose estimation process uses the intrinsic parameters of the camera and distortion coefficients to calculate the position and orientation of each marker detected.

In this setup, the Raspberry Pi executes a Python script that uses OpenCV for image processing and ZeroMQ for networking. This script processes the captured video frames by converting them to grayscale, detecting ArUco markers, and calculating their poses. The data, including the estimated poses and marker IDs, are packaged into a packet. This packet is transmitted to a computer application through a ZeroMQ socket for further processing or analysis.

The computer application, represented in Figure [3.13](#), also developed as part of the research work, receives the data packets and further processes the pose information to obtain several metadata and the absolute poses of every detected marker. It can process the position of markers individually or as a pair. The latter option is to enhance the position estimation stability since computing the pose of the midpoint of a pair of markers is more stable than single markers since they are prone to measurement and quantization errors due to low resolution, illumination, and other external factors. It also provides orientation estimation derived from the pair of markers, which is more stable than estimating from one marker individually [[29](#)]. In this research context, these selected markers, or pairs, represent [AMRs](#) in the factory. Thus, based on the configuration settings, the application can also relay the pose information to an [AMR](#) or a fleet of robots, facilitating real-time interaction and navigation on the factory floor. The data sent to the robots is formatted as a series of key-value pairs representing the robot pose's coordinates.

In Figure [3.13](#), three colored rectangles delineate four sections of the software interface. The area inside the dark blue rectangle presents the interface's section related to the communication between the Raspberry Pi 5 system, the software running on the computer, and the IP of the [AMR](#) that will receive the pose data. This part also presents metadata relative to the current algorithm being executed, such as the average and maximum error of all markers' pose estimation and the current pose of the tracked marker or the pair of markers. Additionally, several types of tabs are delineated by the green rectangle, which modifies the content presented in the complement area of the total interface. The user can check debug information in the debug tab; all the ArUcos' markers ground truth pose or their estimated pose, if they are being detected, the error, and other types of information in the Arucos tab; the configuration tab, currently presented in the illustration, one can choose several different configurations for the detection algorithm, filters, the output method, which IDs to track, and other options. The log tab is responsible for recording the data stream

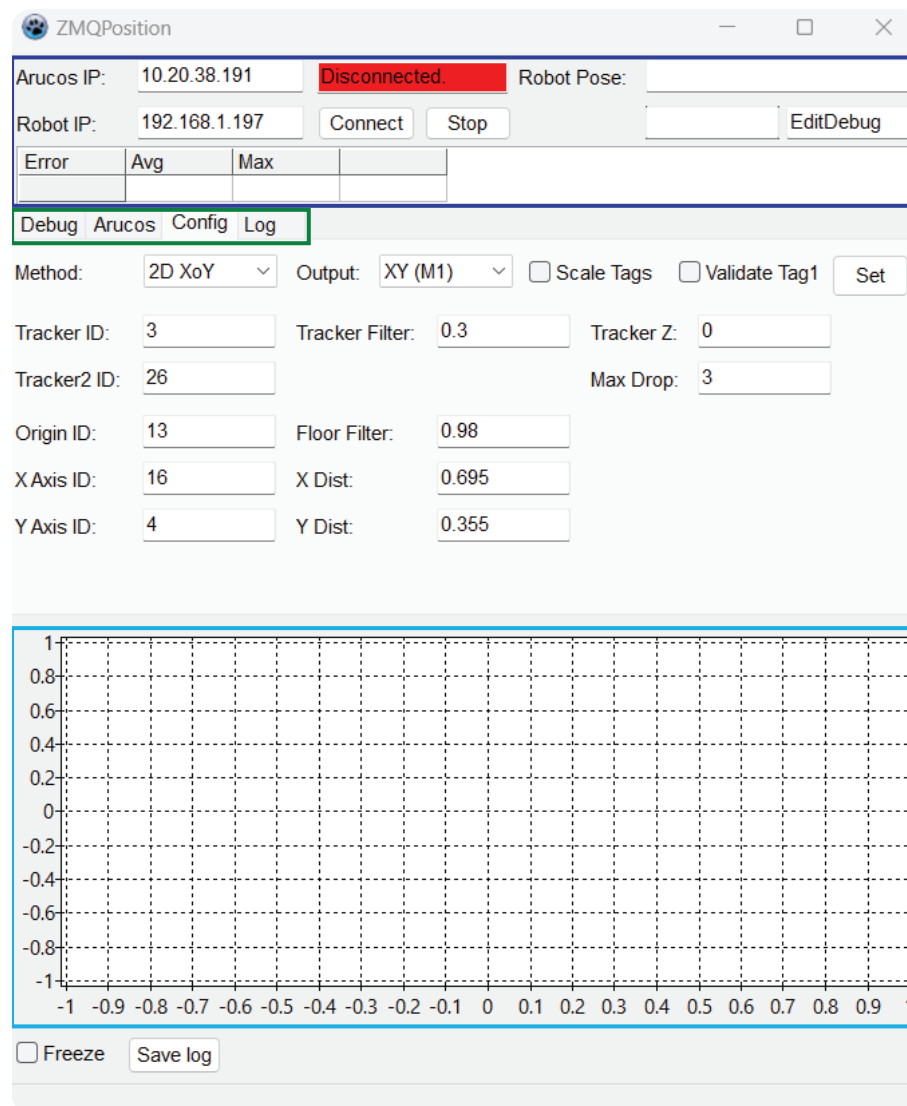


Figure 3.13: ArUco markers detection software's interface

for retrospective usage, ensuring that the system is not just limited to real-time operations but also contributes to data-driven insights and improvements in subsequent iterations. It also allows storing the absolute pose data relayed to the robots in CSVs' files. Finally, the rectangle in light blue represents the 2D Cartesian graph that plots all the estimated absolute markers' poses and their respective error vectors (vectors that point in the direction of the markers' real poses), where their magnitude represents the error. Figure 3.14 displays a DFD summarizing how the system interacts with external and internal entities.

In Figure 3.14, relative pose data refers to the markers' relative pose to the camera's frame, and absolute pose data refers to the markers' pose relative to the world frame.

This section employed two distinct algorithms for estimating the positions of ArUco markers within the camera's FoV, a critical aspect for accurate and reliable localization in robotic applications. The algorithms are LMEA and KMEA. The latter is the primary focus of our validation

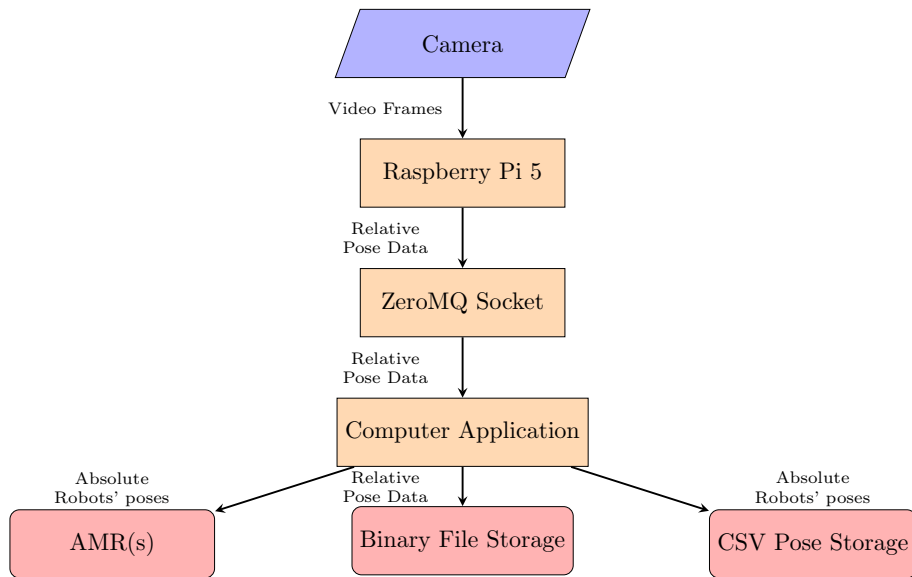


Figure 3.14: Data flow diagram describing the system’s data flow during the pose estimation process

efforts in this study due to its robustness, while the former has been validated in previous work [117]. It is important to note that both algorithms assume that the frames have square-shaped pixels for scaling purposes.

3.2.2 The Local Marker Estimation Algorithm

The **LMEA** transforms the poses of detected ArUco markers from the camera’s coordinate system to a world coordinate system. It leverages three reference markers or anchors — the origin, X -axis, and Y -axis markers. **LMEA** computes a scale factor based on the known distance along the X -axis, which is the X -axis anchor position, computes a rotation matrix for alignment, and applies these transformations to all detected markers. For an in-depth description and validation of the algorithm, readers are directed to [117].

The **LMEA** proceeds in three main stages:

1. **Scale Calculation:** Computes a linear scale factor using the known distance $XDist$ and the camera-estimated distance between the origin and X -axis markers.
2. **Translation and Initial Scaling:** Scale and translate the positions of all detected markers relative to the origin.
3. **Rotation Adjustment:** Calculates a proper orthonormal basis that represents the world’s coordinate frame with the versors derived from the scaled anchor markers and composes a rotation matrix with it. Next, it applies the rotation matrix to align all markers relative to the origin.

For a comprehensive understanding of the algorithm, Figure 3.15 shows the process of one iteration of the LMEA, where the legend is presented below the algorithm's inputs, outputs, constants, methods, and variables. Soon after, Algorithm 15 details its process.

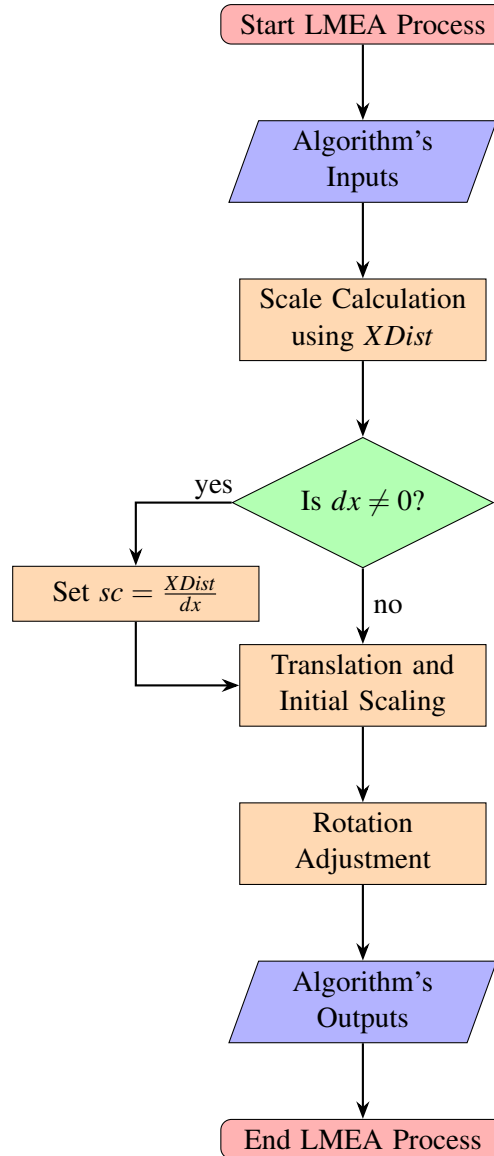


Figure 3.15: Flowchart of the LMEA process

Algorithm's inputs

- All ArUco Markers' relative poses to the camera.
- Ground truth pose of three ArUco markers anchors to form an orthonormal basis.

Algorithm's outputs

- All ArUco Markers' estimated absolute poses (configurable).

- Robots' poses (configurable).

Variables & Methods

- *sc*: Scalar is used to adjust the real positions of the markers in the world's frame.
- *dx*: Norm of the vector pointing from the origin to the *X*-axis marker anchor in the camera's coordinate system.
- *pO*: Origin point of the world's frame.
- *pX*: *X*-axis marker anchor point in the world's frame.
- *XDist*: Real *x*-axis distance of the *X*-axis marker anchor.
- *MkDiff*: Translation operation relative to the origin marker.
- *originID*: Index for the origin anchor marker.
- *Mks[i].Pcam*: Pose of the *i*-th marker in the camera's coordinate system.
- *Mks[i].Pw*: Pose of the *i*-th marker in the world's coordinate system.
- *[VrX, VrY, VrZ]*: $[x, y, z]$ -axes versors of the world's coordinate frame.
- *XAxisID*: Index of the *X*-axis anchor marker.
- *YAxisID*: Index of the *Y*-axis anchor marker.
- *R*: Rotation matrix for aligning the coordinate systems.
- *V3N()*: Computes the magnitude of a vector based on two points *p1* and *p2*.
- *V3ExtP()*: Method to compute the external product between two 3D vectors.
- *calcRM()*: Method to compute the 3D rotation matrix given a set of versors in $[x, y, z]$.
- *Vr()*: Method to compute a unit vector from the origin to the point *p1*.

3.2.3 The Kabsch Marker Estimation Algorithm

The **KMEA** advances ArUco marker-based localization by using all detected markers to compute the transformation matrix between the detected markers and their actual real-world positions. This approach improves pose estimation accuracy within a global coordinate system and addresses the challenges of indoor localization for **AMRs** in Industry 4.0 environments. By employing the Kabsch Algorithm, **KMEA** finds the optimal alignment of the poses of markers detected by the camera with their real-world coordinates. Notable features of **KMEA** include its operation without the camera's extrinsic parameters and its reduced sensitivity to marker misplacement.

The **KMEA** operates in three main stages:

Algorithm 15 Pseudocode - Local Marker Estimation Algorithm

```

1: procedure LMEAPROCESSTAGS
2:    $sc = 1$ 
3:    $dx = V3N(pO, pX)$ 

4:   if  $dx \neq 0$  then
5:      $sc = \frac{XDist}{dx}$ 
6:   end if

7:   for each eligible marker  $i$  do
8:      $MkDiff = Mks[i].Pcam - Mks[originID].Pcam$ 
9:      $Mks[i].Pw = sc \cdot MkDiff$ 
10:  end for

11:   $VrX = Vr(Mks[XAxisID].Pw)$ 
12:   $VrY = Vr(Mks[YAxisID].Pw)$ 
13:   $VrZ = V3ExtP(VrX, VrY)$ 
14:   $VrY = V3ExtP(VrZ, VrX)$ 

15:   $R = calcRM(VrX, VrY, VrZ)$ 

16:  for each eligible marker  $i$  do
17:     $Mks[i].Pw = R^T \cdot Mks[i].Pw$ 
18:  end for
19: end procedure

```

1. **Scale Calculation:** First, [KMEA](#) proceeds to iterate over all detected markers and compute the mean absolute distance of every marker relative to the origin marker. This computation is performed for the markers' camera-detected poses and their ground-truth poses. Subsequently, a scale factor is computed by the ratio of these two mean absolute distances. Finally, the camera's markers poses are scaled with the calculated scale factor.
2. **Optimal Rotation Matrix Calculation:** The next step of the algorithm is the computation of the optimal rotation matrix between the two sets of points. Therefore, two matrices of the Kabsch Algorithm, P and Q , are populated with the camera's scaled markers poses and their ground-truth counterparts, respectively. The markers used in these matrices are identified as eligible after being checked for computation suitability. Afterward, the z coordinate of the ground truth markers poses is injected with a random noise with mean 0 and amplitude $[-0.5, 0.5]$, scaled by a factor of 0.0001. This operation is done to avoid degeneracies during the computations of the Kabsch Algorithm that comes with the possible coplanarity or colinearity of the markers. Soon after, the optimal rotation matrix between the two sets of points is computed, R , with the centroids, $[Po, Qo]$ of the matrices P and Q , respectively. The determinant of R is checked if it is -1 ; if so, the z -axis of the rotation matrix is flipped to ensure proper rotation. For more information about the Kabsch Algorithm, the readers are referred to [\[206\]](#).
3. **Coordinate frame transformation:** The algorithm's last step comprises two translations and one rotation (not in this order). First, each camera's scaled detected marker is translated to its origin marker in the camera frame (Po), and then it is rotated to the world's frame. Finally, it is translated to the world's origin marker position (Qo). A side feature of this approach is that if one considers the camera frame pose, which is the origin in its frame,

as an “tag”, after the transformations, the camera pose will be transformed and estimated relative to the newly considered world frame, which is its extrinsic parameters.

For a comprehensive understanding of the algorithm, Figure 3.16 shows the process of the algorithm with the algorithm’s legend presented below.

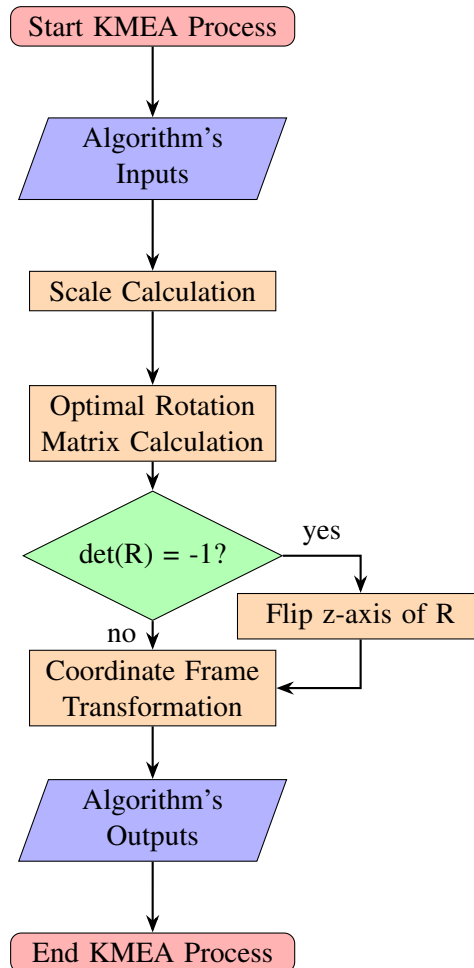


Figure 3.16: Flowchart of the **KMEA** process

Algorithm's inputs

- All detected ArUco Markers' camera poses.
- Ground truth poses of all ArUco Markers.
- Camera pose as origin in the camera frame.

Algorithm's outputs

- All ArUco Markers' estimated absolute poses (configurable).

- Robots' poses (configurable).
- Camera's extrinsic parameters in the considered world frame after transformation. In this context, it was considered Marker 13 for convenience.

Variables & Methods

- N : Counter of eligible markers.
- $OriginID$: The origin ArUco marker index.
- sc : Scaling factor for aligning detected marker positions with real-world coordinates.
- $Mks[i].Pcam$: Camera-detected pose of the i -th marker.
- $Mks[i].Pw$: World pose of the i -th marker.
- $Mks[i].Preal$: Ground truth pose of the i -th marker.
- dx, drx : Mean absolute distances from the origin to the markers in camera and real coordinates.
- P, Q : Matrices with scaled world camera and real positions of eligible markers, respectively.
- R, Po, Qo : Optimal rotation matrix and the centroids from the matrices P and Q , respectively.
- $ILeft$: 3D identity matrix with the element $a_{2,2}$ flipped to reflect the z -axis.
- $\det()$: Method that returns the determinant of a matrix.
- $V3N()$: Method to compute the norm of a 3d vector.
- $Rnd()$: Random number generator from $[0, 1)$.
- $OptRotMat()$: Kabsch-Umeyama Algorithm that returns the optimal rotational matrix between the sets of points of two matrices.

In this reasoning, Algorithm 16 describes the inner details of the algorithm's logic.

3.2.4 Theoretical Advantages of the KMEA Over the LMEA

The KMEA demonstrates several theoretical advantages over the LMEA. Key benefits include enhanced robustness, superior alignment between coordinate systems, independence from specific markers, less sensitivity to marker misplacement, a comprehensive global optimization strategy incorporating all visible markers for more precise estimations, and independence from the camera's extrinsic parameters. It is essential to note that these algorithms' efficacy largely depends on the accuracy of the camera calibration. Meticulous calibration can mitigate errors influenced by distortions during data acquisition for these algorithms. Thus, this study's camera underwent a

Algorithm 16 Pseudocode - Kabsch Marker Estimation Algorithm

```

1:  $N = 0$ 

2: for each eligible marker  $i$  do
3:    $dx = dx + \sqrt{3}N(Mks[i].Pcam - Mks[OriginID].Pcam)$ 
4:    $drx = drx + \sqrt{3}N(Mks[i].Preal - Mks[OriginID].Preal)$ 
5:    $N = N + 1$ 
6: end for

7: if  $N \neq 0$  then
8:    $dx = \frac{dx}{N}$ 
9:    $drx = \frac{drx}{N}$ 
10:  if  $dx \neq 0$  then
11:     $sc = \frac{drx}{dx}$ 
12:  end if
13: end if

14: for each eligible maker  $i$  do
15:    $Mks[i].Pw = sc \cdot Mks[i].Pcam$ 
16:    $P[i, :] = Mks[i].Pw$ 
17:    $Mks[i].Preal[2, 0] += 1e - 4 * (Rnd - 0.5)$ 
18:    $Q[i, :] = Mks[i].Preal$ 
19: end for

20:  $[R, Po, Qo] = \text{OptRotMat}(P, Q)$ 

21: if  $\det(R) < 0$  then
22:    $R = R * ILeft$ 
23: end if

24: for each eligible marker  $i$  do
25:    $Mks[i].Pw = R^t \cdot (Mks[i].Pw - Po^t) + Qo^t$ 
26: end for

```

standard calibration technique, and the calibration results were acceptable. Consequently, no complex filters or outlier rejection algorithms were developed, except for a first-order infinite impulse response filter applied to both algorithms to mitigate noise.

3.2.5 Integration with AMRs

The robot's pose can be estimated individually or with a pair of markers, depending on the configuration of the software. When configured for individual marker pose estimation, the algorithms directly take the pose of a single specified marker as the robot's pose. In the case of a pair of markers, the algorithms calculate the geometric midpoint between the two markers to determine the robot's position. Also, the orientation is orthogonally calculated based on the line connecting the two markers. Thus, this method provides redundancy and can be more robust, as it does not rely on a single marker for pose estimation. In particular, it stabilizes the estimation of the orientation.

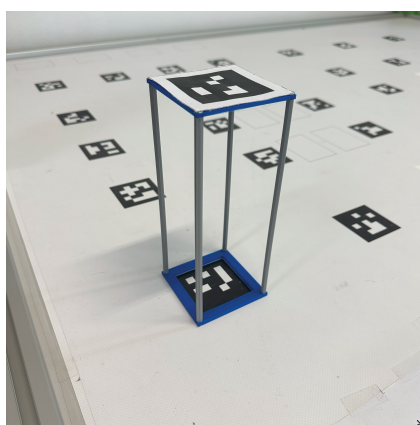
In addition, this algorithm can also be used individually as a localization system for robots. As mentioned in this section, if the camera pose (origin) in its frame is inserted in the input of the algorithm, the output of the algorithm will transform its extrinsic parameters to be relative to the considered space frame. In this reasoning, if this camera is embedded on an [AMR](#), its pose can be estimated relative to a detected group of markers. One can think of the repercussions it can apply

in the industry, for example, carefully positioning a robot or robot arm to perform specific tasks despite its localization system; this system can help since it only needs a grid of fiducial markers.

Finally, as mentioned in the algorithms' descriptions, the output of the algorithms can be configured; that is, it can just estimate the markers that represent the AMRs in an industry factory, for instance. In this case, it was configured to estimate all visible markers.

3.2.6 Improvements to the Ground Truth System

This section describes enhancements to our ground truth system used within an automated 14.0-based factory environment. The system comprises a Raspberry Pi 5 paired with a Raspicam module v2 and runs at 30 Hz with images of 800×600 px of resolution. This configuration was designed to detect fiducial markers and estimate their pose. However, during the testing phase, it was observed that systematic errors emerged as the markers deviated from the center of the image. These errors were particularly noticeable when the markers' height, denoted as z , differed from the z -plane of the factory environment. This phenomenon was attributed to the camera's projective geometry effects (parallax) rather than lens distortion, especially considering the camera had been calibrated previously. Figures 3.17a and 3.17b show the effects of the projective geometry on the marker that is elevated by 20 cm from the factory's z -plane and with a 2D position $[u, v]$ deviated from the camera's image center.



(a) ArUco support of 20 cm of height put over the bottom-left marker of the floor



(b) Illustration of the parallax effect seen from the camera

Figure 3.17: Parallax's effect on the elevated marker perceived pose

As shown in Figure 3.17b, the marker's position significantly deviated from the top of the marker, where its support was placed. However, as seen in Figure 3.17a, the support is carefully placed centered and over to the bottom-left marker. Thus, a software-based compensation mechanism was implemented to address the identified parallax issue. The approach is detailed in Algorithm 17, "Z Correction Algorithm", which effectively adjusts for the height discrepancies between the robot's markers and the factory plane.

Algorithm 17 begins with an initial setup phase, taking a tracked marker's uncorrected world pose estimation as input, including the reference "marker" that determines the camera's position.

Algorithm 17 Z Correction Algorithm

```

1: Constants: maxArucoIds, trackerID, trackerZ ▷ Maximum number of tags, tracked tags' IDs, and the tracked tags' real heights.
2: Input: Tags[maxArucoIds].Pw ▷ Uncorrected world pose estimation of the tags
3: Output: RobotPos ▷ Compensated world pose of the tracked tags
4:  $CamPos \leftarrow Tags[maxArucoIds - 1].Pw$  ▷ Last index is camera pose, determined by KMEA
5:  $RobotPos \leftarrow Tags[trackerID].Pw$  ▷ It could be a group of tags that are tracked
6: if CorrectZ then
7:    $Vec \leftarrow RobotPos - CamPos$  ▷ Vector from the camera pose to the robot pose
8:   if  $Vec_{2,0} \neq 0$  then ▷ If z is not 0
9:      $\lambda \leftarrow \frac{trackerZ - CamPos_{2,0}}{Vec_{2,0}}$  ▷ trackerZ is the real height of the tracked tag
10:     $RobotPos_{0,0} \leftarrow CamPos_{0,0} + \lambda \times Vec_{0,0}$ 
11:     $RobotPos_{1,0} \leftarrow CamPos_{1,0} + \lambda \times Vec_{1,0}$ 
12:     $RobotPos_{2,0} \leftarrow CamPos_{2,0} + \lambda \times Vec_{2,0}$ 
13:   end if
14: end if

```

Crucially, the algorithm incorporates knowledge of the tracked markers' actual physical height (Z-coordinate). Then, a parametric line is established between the camera and the uncorrected marker position. Analyzing only the Z coordinate, the scalar λ is derived from this parametric line equation and represents the point along this line where the marker reaches its accurate physical height. By solving for λ using the known height of the marker, this scalar value is then applied to the parametric equations for all coordinates, yielding the true three-dimensional position of the tracked marker in the space frame.

This method produces a more accurate overall position estimate by maintaining the geometric relationship between all coordinates while enforcing the known physical constraint of marker height. It effectively addresses the parallax-induced errors inherent in monocular vision-based positioning systems, resulting in accurate and precise spatial measurements.

3.3 AI-Enhanced Indoor Localization System

This section will describe the AI-enhanced indoor localization system in its entirety, **RoPM**. It includes the system architecture, **DFD** and modes of operation, the core of the localization system itself, **PM**, the data fusion method, and the outlier rejection methodology. The objective of the localization system is to accurately determine the robot's position within a predefined environment by matching **LiDAR** scans to a reference map stored in the system's memory. This process, however, faces significant challenges due to the presence of outliers in dynamic environments.

Effect of Outliers on Localization Accuracy

The **RoPM** fundamentally relies on the assumption that **LiDAR** scans reflect off static structures, such as walls, as initially mapped in the reference's map. Figures 3.18a and 3.18b illustrate the discrepancies that arise when outliers are present. In Figure 3.18a, **LiDAR** points are "skewed" by reflecting off temporary obstacles or moving objects, deviating from their expected trajectories based on the static scenario depicted in Figure 3.18b. These deviations result in erroneous projections against the reference map, where points expected to hit stable walls instead reflect off transient objects.

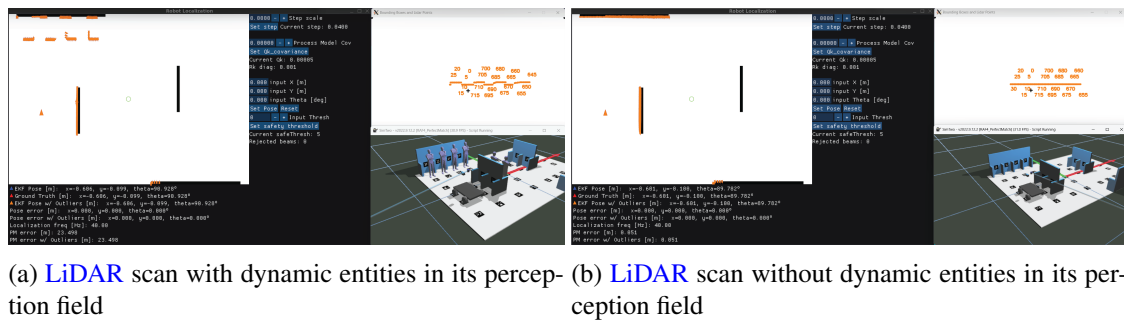
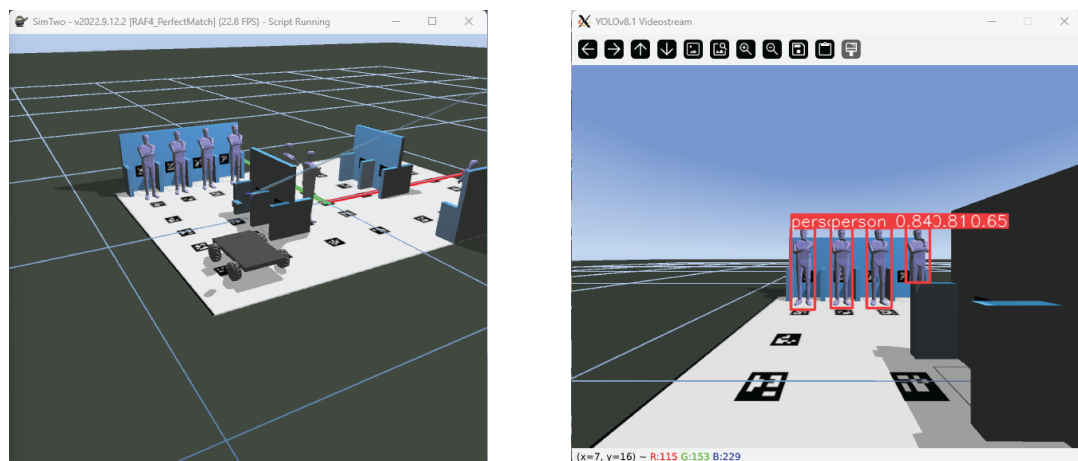


Figure 3.18: Effects of dynamic entities in the perception field of the LiDAR scanner

By adjusting the robot's pose, the localization algorithm minimizes the distance between each projected LiDAR point and the nearest wall in the reference map. When outliers are present, these points are mistakenly projected onto the outliers, not the expected surfaces, distorting the pose estimation process. This misalignment can significantly impair the system's ability to localize the robot accurately, leading to potential navigational errors.

Integration of Object Detection in Localization

To address these challenges, the system employs an object detection framework that semantically interprets objects in the environment, determining their relevance to the localization process. In this sense, Figures 3.19a and 3.19b show the robot looking at the incoming warehouse where there are outliers. The latter shows the image frame with the semantic interpretation of the scene, and the former is the 3D world scene of the simulator so that it can be understood where the robot was during the screenshot.



(a) World scene when the processed frame was taken (b) Processed frame with CNN to detect outliers

Figure 3.19: Simple test showing the robot observing with the camera the outliers

From the perspective of the localization system, what matters are the dimensions in the image frame of the outliers and if LiDAR beams are reflecting from them. Therefore, Figure 3.20 visually represents what the AI-enhanced localization system sees to proceed with its decision-making.

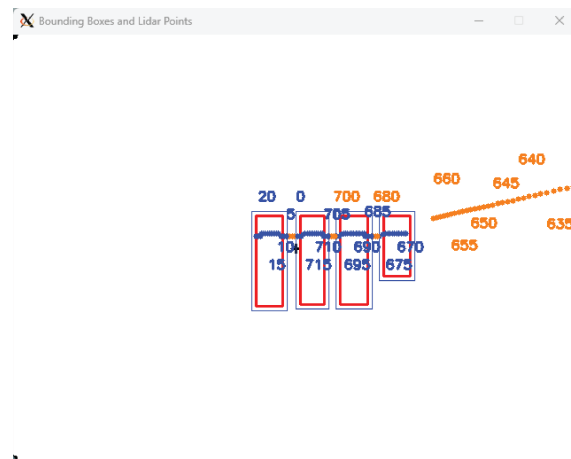


Figure 3.20: Localization system’s visual representation of the detected objects (blue and red boxes) and **LiDAR**’s projected measurements (orange and blue dots) in the image frame. Red boxes represent the actual dimensions of the objects, and the blue ones have a safety threshold. The blue dots are the rejected **LiDAR** beams, and the orange ones are the valid ones. Finally, the number labels are the beams’ indexes for reference purposes.

The illustration shows that the object detection algorithm identifies and categorizes objects into bounding boxes. Points within the red bounding boxes, illustrated in blue, represent detected outliers. These are distinguished from the valid **LiDAR** points, shown in orange, which align with the static elements of the environment. The localization algorithm discards the lidar points considered outliers. This selective rejection is crucial for maintaining the integrity of the localization process, ensuring that only pertinent environmental features influence the pose estimation. By dynamically filtering out irrelevant or transient objects, the system maintains high accuracy and precision in real-time navigation, even in environments with variable and unpredictable elements. This approach enhances the reliability of the localization system. The subsequent sections will detail the implementation and effectiveness of this object detection integration.

3.3.1 System Architecture

Our robotic processing system proposal is designed to handle tasks efficiently. For that, a C++ application primarily powers the system, while the object detection block runs in Python to leverage the YOLOV8.2 model from Ultralytics, which exclusively supports Python². Two high-level **DFDs** were developed, depicted in Figures 3.21 and 3.22, to facilitate understanding. The former illustrates the highest level abstraction of the system architecture, showing which devices communicate with each other. The latter displays a more detailed view of the component interactions that run inside these devices. In the latter, if a term has “Out.” in the name, it corresponds to the localization system with outlier rejection, i.e., **RoPM**.

As it is possible to see in Figure 3.21, it displays the communication between the devices when the **AMR** is operating. As described in this chapter, the ground truth system runs in a Raspberry

²<https://www.ultralytics.com/>

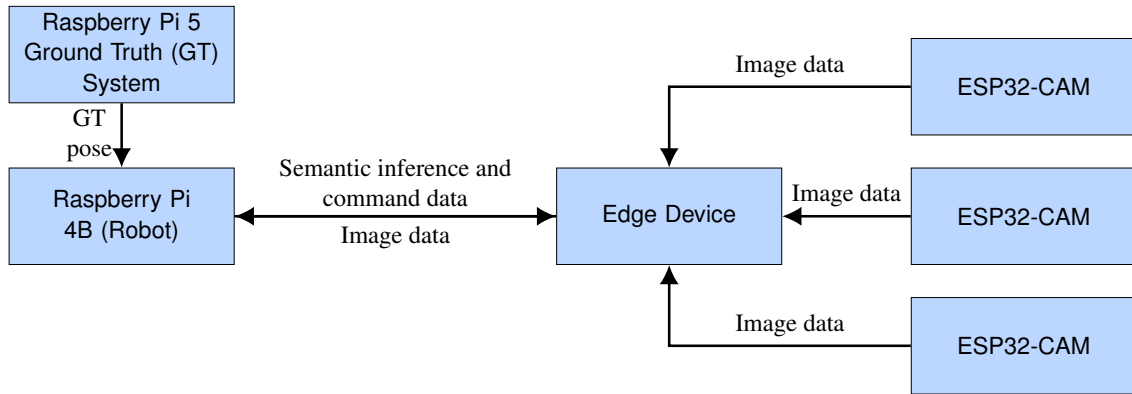


Figure 3.21: Communication diagram between system devices

Pi 5 in the assembled test bed. It sends ground truth 2D pose data (GT pose) to the robot for validation. The robot sends image data to the edge device to process semantic information from them and the three ESP32-CAMs embedded on the robotic platform. The Edge Device processes them asynchronously and sends the semantic inference data for the AMR for decision-making. In addition, optionally, it can send remote control data to control the robot via a gamepad.

With the broad overview of the system functionality in the previous diagram, the enhanced localization software can run in “Online” or “Offline” mode. Figure 3.22 represents the “Online” behavior of the system, where it happens in real time.

The terms’ meanings, depicted in Figure 3.22, are described below:

- Encs: encoder readings 1-4;
- RGB: Imagery data from the RGB camera;
- Inference metadata: data produced by the YOLO model;
- GT: ground truth pose data, for debugging;
- LiDAR: a complete LiDAR scan of 720 beams and 360 degrees of range;
- EKF μ : EKF’s mean, which corresponds to the filtered state variables $\hat{x}, \hat{y}, \hat{\theta}$;
- Odo: odometry;
- PM pose: the matched pose of the PM’s algorithm;
- PM loss: absolute mean Euclidean distance to the nearest wall of each LiDAR beam;
- Timestamp: the current timestamp of the control period;
- Localization: it includes all the components of localization, including its metadata (EKF’s mean and covariance, loss of the localization, and error of the PM and RoPM).

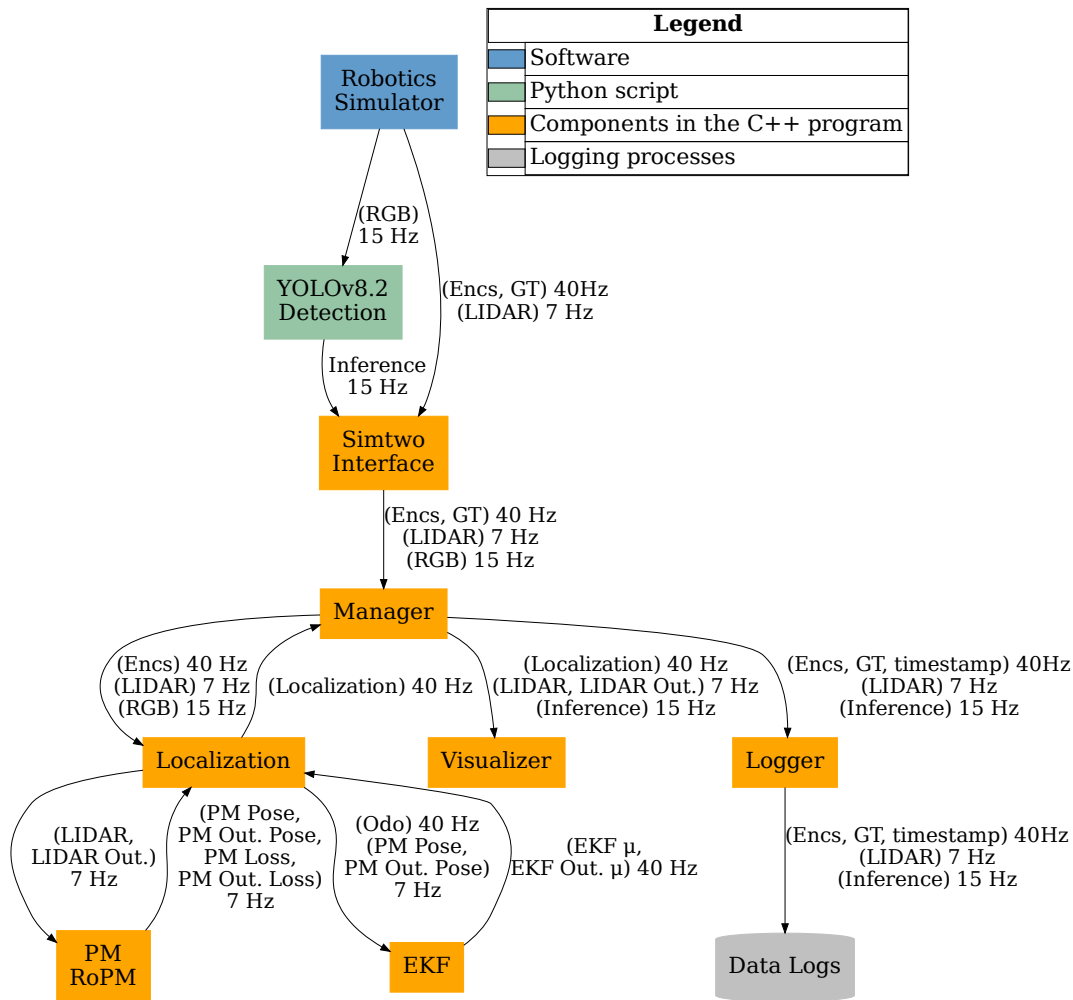


Figure 3.22: System architecture's DFD in "Online" mode.

As it is possible to see in Figure 3.22, the C++ application's components are represented by the orange nodes, the external entity by the blue node, the Python script by the light green node, and the data storage by the grey cylinder. The SimTwo simulator, represented by the `Robotics Simulator` node, delivers RGB data at 15 Hz to the Python script that runs YOLOV8.2 with the default model for inference, which in turn delivers via `UDP` protocol the inferences data to the `Simtwo Interface` component. In addition, the simulator also sends encoders and ground truth pose data at 40 Hz of the robot to the `Simtwo Interface` component. Also, `LiDAR` data is updated and sent at 7 Hz. After that, the `Manager` component centralizes the processes from the other components by managing the data access through the other components. The `Localization` node processes the robot's localization, including the robot's odometry, `EKF`, `PM`, and the outliers rejections. After all of this is performed, updated data is sent to the `Visualizer` node that generates the `Graphical User Interface (GUI)` of the application. Finally, all necessary data is sent to the `Logger` component that logs the data for posterior

analysis in “Offline” mode. Other components not crucial for this thesis were omitted, such as the `AMRController`. The application runs and manually manages three threads, one for the `SimTwoInterface`, another for the `Manager`, and the other for the `Visualizer` nodes, ensuring thread-safe operations and efficient execution. The application uses other libraries that can manage other threads, such as `spdlog`³, a C++ thread-safe and efficient asynchronous library. In this case, this library was implemented as a singleton class, and all components have a reference to it where the logs are directed to prompt and file sinks. Figure 3.23 concisely represents the application’s concurrency and synchronization strategies.

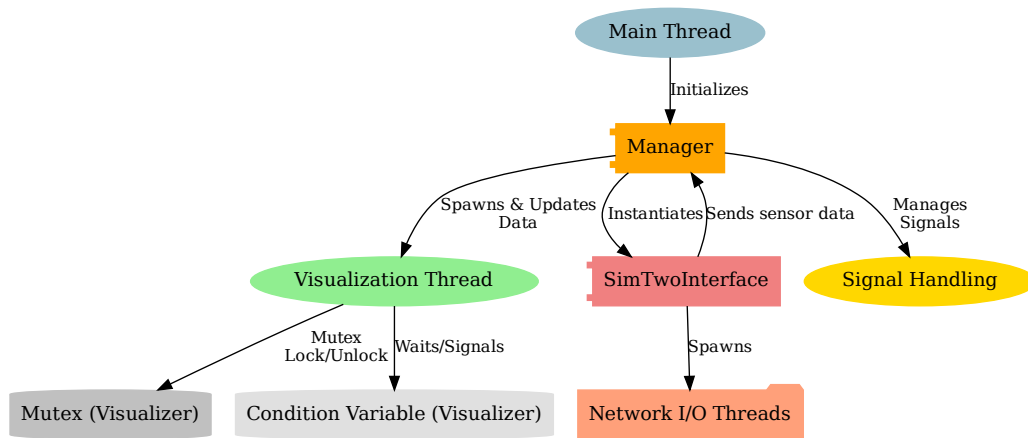


Figure 3.23: System architecture - threads interaction diagram

Figure 3.23 visualizes the application’s architectural interactions and threading model. It depicts the `Main Thread` initiating the `Manager`, the central component that orchestrates the several operations within the application. The `Manager` is responsible for starting and managing the `Visualization Thread`, handling signals for graceful shutdowns through `Signal Handling`, and processing data from the `SimTwoInterface`. The `Visualization Thread` operates concurrently, using synchronization mechanisms such as a `Mutex` and a `Condition Variable` to safely interact with shared data, ensuring thread safety and data consistency. The `Network I/O Threads` represent external communications handled by `SimTwoInterface`. Another way to understand the code logic of the system in “Online” mode is by analyzing Algorithms 18 and 19.

Algorithm 18 attempts to parse the received data to extract sensor readings. If successful and readings are available, it proceeds to further data processing, updating the state of the localization module and potentially the visualization. If data is absent or corrupted, it logs it.

Algorithm 19 waits for a readiness signal to ensure the data is ready for rendering. Once confirmed, it engages in a continuous rendering loop (until instructed to close), during which it

³<https://github.com/gabime/spdlog>

Algorithm 18 Data Reception and Processing

```

1: Input: data_string
2: procedure PROCESSRECEIVEDATA
3:   try
4:     encoders, ground_truth_pose, laser_readings  $\leftarrow$  ParseSensorData(data_string) ▷ Parser method
5:     if data is available then
6:       Process data ▷ Other robot's modules will use
7:       Update visualization
8:     else
9:       Log info about missing data
10:    end if
11:  catch Exception
12:    Log error details
13:  end try
14: end procedure

```

Algorithm 19 Visualization Rendering

```

Require: Window context
1: Input: runRenderLoop ▷ Flag that monitors if the rendering should execute
2: Wait for readiness signal
3: glfwMakeContextCurrent(Window) ▷ Activate context
4: while runRenderLoop and not glfwWindowShouldClose(Window) do ▷ Checks if it is to render and the window is open
5:   handleEvents() ▷ it blocks until new data comes from main thread
6:   swapBuffers() ▷ triple-buffer approach to avoid data races
7:   drawUIElements() ▷ Render user interface elements
8:   drawCamVis() ▷ Render camera visualization
9:   finishRender() ▷ Finalize rendering the texture and poll for window events
10:  updateDataAvailability() ▷ Set new_data conditional variable to false
11: end while

```

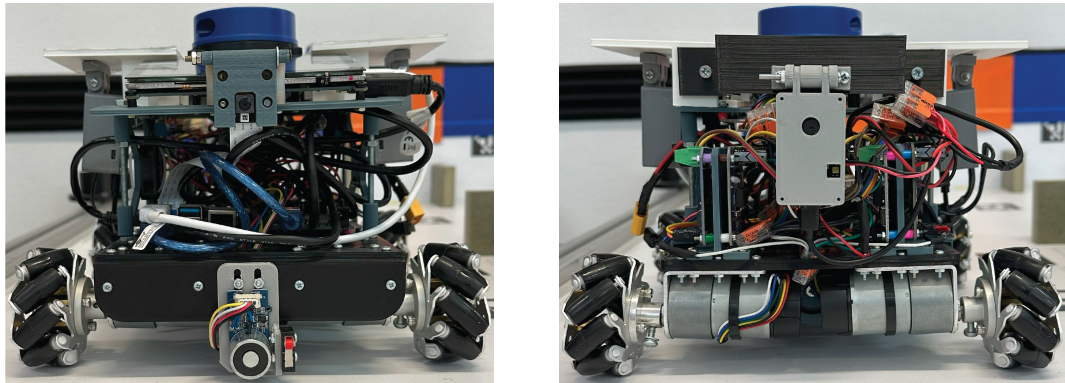
handles user interactions, updates the display buffers, and renders the **User Interface (UI)** and camera visuals based on the latest available data.

The system includes an “Offline” mode (Figure 3.25) for processing data logs from the robotics simulator or robotic platform. In this mode, the platform uses inference data from four cameras instead of just the frontal camera (Figure 3.21). It was decided to use ESP32-CAMs on the robot to compromise cost-effectiveness and ease of use. While an omnidirectional camera was considered, the ESP32-CAMs offer several advantages. They allow the hypothesis’ validation without complex image distortion processing, provide a more modular and cost-effective solution, and support a decentralized system that can be integrated into an embedded system. Figure 3.24 shows four straight views from the robot’s cameras.

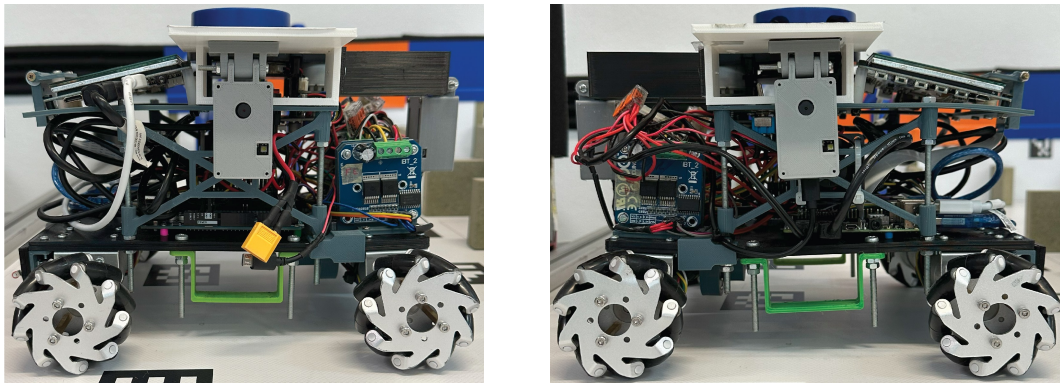
The logging system of the robotic platform follows the same architecture as this system’s architecture when it runs in “Online” mode with the robotics simulator, ensuring system compatibility between both.

The “Offline” mode uses another component named “OfflineAnalysis”, which decodes the raw data logs for posterior analysis, considering all the data different sources’ frequencies and synchronisms, mimicking the system as if it were running in “Online” mode. This approach increases the system’s flexibility for tuning and analysis. In the same reasoning as the “Online” mode, the “Offline” mode can also be understood as verifying key methods from the code logic in Algorithms 20, 21, and 22.

The `runOfflineAnalysis` method checks the existence of the log file and logs an error if the file is not found. If the file exists, it sets up loggers specific for offline analysis and then calls



(a) Frontal view of the real robot and the Rpi Cam module V2 (b) Back view of the real robot and the ESP32-CAM module



(c) Left view of the real robot and the ESP32-CAM module (d) Right view of the real robot and the ESP32-CAM module

Figure 3.24: Four straight views of the real robot with the camera modules

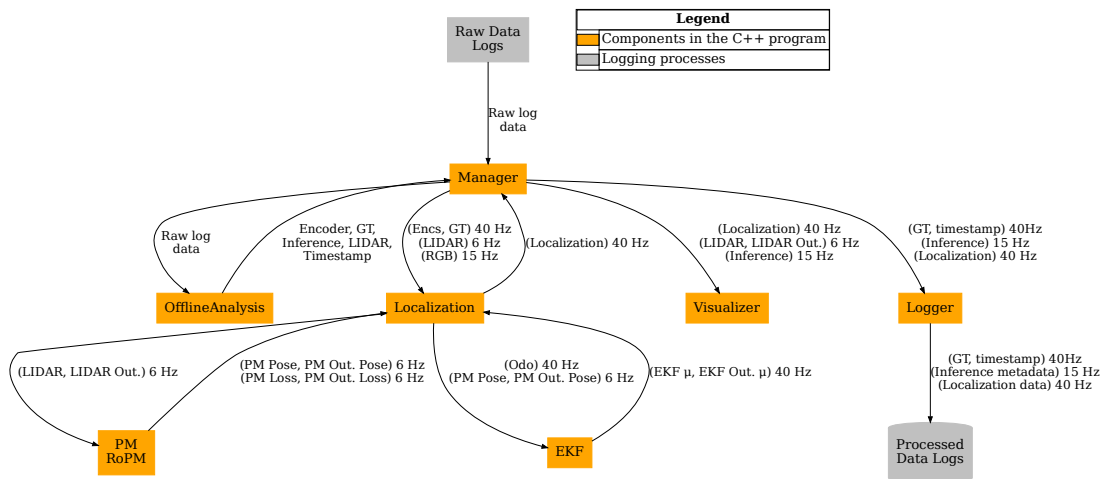


Figure 3.25: System architecture DFD in “Offline” mode. Raw data logs could be from the robotics simulator or the real robotic platform.

another method to process the log file.

Algorithm 20 runOfflineAnalysis(logFilePath)

```

1: Input: logFilePath ▷ File path of the log file
2: procedure RUNOFFLINEANALYSIS
3:   Log start of the analysis process
4:   if logFile does not exist at logFilePath then
5:     Log error message return
6:   end if
7:   processLogFile(logFilePath) ▷ Algorithm 21
8: end procedure

```

Algorithm 21 processLogFile(logFilePath)

```

1: Input: logFilePath ▷ File path of the log file
2: procedure PROCESSLOGFILE
3:   Open log file at logFilePath ▷ Log existence already checked in Algorithm 20
4:   while not end of file do
5:     Read next line from file
6:     parseLine(line) ▷ Algorithm 22
7:   end while
8:   Close log file
9: end procedure

```

The processLogFile method opens the specified log file and processes it line by line using the parseLine method. It continues reading and processing until all lines are processed, marking the end of the data log process.

Algorithm 22 parseLine(line)

```

1: Input: line ▷ String
2: procedure PARSELINE
3:   try:
4:     Extract components from line ▷ Components are encoders, ground truth, laser readings, object detection, and timestamp data
5:     Process data without semantic interpretation
6:     Process data with semantic interpretation
7:   catch exception
8:     Log it and return the error code
9: end procedure

```

The parseLine method attempts to extract several pieces of data from a single log file line. If successful and LiDAR data is available, it processes this data with standard algorithms and semantic interpretation. If the data extraction fails, it logs the failure.

Payload communication approach

The sensor payload from the robotics simulator is larger than the MTU to send over the network layer. The robotics simulator and the C++ application run on different machines (the latter runs on Windows Subsystem for Linux 2 (WSL2)); thus, an approach to split the payload into smaller packets was performed to avoid datagram fragmentation. Therefore, on the side of the simulator, Algorithm 23 describes the pseudocode to split and encode the data.

The encoder algorithm, “Split and Send UDP Data”, splits a large data block into manageable UDP packets. It formats each packet with a unique header, including sequencing information, computes a checksum for error detection, and sends the packet to a specified IP and port. The sequence ensures that the data can be reconstructed in the correct order on the receiving end.

Algorithm 23 Sensor payload encoder - Robotics Simulator

```

1: Constants: MaxPayloadSize                                ▷ 1472 bytes, MTU limit minus IPv4 and UDP header sizes
2: Input: IP, Port, Data
3: procedure SPLITANDSENDUDPDATA
4:   Convert Data to string format
5:   Calculate total data length and HeaderSize                ▷ HeaderSize also takes into account the checksum size
6:   Determine effective payload size                          ▷  $EffectiveMaxPayloadSize \leftarrow MaxPayloadSize - HeaderSize$ 
7:   Determine the number of packets based on effective payload size
8:   for each packet do
9:     Generate packet header
10:    Extract data segment for the current packet
11:    Calculate checksum for packet data                       ▷ XOR each byte in PacketData
12:    Prepend checksum to packet data
13:    Send UDP packet with checksummed data
14:    Update current position in data
15:   end for
16: end procedure

```

Conversely, the decoder algorithm is displayed in Algorithm 24. The “MaxPayloadSize” is 1472 bytes, 1500 of the MTU limit subtracting the IPv4 and UDP header sizes.

Algorithm 24 Handle Received UDP Data - Localization application

```

1: Inputs: error, bytes_transferred
2: procedure HANDLERECEIVE
3:   if no error then
4:     Extract received data from the buffer
5:     if received data length < minimum required length then
6:       return                                                ▷ Discard message
7:     end if
8:     Extract received checksum from the first byte
9:     Compute checksum for remaining packet data
10:    if received checksum  $\neq$  computed checksum then
11:      return                                                ▷ Discard packet
12:    end if
13:    Extract packet ID and total packets count from header
14:    if packetID == 1 and packetBuffer not empty then
15:      Clear packetBuffer                                    ▷ New transmission started
16:    end if
17:    if packetID > totalPackets or packetBuffer.size()  $\geq$  totalPackets then
18:      Clear packetBuffer
19:      return                                                ▷ Discard invalid packet
20:    end if
21:    Store packet data in packetBuffer[packetID]
22:    if packetBuffer.size() == totalPackets then
23:      if all packets received in correct order then
24:        Reconstruct combined data from packetBuffer
25:        Process combined data (via callback if set)
26:      end if
27:      Clear packetBuffer                                    ▷ Clear after processing or if incomplete
28:    end if
29:  else
30:    Log error message
31:  end if
32: end procedure

```

The decoder algorithm, “Handle Received UDP Data”, processes incoming UDP packets. It verifies the integrity of each packet by comparing the received checksum with a computed one. It extracts packet sequencing information from the header and checks for the expected order. The original message will be reconstructed if all packets are received correctly and in order. The message is discarded if discrepancies are found (such as missing packets, packets arriving out of

order, or checksum mismatches).

With all this in mind, Figure 3.26 shows the outline of the GUI interfaces of the application.

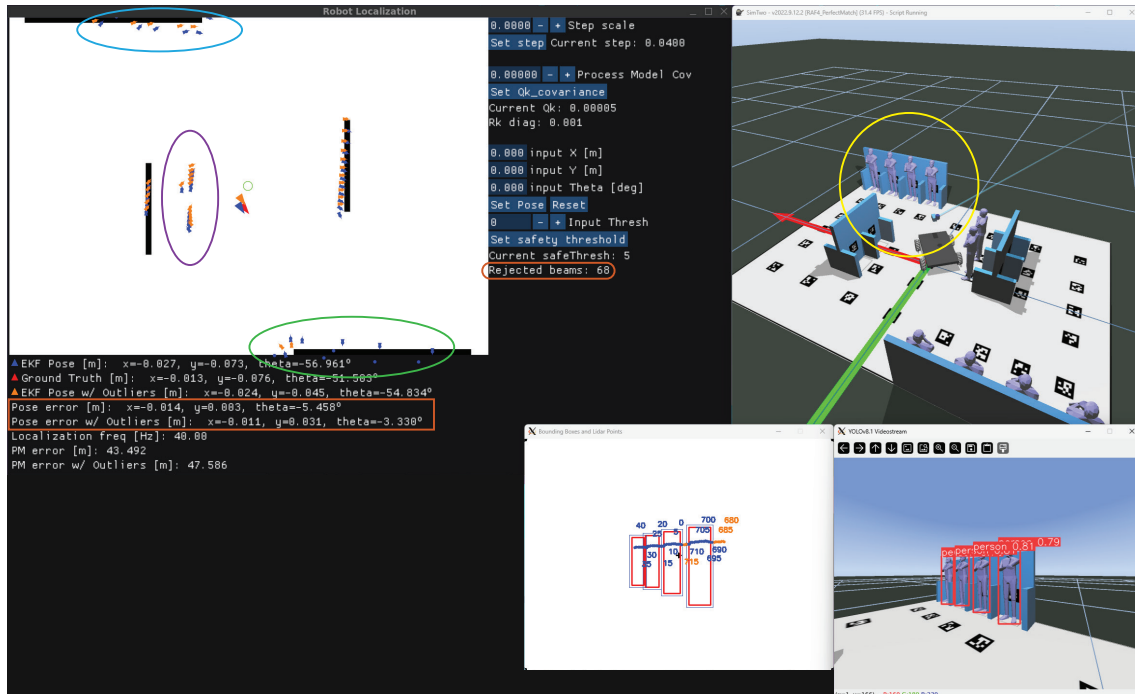


Figure 3.26: GUI interface of the AMR application. The main window (top-left) shows the robot map and poses. Blue, green, and purple ellipses highlight incorrect LiDAR point minimization. The green ellipse also shows rejected points. Orange delineations indicate beam rejections and localization errors. The yellow circle corresponds to the robot's view in video streams. Simulator world (top-right), camera feed (bottom-right), and LiDAR projections (mid-bottom) provide additional system information.

As it is possible to see in Figure 3.26, the upper-left window is the main application window where it shows the current map of the robot graphically as well as pertinent logging data. In the map, the bigger triangles represent the pose of each system, where the red represents the ground truth pose, the orange represents the RoPM's localization system, and the blue represents the original system. Some points (dots) are painted on the map; these represent LiDAR points projection according to each localization system of the PM; they are also color-coded with the same colors as their pose counterparts. The software does not paint all the LiDAR points to avoid too much cluttering in the GUI, so it is possible to configure the decimation factor. These points also have tiny triangles displaying the gradient direction for each LiDAR point to show the optimization algorithm's direction for a given control cycle. The main window GUI also offers options to change the localization systems' parameters for debugging. The window at the top-right part of the illustration displays the simulation world of the Simtwo simulator with the factory mockup, the personnel, and the AMR. The bottom-most-right figure shows the software displaying the AMR's camera transmission with the YOLO's inference information. Finally, the mid-bottom window shows the LiDAR and bounding boxes information from the perspective of

the **AMR**'s camera. In this window, the orange points represent the projection of the **LiDAR** points in the camera's image plane. The rectangles show the bounding boxes of the outliers, where the red rectangles represent the actual dimensions of the bounding boxes and the blue ones are their dimensions, plus a safety threshold that can be finely adjusted to compensate for errors in camera calibration and approximations. This adjustment presents a trade-off: a stricter threshold ensures better rejection of outlier **LiDAR** points but may discard valid point projections. In contrast, a more lenient threshold preserves more useful projection data but risks retaining some erroneous points in the minimization process. A good calibration should avoid the need to use the safety threshold. Our methodology explicitly avoids injecting wrong measurements into the **EKF** to ensure more stability in the localization process. Finally, the blue points are the **LiDAR** projections that fall into the bounding boxes and are rejected. The numbers' labels on this window show the index of every n -th **LiDAR** point for debugging purposes.

In addition to this explanation, there are some colored ellipses, rectangles, and circles drawn on top of the figure. This was done to help describe the software since a lot of information is embedded. The blue ellipse at the top-left of the figure emphasizes some **LiDAR** points. These points are on the back of the robot, which in the simulation has only one camera, and they are "falling" or being "minimized" to the wrong positions because they are hitting the personnel at the garages. This can be confirmed in the world's scene in the simulator window (the personnel away from the yellow circle). Also, the purple ellipse is in the back-right of the robot, which shows much clearer the **LiDAR** points being minimized to the wrong positions (for the **PM** to match the maps flawlessly, the **LiDAR** points should hit the walls).

The green ellipse highlights rejected **LiDAR** points (those without triangles). The other points were not rejected since they were not projected inside the bounding boxes in the given frame. The potential misalignment of the bounding box or personnel's **LiDAR** points in the image plane may also affect the rejections. The **GUI** displays every n th **LiDAR** point, so only a few rejection examples are visible. The orange rounded rectangle on the right emphasizes the number of rejected beams during this screenshot period, corroborated by blue points in the camera's perspective window. The straight orange rectangle compares localization errors: the **AI**-enhanced system (`Pose error w/ Outliers`) shows fewer errors than the non-**AI** system (`Pose error[m]`). Lastly, the yellow circle indicates the robot's current view, aligning with the two video stream windows: one with inference information and another showing the robot's perspective with **LiDAR** points and outliers.

Figure 3.27 shows the software application running without personnel to show the localization algorithm running as it should when no outliers exist.

In this figure, it is possible to see both localization systems having the same error metrics with practically zero error in the pose states, as expected. In addition, the pose triangles of the ground truth and the two systems are overlapped, as with the **LiDAR** projections, showing that both approaches converged to the same value. Finally, Figure 3.28 displays the synchronicity of the camera of the simulator and the application.

As shown in Figure 3.28, red boxes represent the dimensions of the outliers. Blue ones have

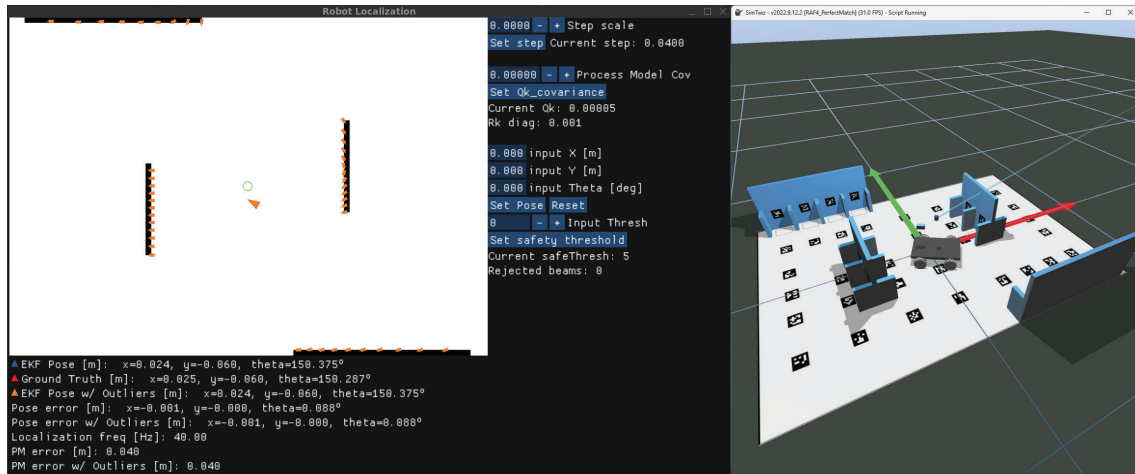
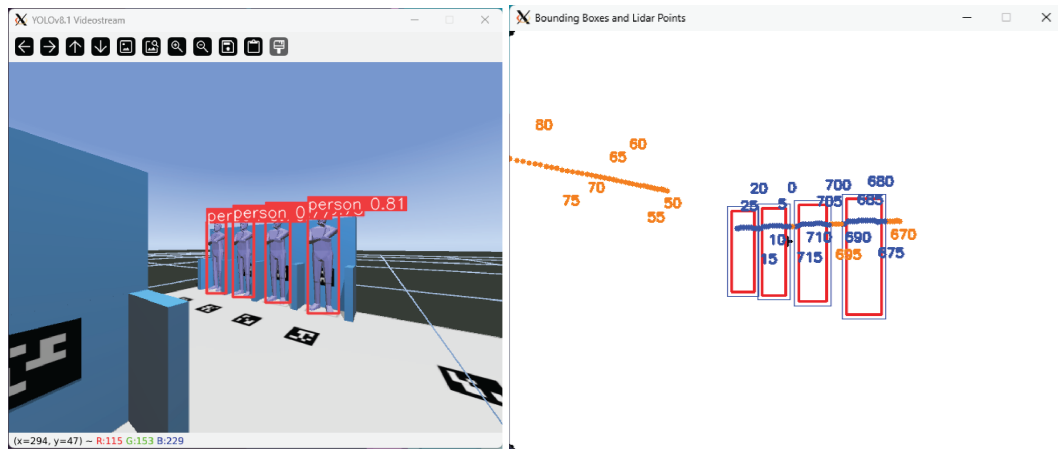


Figure 3.27: Localization application running in “Online” mode without personnel



(a) Application's image stream post-semantic (b) Robot's LiDAR points and outliers in the camera inference process perspective

Figure 3.28: Outlier detection and rejection process

their dimensions added to a safety threshold. The allowed LiDAR points are orange, and the rejected ones are blue. The labels are the indices of the LiDAR points, shown every 5 points. The x axis of the LiDAR frame on the robot is aligned with its front, and it rotates Counter Clockwise (CCW). For this reason, it is possible to see the numbers increasing to the left (the LiDAR points are hitting the front and left wall in the image) and reaching approximately 715 near the image's center. This happens as the simulated LiDAR follows a model of the real LiDAR used in the robotic platform, YDLIDAR X4, which was configured to have 0.5° of resolution for 360° of FoV.

3.3.2 Perfect Match

This section will discuss the PM, as it was described with details in [86, 88]. In essence, the PM algorithm was first developed by Martin Lauer in [86] for the Robocup Midsize League, and it was designed for the omnidirectional cameras of the robots. This algorithm works by minimizing

the error, or maximizing the match, between a given current scan of the map from the robot's perspective and the ground truth map of its workspace. In the case of Martin Lauer's work, it was the field markings of the RoboCup field. Later, researchers in [88] developed an adaptation of the algorithm to support inputs of 2D Laser Scanners instead of camera images. In the case of this research work, the developed version of the **PM** supports 2D **LiDAR** data, and the map chosen for the case study is the scaled **I4.0** factory mockup for autonomous robots, where the walls of the environment are features used to match the maps. Besides the input, the other difference from the original **PM** is that this work's minimizer is the gradient descent, not the original resilient backpropagation. Thus, the following steps were taken to implement this variant of the **PM**:

Step 1. Pre-compute the distance map of the environment using an M-estimator, the same estimator as in [86]. The computation used the morphological transformation called `Exact Euclidean distance transform`⁴. Figures 3.29a and 3.29b show the map of the factory floor and the transformed distance map, respectively.

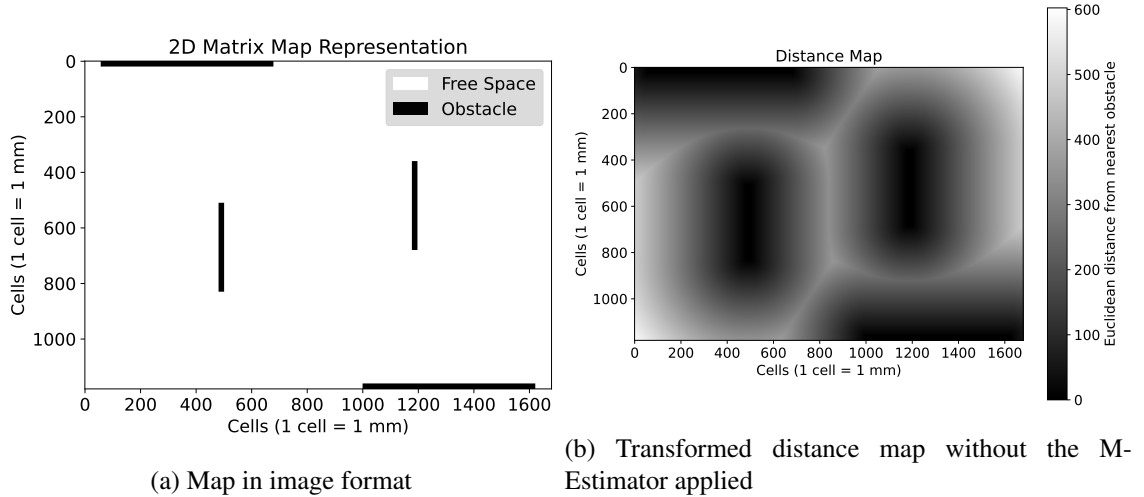


Figure 3.29: Map (a) and distance map (b) of the environment

Given a measurement error d , the function for the M-estimator that filters outliers can be defined as:

$$d_{\text{err}}(d) = 1 - \frac{c_{\text{err}}^2}{c_{\text{err}}^2 + d^2} \quad (3.5)$$

where c_{err} is a predefined threshold constant that helps in determining the impact of an outlier based on its deviation (d) from the expected value. The function operates by adjusting the weight assigned to each data point based on its error d . For points where d is small, indicating they are close to their correct locations, the function returns a value close to 0, thereby minimizing their contribution to the error term. Conversely, for points with large deviations d , indicative of potential outliers, the function yields values approaching 1, effectively saturating their influence in the error computation. In our context, this means that **LiDAR** points far from the walls contribute

⁴https://docs.scipy.org/doc/scipy/reference/generated/scipy.ndimage.distance_transform_edt.html

significantly to the error term, helping identify and handle outliers. Figure 3.30 shows the curve of the M-estimator with the c_{err} of 70, where it was defined via a sensitivity analysis.

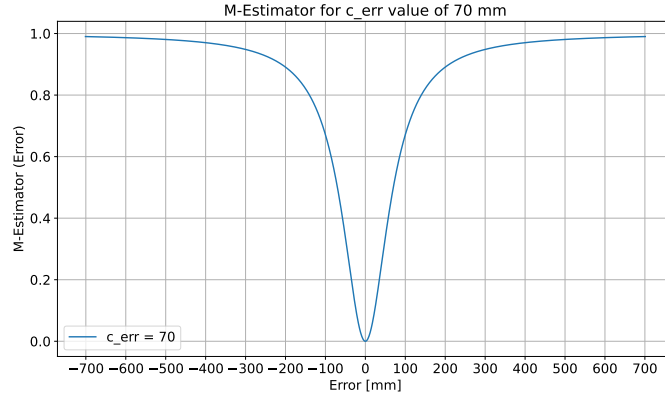


Figure 3.30: Curve of the M-estimator with c_{err} of 70

Step 2. Pre-compute the gradient maps in x and y directions to ensure fast convergence with gradient descent. The implementation of the gradients used the 1-th order accurate differences at the boundaries⁵. Figure 3.31 shows the gradient in x and y axes before and after applying the M-Estimator with c_{err} of 70.

Step 3. Propagate the robot's pose in the world's frame via the process model. It used an **EKF** with the robot's kinematics as the process model.

Step 4. Process the current laser scanner data in the **LiDAR**'s frame, transform it into the robot's frame, and project it into the world's frame.

Step 5. Minimize the distance of each valid laser beam's position to the nearest wall from the original map by adjusting the robot's pose. This distance is the error of where each laser beam projection was propagated and where it should have been if the robot's pose had been correctly propagated.

Step 6. Stop criteria is running several iterations (steps 4 to 6) or a limited time (must finish before the next control cycle).

Step 7. Repeat steps 3 to 6 every time a new control cycle happens.

The chosen **PM** loss for the algorithm was the mean absolute distance of all **LiDAR** measurements to the closest wall. The conditions of the stop criteria were chosen as the ones in step six and not reaching the desired level of the loss function for two reasons:

1. Real-time execution constraint. The robot cannot face a situation where it needs to wait for the **PM** to reach a desired **PM** loss threshold. Instead, it is of the utmost importance that the **PM** stops optimizing until a new **LiDAR** scan arrives, i.e., in approximately 166 ms for a 6 Hz **LiDAR** operation frequency and ideally before the next control cycle, 25 ms.

⁵<https://numpy.org/doc/stable/reference/generated/numpy.gradient.html>

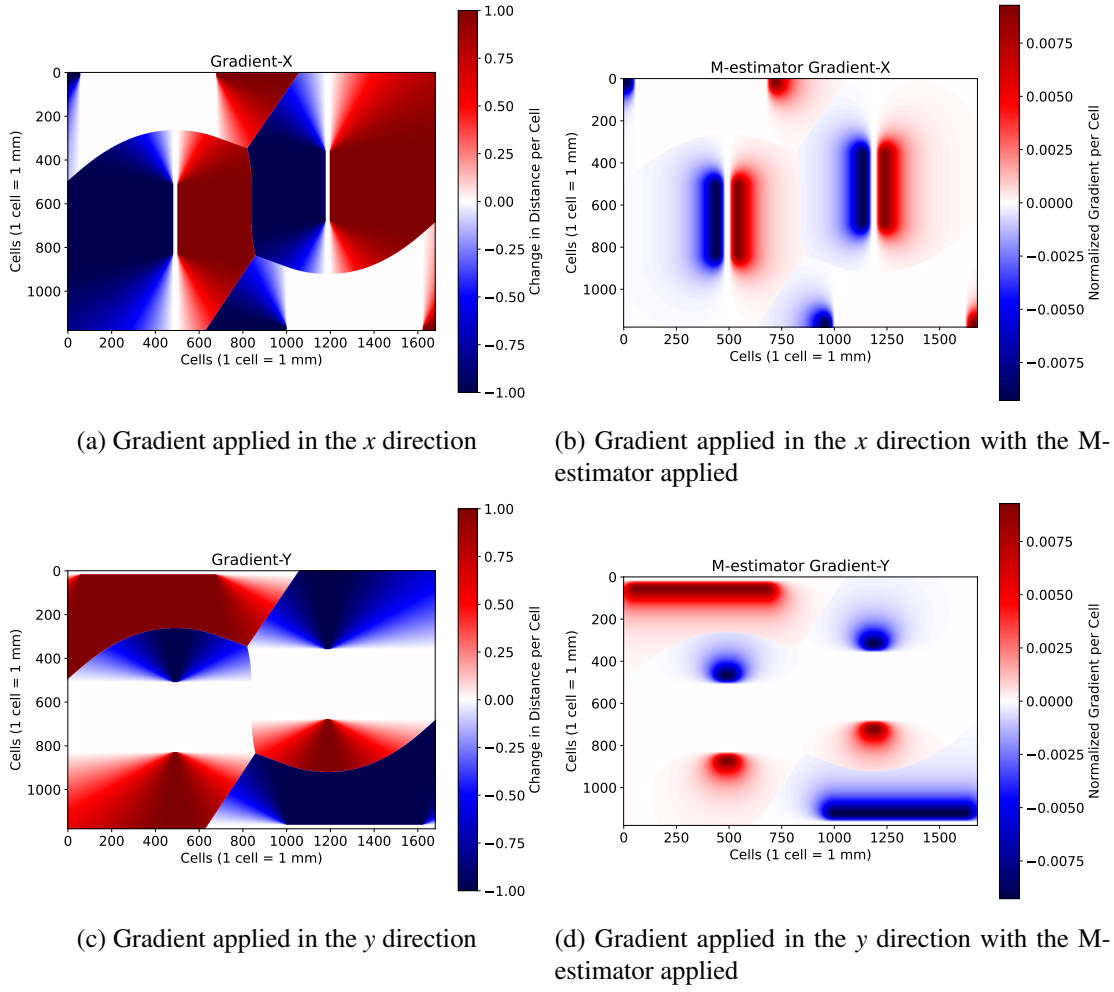


Figure 3.31: Gradients applied in x and y direction of the distance map before and after applying the M-estimator

2. As the AI-enhanced indoor localization system will reject LiDAR measurements, depending on the approach of rejecting them, the PM loss will be affected, and the meaning of its magnitude will change.

Therefore, Algorithm 25 shows a high-level representation of the PM's implementation.

3.3.3 Localization System

The localization system comprises an EKF where the robot's odometry is the process model, described in Section 3.1.5. The update stage is the PM's measurement after the stop criteria are met (step 6 of the PM in Subsection 3.3.2). The process model runs at 40 Hz, i.e., the encoders are measured at 40 Hz frequency, and the robot's odometry and the predicted stage of the EKF are processed. Asynchronously, the update stage of the EKF runs at 7 Hz, the sampling frequency of the LiDAR. Figure 3.32 displays a DFD of the EKF, where $enc_{1..4}$ are the robot's encoders measurements, $\hat{x}_{k|k-1}$ the predicted *a priori* state estimate, $P_{k|k-1}$ the predicted *a priori* estimate

Algorithm 25 Perfect Match Algorithm

```

1: Input:
2: PM                                ▷ Reference to Perfect Match object
3: lidarData                          ▷ container of LIDAR scan
4: max_iter                            ▷ maximum number of iterations
5: max_duration                        ▷ maximum duration
6: procedure MATCH
7:   timeout_time  $\leftarrow$  current time + max_duration
8:   result  $\leftarrow$  new Pose()
9:   PM.setPose(EKF.getPose())
10:  for iter  $\leftarrow$  0 to max_iter - 1 do
11:    result  $\leftarrow$  ITERLASER(PM, lidarData)
12:    if current time  $\geq$  timeout_time then
13:      return result
14:    end if
15:  end for
16:  return result
17: end procedure
18:
19: Input: PM, LaserPoints
20: procedure ITERLASER
21:   Initialize dx, dy, dtheta, pmLoss to 0
22:   for each point in LaserPoints do
23:     if point is valid then
24:       Calibrate to robot frame and transform point to world coordinates
25:       Calculate pixel coordinates u, v                                ▷ From the pre-computed map
26:       if u and v are within map bounds then
27:         Update dx, dy, dtheta based on gradients of the maps          ▷ Accumulate every point contribution
28:         Update pmLoss                                                  ▷ Accumulate the error for every point
29:       end if
30:     end if
31:   end for
32:   Adjust RobotPose based on the accumulated dx, dy, dtheta
33:   Calculate mean absolute pmLoss
34: end procedure

```

covariance, z_k is the observation of the true state, $\hat{x}_{k|k}$ is the updated *a posteriori* state estimate, $P_{k|k}$ is the updated *a posteriori* estimate covariance, and $\hat{x}_e$ the robots' states estimates. The process and observation noise covariances, Q_k and R_k , respectively, were determined through a sensitive analysis aiming for a balanced convergence speed and robustness against measurement noise.

Q_k and R_k have essential roles in ensuring the accuracy and reliability of the state estimation in EKF-based systems. Ideally, these covariances should be modeled systematically based on the underlying physical processes and measurement principles. The process model relies on the robot's odometry, and the observation model uses the PM. However, accurately modeling the covariance matrices for both process and observation noises is filled with challenges:

1. The process noise in odometry can be highly variable, influenced by factors such as wheel slippage, dead zones in motor operations, and variations in battery voltage affecting motor torque and controller outputs. These factors introduce non-linearities and uncertainties that are difficult to capture in a static or variable-dependent model.
2. The observation noise of the PM algorithm is challenging to model due to multiple factors:
 - Distance-dependent noise characteristics of each LiDAR measurement, already accounted for in LiDAR's noise model described in Section 4.1.

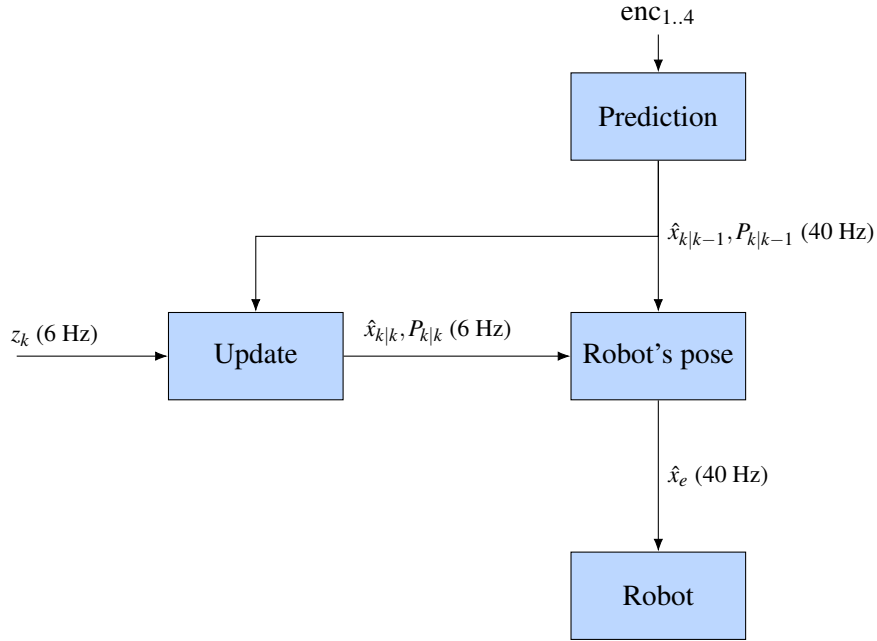


Figure 3.32: DFD for EKF-based localization using PM/RoPM as absolute reference sensors in the AMR

- Quantization errors and error distribution caused by the LiDAR scanner's resolution and accuracy.
- Dependency on the map's resolution and potential computational rounding errors of the PM algorithm affecting the pose estimation's quality.
- Variation in the 2D pose of the LiDAR alters its perspective and thus the perceived environment.
- Environmental variables, such as lighting conditions and dynamic objects (e.g., people moving through the sensor's coverage area), introduce random error components to the error distribution.

Given these complexities, a purely systematic modeling approach was considered impractical for this research. Such an approach would require extensive discretization and testing across different scenarios on the factory floor, which would still not guarantee an accurate representation of the noise characteristics under all operational conditions. This hypothetical heuristic-based covariance model would depend on the robot's pose to determine the current covariance. This could destabilize the filter if the estimate diverges, causing unstable feedback and making it diverge completely.

Instead, a sensitive analysis was performed, focusing on the adaptive tuning of R_k given a fixed Q_k to achieve adequate filter performance since the relation of these two dictates the convergence speed of the filter and its sensitivity to noise. The approach focused on adjusting the former to achieve stability of the filter and a practical speed convergence to the observation that enabled the robot to navigate smoothly at a given nominal speed, ensuring that the localization's estimate

would not lag behind the navigation. In addition, the application in “Online” mode was designed with a flexible adjustment mechanism in the GUI, allowing for online tuning of the R_k matrix by changing the values of the principal diagonal.

Extrinsic calibration of coordinate frames

The accurate operation of the localization system and consequently the AMR necessitates precise calibration between different coordinate frames used by several subsystems. Specifically, the ground truth system and the LiDAR sensor need their pose estimations aligned with the robot’s frame of reference. The LiDAR’s coordinate frame, which serves as the PM’s reference, primarily captures estimations directly linked to its perception field. In contrast, the ground truth system independently estimates the pose relative to its distinct reference frame.

Homogeneous transformations are applied to reconcile these diverse coordinate frames into the robot’s central frame. This process involves translating and rotating the pose estimates obtained from the LiDAR and the ground truth system into the robot’s frame. To visualize better the coordinate frames of these two systems, the reader is referred to the beginning of the chapter in Figure 3.4. In the illustration, it is possible to notice a pose offset between the LiDAR frame, the ground truth frame, and the robot’s centroid. Some offsets were caused by the assembly of the ground truth system itself since the designed support for the markers did not align correctly with the robot’s chassis. Thus, Figure 3.33 displays a diagram that illustrates the transformation relationships between the coordinate frames:

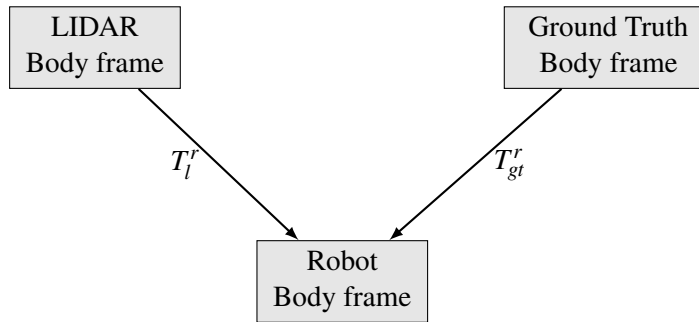


Figure 3.33: Sensor coordinate frames transformations for the AMR. T_l^r and T_{gt}^r represent the homogeneous transformations of the LiDAR frame and the ground truth systems relative to the robot’s frame, respectively.

This diagram visually represents the spatial relationships and transformations required to spatially register data from the LiDAR and ground truth systems into the robot’s body frame, facilitating accurate and reliable robotic actions.

3.3.4 Outlier Rejection Approach

The localization system incorporates a sensor fusion strategy using multiple cameras with LiDAR technology. This integration facilitates an outlier rejection algorithm explicitly designed to be minimalist and efficient, aligning with the constraints of embedded systems commonly deployed

in AMR within industrial settings. The key objective of the outlier rejection approach is to process LiDAR data by discarding any points that fall within the bounding boxes detected by the AI across four cameras. This is crucial to avoid the errors propagation into the pose minimizer of the localization system. In this reasoning, the procedural steps are detailed in Algorithm 26, which filters out irrelevant LiDAR points, enhancing the localization system's accuracy and performance.

Algorithm 26 Process Outliers for Multiple Cameras

```

1: Input: Points and Cams                                ▷ list of points and list of cameras, respectively
2: Output: Points, cnt
3: Initialize cnt  $\leftarrow$  0                                ▷ counter for invalid points
4: for each camera in Cams do
5:   Get bounding boxes for the current camera
6:   for each point in Points do
7:     Transform point to camera coordinates
8:     if point is behind camera then
9:       continue
10:    end if
11:    Project point to the image plane
12:    if bounding boxes exist then
13:      Check if the point is inside any bounding box
14:    end if
15:    if point is inside a bounding box then
16:      Increment cnt
17:      Invalidate point in Points
18:    end if
19:  end for
20: end for
21: return Points, cnt

```

Therefore, the outlier rejection process involves several key steps. Initially, a homogeneous transformation matrix is established for each camera, representing the extrinsic calibration between the relative poses of the LiDAR and each camera. This step allows the transformation of LiDAR points into camera frames. Subsequently, each LiDAR point is projected onto the cameras' image planes using the camera-specific projection matrices, following the pinhole camera model [207]. Finally, the points are rejected according to the algorithm's heuristic. This approach is designed to run at 40 Hz, allowing the system to maintain robust performance while meeting the requirements of industrial AMR applications.

Chapter 4

Experimental Setup and Validation Methodology

This chapter delineates the comprehensive experimental framework and validation methodology designed to substantiate the performance of the proposed localization system within both simulated and real-world environments. Initially, a detailed overview of the simulation environment and the specifics of the simulated [LiDAR](#) scanner model are presented. Subsequently, the chapter describes the real-world [I4.0](#) factory mockup, serving as the physical testbed. The focus then shifts towards the data collection processes and the validation criteria employed across these testbeds, ensuring a robust evaluation of the localization systems' effectiveness. Through systematic validation approaches and meticulous data analysis, this chapter sets the groundwork for the subsequent presentation of results, thus bridging theoretical development and practical application.

4.1 Simulation Environment

The simulation environment serves as the primary testbed for preliminary validations. This section describes the simulated [LiDAR](#) scanner model and other architectural elements of the simulation environment that are important for testing the localization system under controlled conditions. The simulation software used was the SimTwo software [208], a realistic rigid-body simulator that uses Open Dynamics Engine as the physics engine. The simulation scenario chosen as the case study is an [I4.0](#) factory mockup in scale. This scenario is also for a national Portuguese robotics competition, [RAF](#) [117]. Figure 4.1a shows the actual scenario, and Figure 4.1b shows the simulated scenario.

As shown in Figures 4.1a and 4.1b, it is possible to see several ArUco markers on the floor and the walls. The bigger walls on the bottom and the top of the image are the entrance and the exit of the warehouse for materials, respectively. The middle structures are intermediate locations mimicking processes in a factory named machines. In Figure 4.1b, a person is positioned at the warehouse entrance. Besides that, there is an [AMR](#) with mecanum wheels, a camera (blue cuboid), and a 2D [LiDAR](#) scanner (blue cylinder) near the center of the image. This is the simulation model

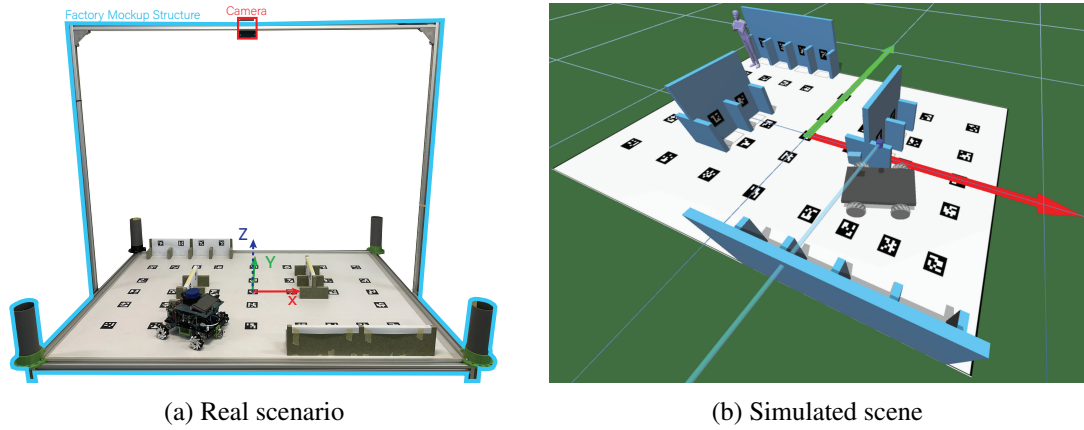


Figure 4.1: Simulated environment and the real scenario

(in the accurate scale of the physical dimensions) of the real **AMR** for testing in the actual scenario, represented in Figure 4.1a.

Noise Model Characterization of the YDLIDAR X4

An experimental framework was established to model the noise characteristics of the YDLIDAR X4 laser scanner used in the **AMR**. Measurements were conducted at ten evenly spaced intervals from 20 to 200 cm, with one hundred readings taken at each distance. This setup was intended to cover the entire operational range of the scanner in the controlled environment of the testbed. In this subsection, error and residual will be used interchangeably.

The distribution of measurement errors was initially analyzed to determine their conformity to a normal distribution. Standard normality tests were applied, including the Shapiro-Wilk, Anderson-Darling, Kolmogorov-Smirnov [209]; and Lilliefors tests¹.

Additionally, histograms and Q-Q plots were generated. However, these tests and plots did not support the assumption of normality. The primary reason identified was the quantization error inherent in the **LiDAR**'s measurement process, which limited error variability.

Despite the challenges in normality testing, further analysis of the mean and standard deviation of the errors was conducted. The mean errors across different distances consistently approximated zero, suggesting no systematic measurement bias. This behavior is shown in Figure 4.2a. Conversely, the standard deviation of the errors exhibited a discernible increase with distance, as shown in Figure 4.2b. This trend was quantitatively modeled using linear regression, indicating that measurement uncertainty grows as distance increases.

The standard deviation's relationship to distance was modeled with a linear equation:

$$\sigma(d) = \alpha d + \beta \quad (4.1)$$

where d represents the distance, and α (slope) and β (intercept) are coefficients derived from the data, as illustrated in Figure 4.3.

¹ <https://www.statsmodels.org/stable/index.html>

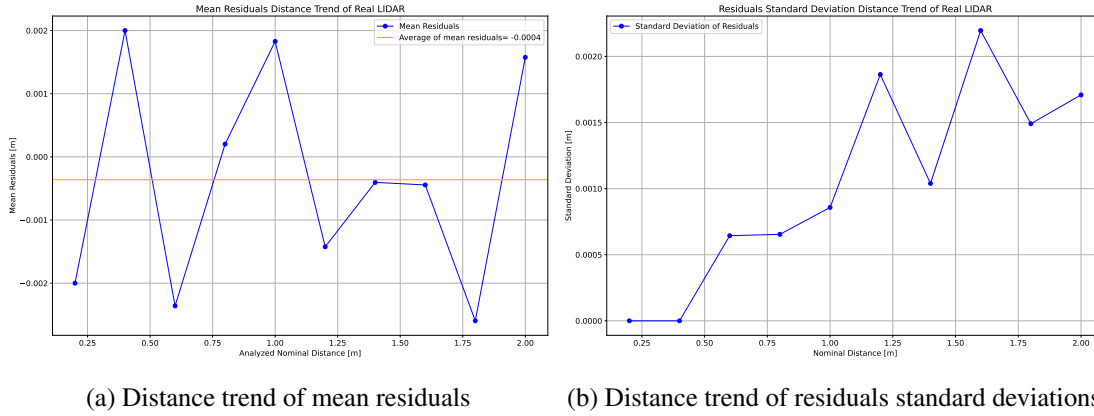


Figure 4.2: Distance trend of mean residuals and standard deviations - real LIDAR

Two linear regressors were tested, the ordinary least squares linear regression and the [Random Sample Consensus \(RANSAC\)](#) regressor using Scikit-learn [210]. [RANSAC](#) regressor was fit using two loss functions, `absolute_error` and `squared_error` with the rest of the other parameters being default. The chosen model was from the linear regression that coincided with the [RANSAC](#) with the squared loss because they had better R^2 scores. This result is expected since the `absolute_error` is less sensitive to outliers. However, the higher deviations with the distance are not outliers. Given all the circumstances and observations of the analysis, it was concluded that the noise can be reasonably assumed to follow a Gaussian distribution with a mean of zero and a standard deviation dependent on the distance.

The implementation of the noise model in the simulation environment is described in Algorithm 27, which uses a Gaussian distribution to add noise based on the calculated standard deviation:

Algorithm 27 LIDAR Noise Model

```

Input: laserBeam
Output: result
function LIDARNOISEMODEL
    double std
    double result = laserBeam
    if laserBeam ≥ 0 then
        std = slope × laserBeam + intercept           ▷ Slope and intercept comes from the linear model
        if std ≥ 0 then
            result = result + RandG(0, std)           ▷ RandG returns a random noise with mean zero and standard deviation std
        end if
    end if
    return result
end function

```

Algorithm 27 consistently applies the noise to each laser beam measurement with the distance-dependent noise characteristics defined by the linear model. The function `RandG` represents a Gaussian distribution generator with the specified mean and standard deviation. The characterized noise model for the YDLIDAR X4 provides a reasonable assumption for simulating the scanner's noise to avoid performing tests in simulation with ideal measurements. Despite the non-normal distribution of measurement errors due to quantization effects, the analysis confirmed the appro-

priateness of a linear model to represent the increase in error variability with distance. In other words, this approach enhances realism by increasing the challenges of the localization algorithms that depend on the reliability of simulations employing the YDLIDAR X4 in indoor navigation scenarios. This step eases the transition to the real-world scenario by providing non-idealities in the tests. The characteristics of the noise model are illustrated in Figure 4.3, showing the linear relationship between standard deviation and distance.

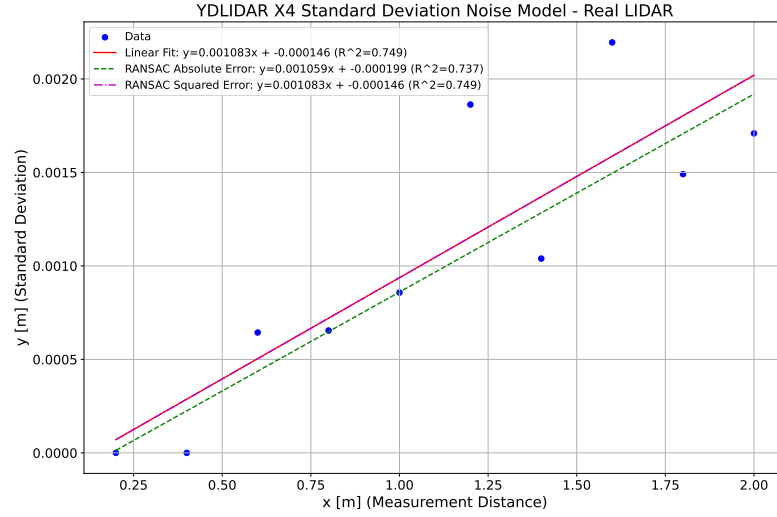


Figure 4.3: Linear model of the standard deviation of the YDLIDAR X4 scanner

After the implementation of the model, the same methodology of tests was applied in simulation following the developed noise model. Figures 4.4a and 4.4b show the distance trend of mean errors and standard deviations of the simulated model.

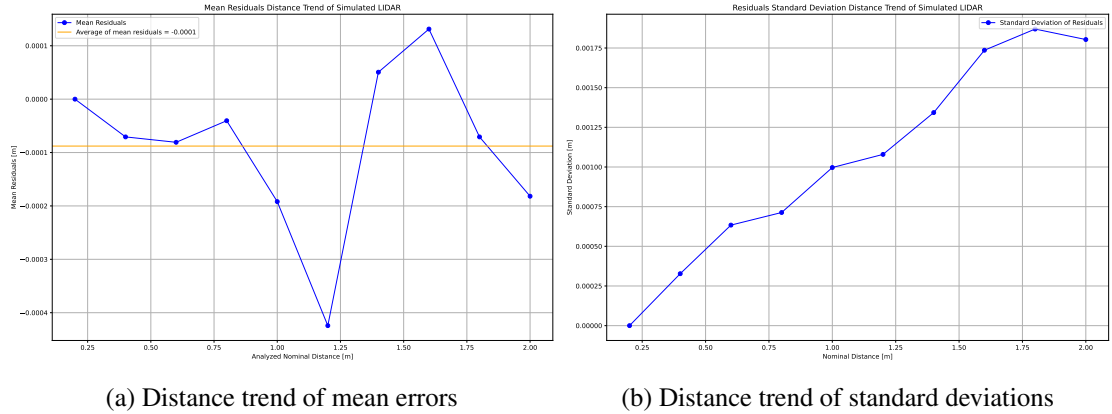


Figure 4.4: Distance trend of mean errors and standard deviations - Simulated LIDAR

As shown in Figure 4.4, the simulation results' trends match the real-world ones within the established working distance range used for testing the localization systems. It is expected that they are not going to be equal because some assumptions that are not true were made, namely assuming the noise model follows a normal distribution with mean zero and [Standard Deviation](#)

(STD) σ ($N(0, \sigma)$), where σ is the linear regressed model. Despite not being equivalent, the distance trend of the mean error still wanders around 0, and the bias of the mean line is less than 1 mm difference. The distance trend of standard deviations also grows at a similar rate.

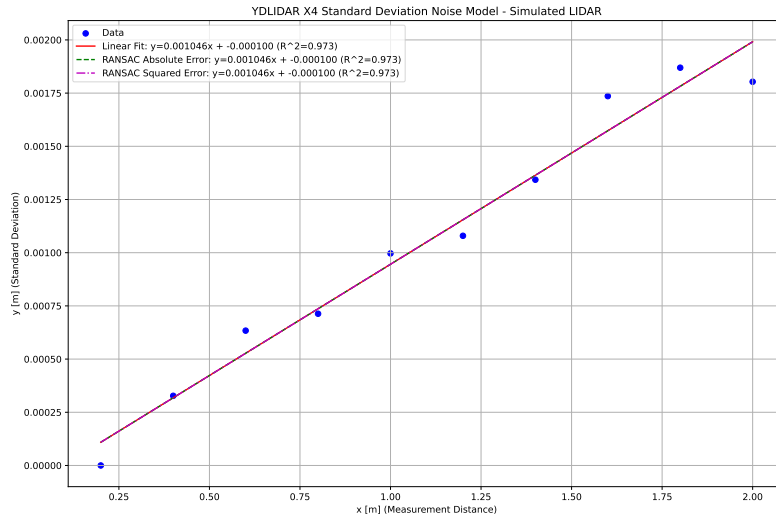


Figure 4.5: Linear model of the standard deviation of the simulated YDLIDAR X4 scanner

For comparison purposes, Figure 4.5 shows the model regressed from the simulated tests. The similarity between simulated and real results can be seen by comparing this model's slope and intercept with those from the real tests model shown in Figure 4.3.

4.2 Real-world Industry 4.0 Factory Mockup

This section provides a detailed description of the developed I4.0 factory mockup, emphasizing its role in validating the localization systems. The system was designed and assembled using as a case study a specific Portuguese Robotics Open competition, RAF². The competition exemplifies I4.0 principles by showcasing AMRs that perform tasks autonomously in an automated warehouse, reflecting the future of smart manufacturing. These robots navigate, recognize, and sort boxes using fiducial markers and communication protocols, demonstrating the essence of I4.0: digitalization, connectivity, and data-driven operations. This scenario highlights how robotic automation fosters efficiency, minimizes errors, and optimizes manufacturing and logistics processes. Moreover, the competition promotes innovation and digital skills development, which is crucial for I4.0's focus on technological advancement and workforce transformation. For more details, the reader is encouraged to read more about it in the link provided in the footnote and the research works [117, 204, 211]. In this reasoning, a top-view representation of the mockup is displayed in Figure 4.6.

In Figure 4.6, the factory floor comprises an entrance, denoted by “Warehouse in”, an exit, “Warehouse out”, and two machines, “Machine A”, and “Machine B”. Each blank rectangle inside

²<https://github.com/P33a/RobotAtFactory>

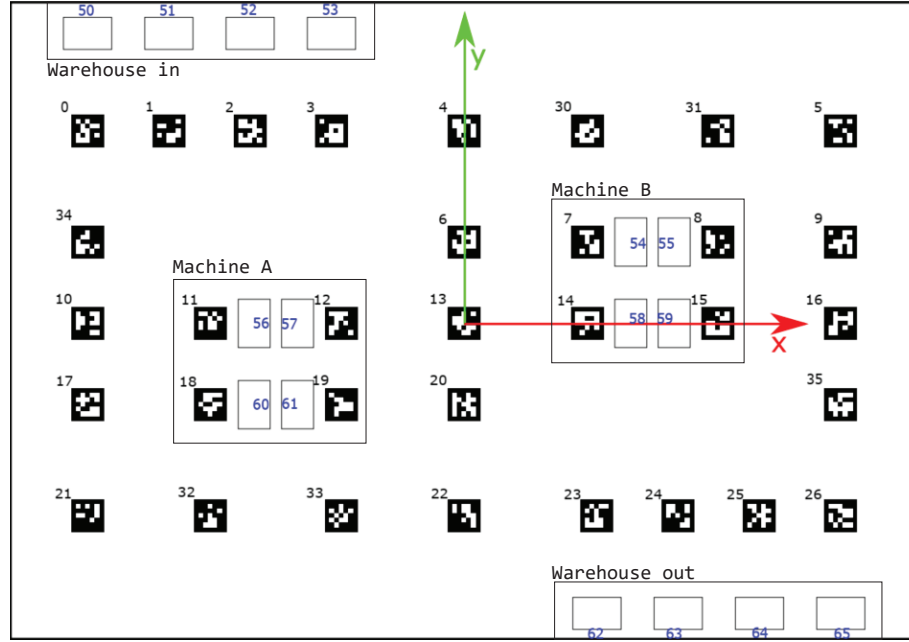


Figure 4.6: RAF factory floor mockup

these sections represents positions where boxes can be removed or taken in for subsequent operations, such as transport and processes. Each number represents an ID of an ArUco marker, where the world frame is aligned with the marker 13 pose. As this illustration is a top-view perspective, the blue numbers represent the fiducial markers' IDs placed on the factory scenario's walls.

The experimental setup comprises the RAF structured assembly in the laboratory, a physical replica of the official factory floor displayed in Figure 4.6. Figures 4.7a and 4.7b display the testing scenarios where the camera's extrinsic parameters were changed.

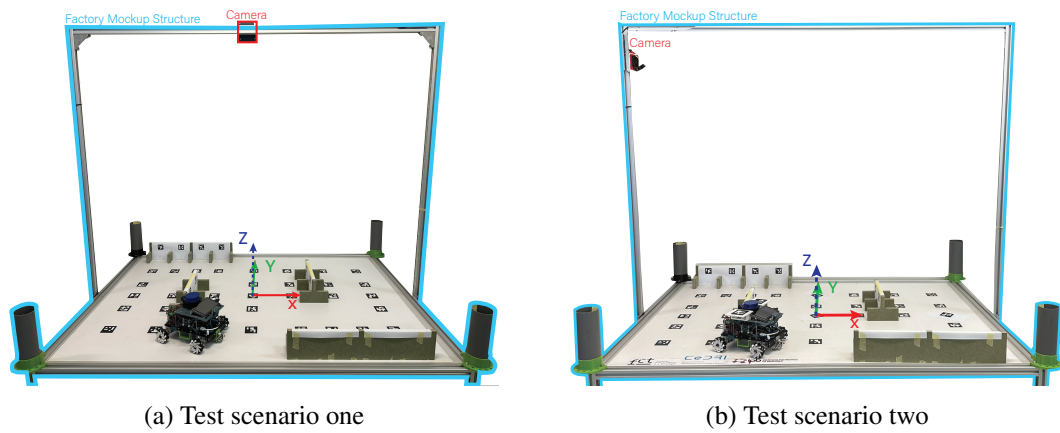


Figure 4.7: RAF structure

The factory floor was printed on canvas and followed the official dimensions of the competition floor, 1.7×1.2 m. In this sense, the markers' real poses follow the accuracy and precision of the industrial printer. In addition, the poses of the markers used as ground truth were the ones

made available on the official map. Thus, Figure 4.7 displays the assembled factory floor on the structure delineated in light blue. The camera, outlined in red, is mounted on the central part of the structure in scenario one, pointing downwards, approximately pointing at the central part of the factory scenario. In scenario two, the camera is positioned at the top-left part of the structure, pointing at the factory floor's origin. There is a coordinate frame drawn on the central part of the floor. It represents the world coordinate frame (on top of ArUco marker 13).

4.3 Data Collection and Preprocessing

This section delineates the methodologies employed for data collection and validation and the statistical analysis tools used in assessing the performance of the systems developed for this dissertation.

4.3.1 Statistical Metrics for System Validation

Evaluating the systems' performance uses a suite of statistical metrics, each designed to provide specific insights into the system's reliability under test. The mathematical formulations of these metrics allow for a quantitative assessment of error characteristics. Below are the detailed descriptions and equations for each metric:

- **Root Mean Squared Error:** **Root Mean Squared Error (RMSE)** is a widely used metric for evaluating model performance. It quantifies the average magnitude of the error between predicted and observed values. **RMSE** is particularly valuable when the error distribution is expected to be normal. **RMSE** is calculated using Equation 4.2:

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2} \quad (4.2)$$

In this equation, y_i denotes the observed values, $\hat{y}_i$ denotes the predicted values, and n is the number of observations. By squaring the errors before averaging, **RMSE** gives more weight to larger errors. The square root is then applied to return the metric to the same units as the original measurements [212].

- **Sample Standard Deviation:** In the context of localization systems, the **STD** is an essential statistical tool used to quantify the variability or dispersion in the measurement errors of a system. This measure is particularly useful in understanding how consistently a localization system performs under different conditions.

STD can be computed using Equation 4.3:

$$STD = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (y_i - \bar{y})^2} \quad (4.3)$$

where y_i represents the individual error measurements (e.g., differences between the estimated positions and the true positions), $\bar{y}$ is the mean of these errors, and n is the number of measurements or trials.

Using the sample standard deviation with $n - 1$ in the denominator—rather than n , a method known as Bessel's correction—helps correct for bias in estimating the population variance from a sample. The adjustment ensures that the estimated standard deviation better reflects the variability expected in the entire population of measurements [213].

- **Mean Absolute Error:** **Mean Absolute Error (MAE)** is another important metric for model evaluation. It measures the average magnitude of the absolute errors between predicted and observed values. **MAE** is appropriate when the error distribution is expected to be Laplacian. **MAE** can be calculated following Equation 4.4:

$$MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i| \quad (4.4)$$

Here, y_i represents the observed values, $\hat{y}_i$ denotes the predicted values, and n is the number of observations. **MAE** is considered more robust than **RMSE** because it is less sensitive to outliers [212].

- **Maximum Absolute Error:** In the context of localization systems, **Maximum Absolute Error (MaxAE)** is an important metric for assessing the quality of pose estimations. It quantifies the most significant discrepancy between the estimated and actual positions, providing insight into the worst-case error scenario. This metric is crucial for applications where the maximum error can determine the system's reliability and operational safety, such as autonomous vehicle navigation, mobile mapping, or robotic path planning.

The formula for **MaxAE**, which highlights the maximum pose error, is given by Equation 4.5:

$$MaxAE = \max_{i=1, \dots, n} (|p_i - \hat{p}_i|) \quad (4.5)$$

Here, p_i represents the actual coordinates, $\hat{p}_i$ denotes the estimated coordinates provided by the localization system, and n is the number of position estimates analyzed. **MaxAE** helps engineers identify and mitigate outlier errors by pinpointing the maximum error.

- **Minimum Absolute Error:** In contrast to **MaxAE**, **Minimum Absolute Error (MinAE)** quantifies the smallest discrepancy between estimated and actual positions, indicating best-case accuracy. It is defined as Equation 4.6:

$$MinAE = \min_{i=1, \dots, n} (|p_i - \hat{p}_i|) \quad (4.6)$$

where p_i represents actual coordinates, $\hat{p}_i$ denotes estimated coordinates, and n is the number of position estimates. **MinAE** establishes the lower bound of system error. Used alongside **MaxAE**, it provides insight into a localization system's performance range.

- **Mean Relative Error:** In the context of assessing the performance of a localization system within a factory environment, **MRE** allows for quantifying the average error normalized by a scale factor—specifically, the maximum dimension of the factory floor. This normalization allows for the measurement of the localization error in proportion to the size of the area, providing a meaningful evaluation of system accuracy relative to the workspace settings.

Adjusted to reflect this context, **MRE** can be computed using Equation 4.7:

$$MRE = \frac{1}{n} \sum_{i=1}^n \frac{|y_i - \hat{y}_i|}{D} \quad (4.7)$$

where y_i denotes the true position coordinates, $\hat{y}_i$ represents the coordinates as determined by the localization system, n is the number of measurements, and D is the maximum dimension of the factory floor.

- **Percentage Change:** This metric quantifies the difference in performance between two distinct methodologies or systems regarding a particular error metric. The percentage change evaluates the relative reduction or increase in an error metric when transitioning from one approach to another, providing a standardized measure of comparative effectiveness.

The percent change can be calculated following Equation 4.8:

$$\text{Percentage Change} = \frac{\text{Metric}_{s1} - \text{Metric}_{s2}}{\text{Metric}_{s2}} \times 100\% \quad (4.8)$$

where Metric_{s1} and Metric_{s2} are the values of a specific error metric under the first and second systems, respectively. A positive value indicates that the second approach has a smaller error metric than the first, signifying an improvement. Conversely, a negative value suggests that the first approach performed better.

These metrics evaluate the system's accuracy and reliability, offering insights into the error profile. The complementary nature of these metrics allows for a thorough understanding of system behavior under several conditions.

4.3.2 Ground Truth Validation Methodology

The reliability of the ground truth system is critical for validating the experimental results. This subsection describes the data retrieval processes and the statistical tools employed to analyze the data collected from the ground truth system. The validation methodology was separated into four stages. The first three stages of tests were performed in scenario one, illustrated in Figure 4.7a, and the fourth stage test was conducted in scenario two, delineated in Figure 4.7b.

- Stage I: 2D position assessment of several floor ArUco Markers.
- Stage II: 2D position assessment of several floor ArUco Markers with 20 cm height support (simulating an average height of an AMR in this scenario).

- Stage III: 2D position assessment of several floor ArUco Markers with some fiducial markers distorted.
- Stage IV: 2D position assessment of several floor ArUco Markers with some fiducial markers heavily distorted and the camera's extrinsic parameters changed.

In addition, it is important to emphasize that in this subsection, when it is referred to the pose unless stated otherwise, it is expected to be assumed to be a 3D pose. When estimating the robots' poses, it is only necessary to compute the 2D pose of them, that is, (x, y, θ) . Also, only the 2D position of the assessed markers will be evaluated (x, y) since their orientation is derived from their 2D position. In this reasoning, the assessed markers for Stage I of the validation methodology were: (0, 3, 5, 10, 12, 15, 18, 21, 22, 26, 30, 35). In turn, for Stage II, some markers could not be verified because of the height support, which made them vanish from the camera's FoV; in this way, the evaluated markers were: (5, 12, 15, 18, 22, 30, 33, 35). Stage III evaluated the same set of markers of Stage I plus marker 34. Stage IV comprised of assessing markers (0, 3, 5, 10, 12, 21, 22, 26, 30, 34). The selection criteria considered their radial distance to the camera's optical center. The markers were selected to be very dispersed to assess the algorithm's accuracy even with radial and transversal distortions that lens cameras have. Markers 4, 13, and 16 were excluded from the selection process to ensure fairness in the comparison since [LMEA](#) uses these markers for pose estimation.

Stage I assessed the accuracy and precision of the novel algorithm, [KMEA](#), proposed in this thesis, and the previous one developed in [117] for baseline comparison, [LMEA](#). In this way, during Stage I, for each marker and algorithm, 256 samples were measured. Figure 4.8 displays the detection software working during the Stage I test.

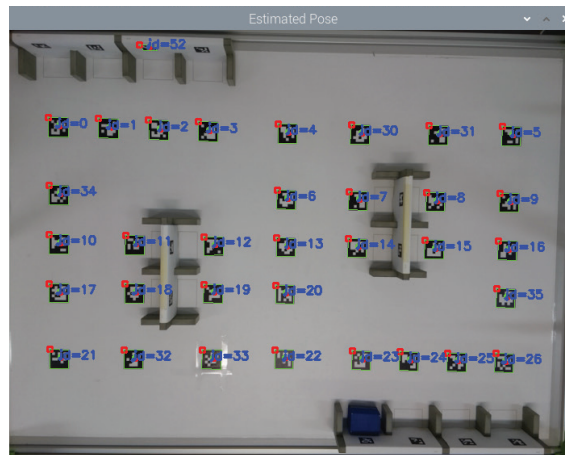


Figure 4.8: ArUco detection software - Stage I validation

Stage II comprised assessing the same metrics; however, changing the z position of the estimated marker to mimic the height of an AMR developed for this scenario. Therefore, a 20 cm height support was designed and 3D printed to accurately position the assessed marker at 20 cm of height. Figures 4.9a and 4.9b show the support outlined in light blue and the support deployed

on the test scenario, respectively. It is also important to note that for the [KMEA](#), each evaluated marker was removed from the algorithm's computations since it would be disingenuous to estimate its pose if it used its ground truth pose during the calculations.

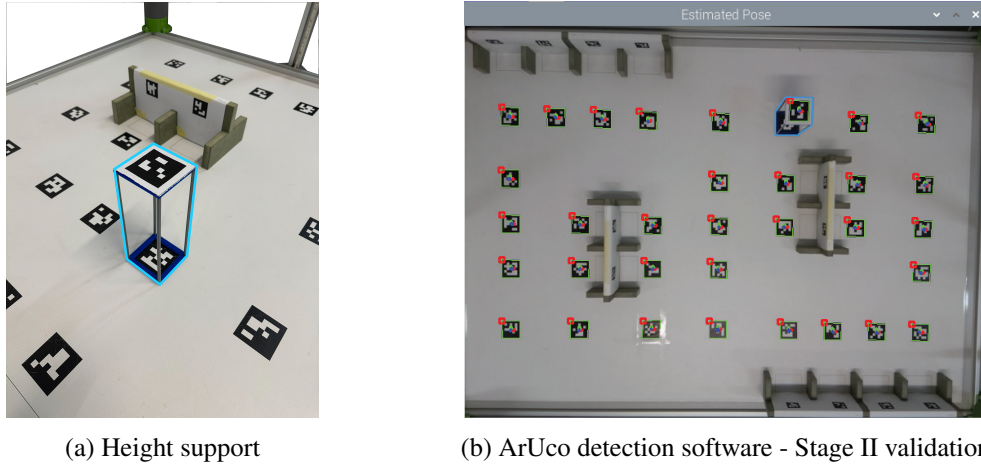


Figure 4.9: Stage II validation

As it is possible to see in Figure 4.9a, the bottom part of the support has a gap to neatly fit an ArUco marker from the floor. This support was put on top of every selected marker of this validation stage. In the same way as Stage I validation, 256 samples were taken for each marker and algorithm, and statistical metrics were computed.

Stage III validation assessed their robustness, one essential aspect of a localization system. [LMEA](#), presented in [117], has one limitation: it depends on three markers to generate the orthonormal basis for the world's frame. However, suppose one or more of them are incorrectly placed or occluded, which can happen in dynamic scenarios such as industrial warehouses. In that case, it can generate errors in estimating the detected markers, including the tracked ones. [KMEA](#) does not have this flaw since it takes the information of all detected markers in a given time to estimate the pose of all the detected markers. Therefore, Stage III of the validation methodology was to print markers 4, 13, and 16 of the same size, place them on top of their respective counterparts on the factory floor, and distort them by moving and rotating to generate errors in the estimation process. This principle will stress out both algorithms, however much more the predecessor to showcase its vulnerability. After that, the two algorithms were assessed similarly to the other parts of the validation methodology, retrieving 256 samples for each marker and each algorithm and computing the respective statistical metrics.

Figure 4.10 shows the Stage III validation scenario configuration, with the printed markers outlined in light blue. It was decided not to distort other markers than the three ones used by the [LMEA](#) (4, 13, 16) for estimation because the test was not going to add any value to the research work since [LMEA](#) would not be affected by them. Instead, it would just misestimate the distorted markers. The point of this test is to show that the [LMEA](#) is heavily dependent on only three markers, which is not robust, in contrast to the novel, which does not have this caveat.

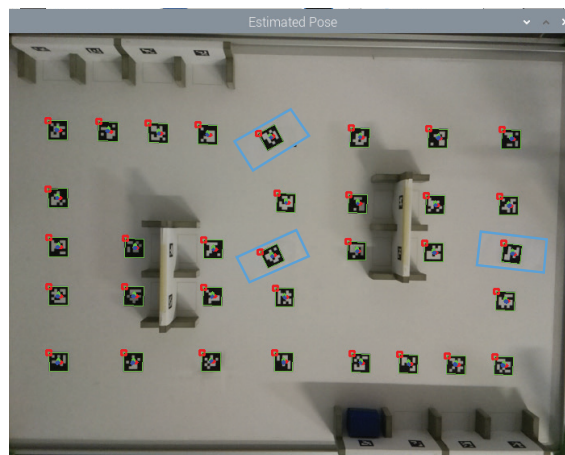


Figure 4.10: ArUco detection software - Stage III validation

Finally, Stage IV of the validation methodology was to perform tests in conditions similar to Stage III; however, the camera's extrinsic parameters changed, as explained at the beginning of this subsection. The objective of the Stage IV tests was to display the robustness of the [KMEA](#) under a different scenario and to illustrate that it is independent of the camera's extrinsic parameters. Stage IV's set of distorted tags remains the same, although with different distortions. Figure 4.11 displays a captured camera frame under scenario two of testing; the light blue rectangles delineate the distorted tags. The AMR was not present on the table during testing.

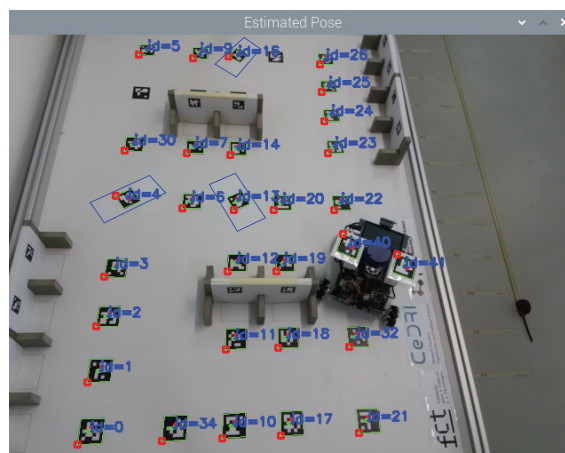


Figure 4.11: ArUco detection software - Stage IV validation

4.3.3 AI-Enhanced Indoor Localization System Validation Methodology

The [AI](#)-Enhanced Indoor Localization System underwent thorough validation through designed tests in simulated and real-world environments. This section outlines the distinct methodologies employed in each setting, integrating detailed descriptions of the technical setup and case study scenarios for system validation.

1. Simulation Testing:

- **Case Study 1:** In this scenario, represented in Figure 4.12, the robot operates in a well-equipped factory section, abundant in localization data but crowded with personnel, potentially affecting localization accuracy. The performance of the original and the RoPM systems in dealing with human obstructions was analyzed.

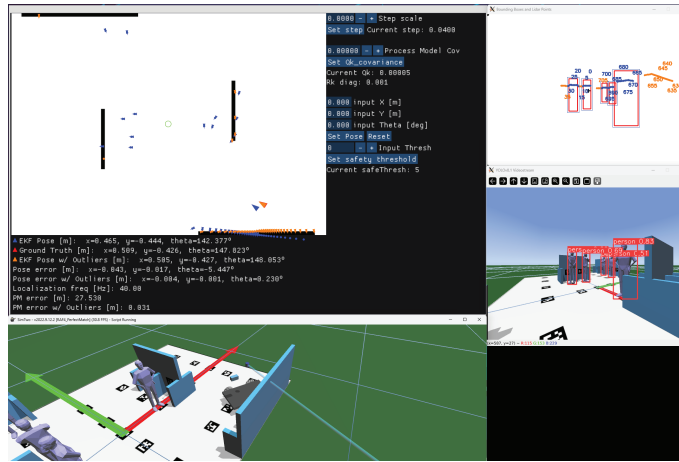


Figure 4.12: Simulation - Case Study I conditions. The red triangle represents the ground truth pose, the orange RoPM, and the blue is the original system.

- **Case Study 2:** The robot started in the corner of the factory floor with few yet adequate data for effective map matching, as shown in Figure 4.13. Personnel movement at the warehouse entrance challenged the localization process, stressing the localization systems' ability to maintain accurate estimations during navigation.

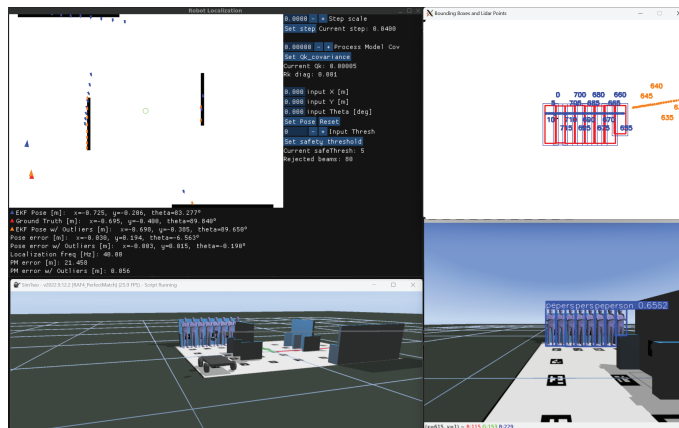


Figure 4.13: Simulation - Case Study II conditions. The red triangle represents the ground truth pose, the orange RoPM, and the blue is the original system.

2. Real-World Testing:

- **Baseline Scenario:** The original EKF-based PM system was benchmarked on the outlier-free factory floor. Figure 4.14 shows the isometric and top view of the scenario.

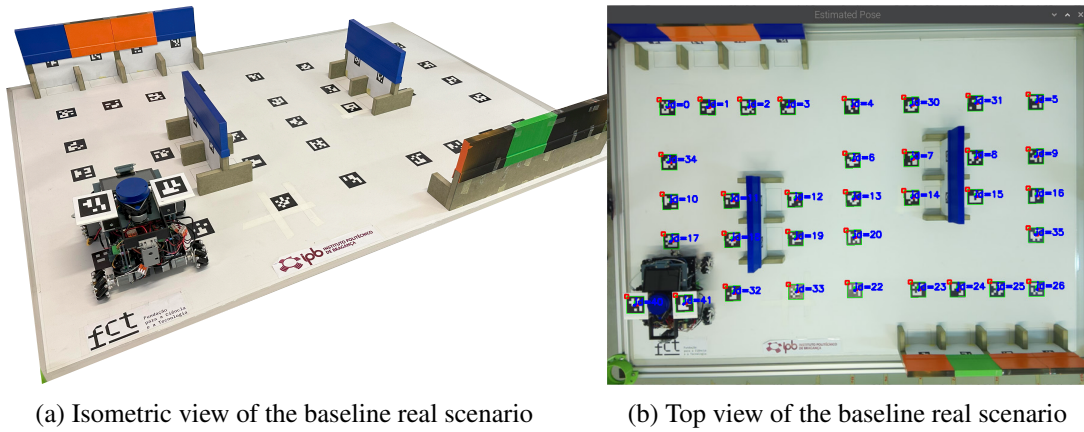


Figure 4.14: Real - Baseline scenario

- **Case Study 1:** This mirrors the second simulation scenario, where the robot navigates laterally at the warehouse entrance amid personnel presence, enabling a direct comparison of simulated results with real-world outcomes, evaluating both systems. Figure 4.15 shows this case study's isometric and top views.

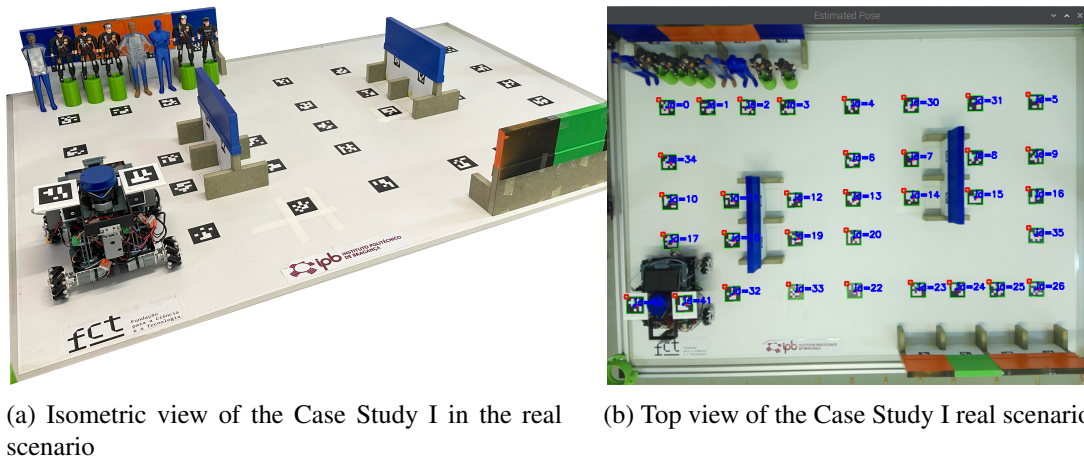


Figure 4.15: Real - Case Study I scenario

- **Case Study 2:** This case study evaluates the localization systems' performance under dynamic conditions with personnel movement. The experiment begins with the **AMR** stationary in the pose represented in Figure 4.15. Initially, subjects remain outside the robot's camera and **LiDAR** coverage. Then, when the test starts, the personnel move laterally, entering the sensor's coverage as Figure 4.15 shows, and after some time, they leave the coverage area. This setup assesses the systems' robustness to environmental changes caused by moving objects. The dynamic nature of this case study prevents static representation, but movement patterns align with those in Case Study 1's real-world scenario (Figure 4.15).

3. Simulation and Real-World Testing Differences:

- The simulation features a simplified setup with only one camera due to simulator constraints.
- The real-world scenario uses a more complex multi-camera setup, and tests are conducted under controlled slow movements to address synchronization issues with the ground truth system and the [LiDAR](#)'s scanning frequency.
- In simulation, tests follow the online [DFD](#) framework as detailed in Chapter 3 (Figure 3.22), whereas real-world tests use the “offline” [DFD](#) (Figure 3.25), ensuring compliance with the designed data flow and integration requirements.

This validation approach thoroughly tests the [RoPM](#) system under diverse and challenging conditions in simulation and real-world scenarios. The performance indicators described in Section 4.3.1 provide a robust framework for assessing improvements brought by [AI](#) integration into the dynamic and complex environments typical of robotic indoor localization.

Chapter 5

Results and Discussion

This chapter presents the results obtained from the comprehensive evaluation, displayed in Chapter 4, of the systems developed as part of this work. In this reasoning, Section 5.1 presents the results of the ground truth system across four distinct case studies and after the improvements made to the algorithm to compensate for the parallax effect of the monocular vision. Section 5.2 focuses on validating the RoPM. Validation results are categorized into two main subsections: simulation and real-world results. The simulation results encompass two case studies. These studies are specifically designed to test both the RoPM and the original EKF-based PM under demanding conditions, allowing for a comparative analysis of their performances. Lastly, the real-world results are divided into three case studies: a baseline scenario and two additional scenarios similar to the simulated ones, challenging the systems.

5.1 Ground Truth Results

This section presents the empirical validation of the algorithms developed throughout this research, demonstrating their capabilities and limitations within controlled and uncontrolled environments.

5.1.1 Stage I. Several Markers Estimated Under Normal Conditions

The first part of the validation assessed both algorithms' error metrics under typical conditions in the testbed. Thus, several markers were chosen to evaluate the whole workspace captured in the images, excluding, for fairness, the specific markers that LMEA uses for localization. In this sense, 256 samples were taken for each analyzed marker for each algorithm. Figures 5.1a and 5.1b display scatter plots of the KMEA and LMEA against the markers' ground truth values, respectively.

Figure 5.1 shows an overall visualization of Stage I validation. As these scatter plots cover the whole workspace and both algorithms are very accurate and precise, they do not show their proper performance with low resolution. For this reason, several metrics were computed. Thus, Figure 5.2 shows the metrics RMSE, MAE, MaxAE, STD, and MRE for both algorithms as bar plots.

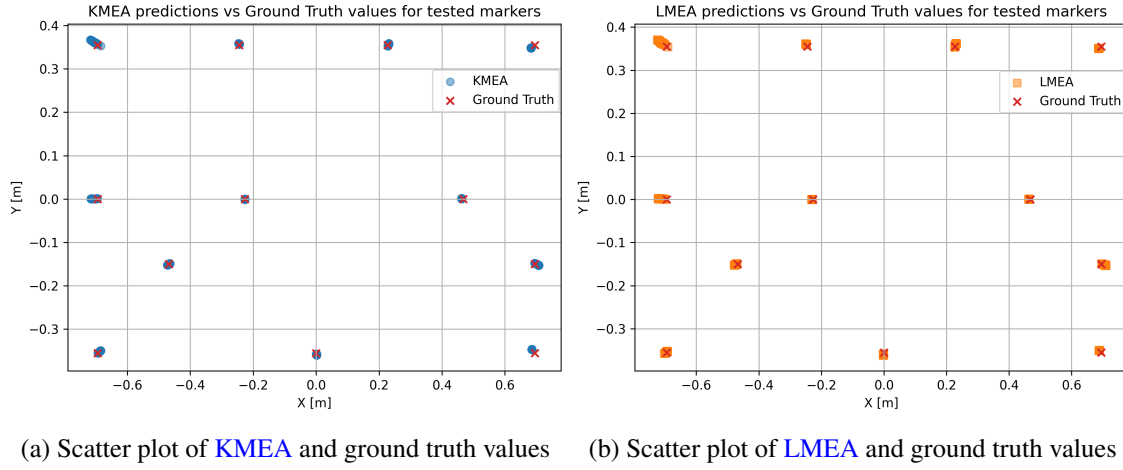


Figure 5.1: Stage I - Comparative scatter plots of markers' position estimations by KMEA and LMEA

For a structured analysis, it is suggested to first visualize the MAE bar plot in Figure 5.2a. As illustrated, both algorithms show consistent errors in the millimetric range, with some markers having a centimetric range. The errors change with the markers because they have different positions in the factory mockup and, therefore, in the image plane, causing potentially diverse effects of deterministic and non-deterministic errors such as quantization, distortions, and noise. In addition, the result bars of markers 0 and 10 showed the highest error metric. Two common characteristics of both markers are that they are on the edges of the camera's FoV and more distant from the camera's optical center, making them more sensitive to the mentioned source of errors and distortions, especially the radial type of distortion. Nevertheless, the maximum MAE of the KMEA is 1.5 cm of error, and the LMEA is 2.2 cm. Regarding MaxAE metric, both algorithms showed a consistent centimetric error below 2 cm, with two errors above 2 cm, markers 0 and 10, and LMEA again underperformed compared to the KMEA. The exact interpretation of MAE can be applied to the RMSE bar plot since they have similar results. The latter metric is more sensitive to outliers because of the residuals' squaring process. However, since the errors were consistent with their averages and the scale is small, both metrics show similar results.

Regarding STD, the deviations were consistently below 0.7 cm, which shows that both algorithms are very precise, with a notable pattern with markers 0 and 10 having the worst results again. Finally, one important metric to assess is the MRE, which is the mean relative error to the highest dimension of the testbed, i.e., the factory mockup, expressed in percentage. The factory mockup has the shape of a rectangle of 1.7 x 1.2 m, and its hypotenuse was used as the maximum dimension. Verifying the plot more closely, one can note that the highest error that the KMEA showed was 0.72%, with most errors being below 0.4%. In contrast, the highest error of LMEA was 1.04%, and most errors were below 0.4%.

Figure 5.3 displays the scatter plot of both algorithms of marker ID 12 in a zoomed scale to show more details about the estimations. The ellipses are confidence ellipses of three standard deviations, and they were generated assuming a normal distribution of the samples. The KMEA

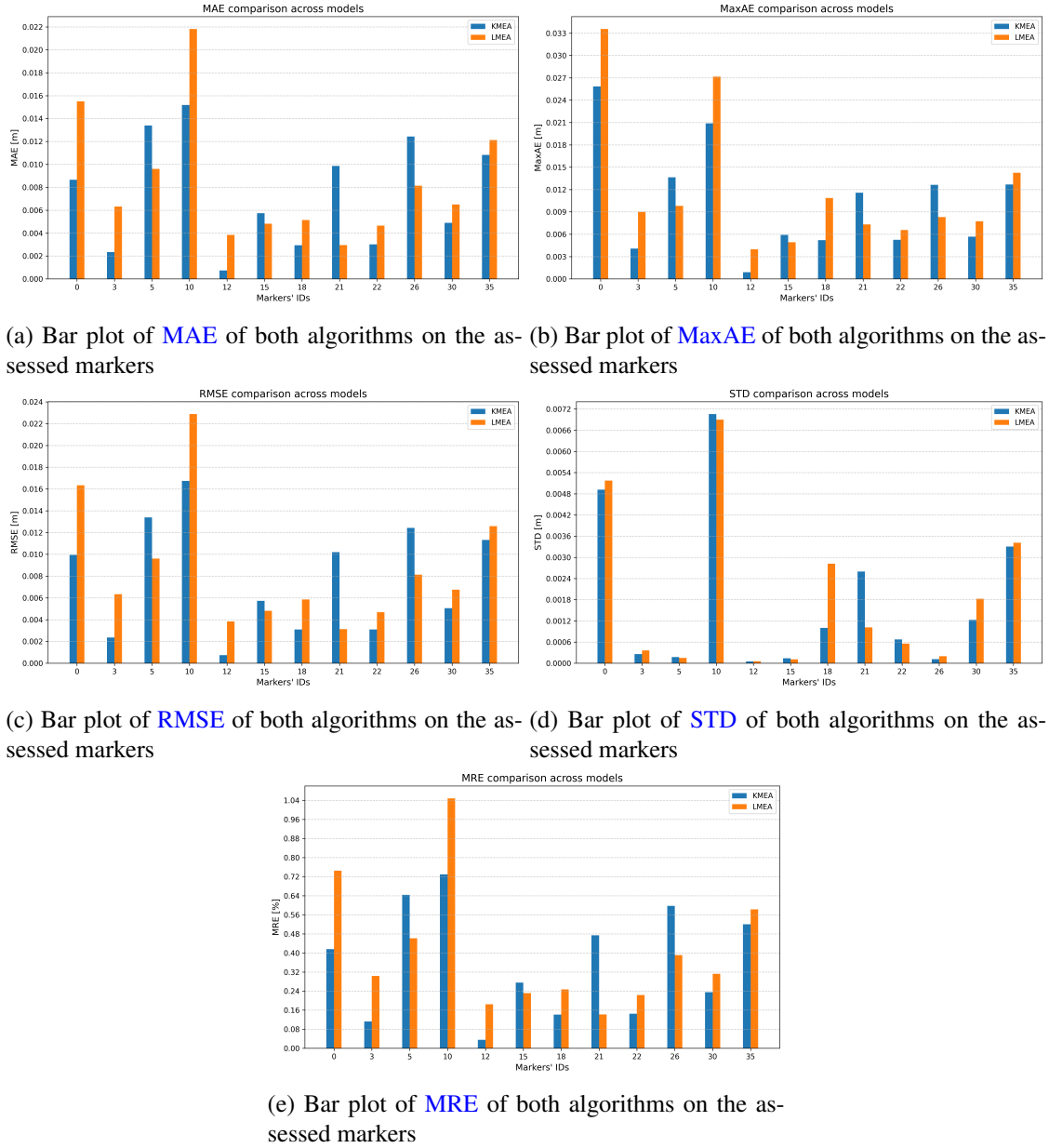


Figure 5.2: Stage I - Bar plots of error metrics of both algorithms described in the test methodology

samples can be seen in blue, and the **LMEA** in orange. The red x displays the ground truth position of marker 12. It is possible to perceive that both sample distributions have low variance and bias relative to the actual value. Finally, Table 5.1 shows the error metrics' summary of Stage I applied to both algorithms for better overall analysis.

As seen in Table 5.1, both algorithms presented accurate and precise error metrics, indicating that they are both suited to be localization algorithms using fiducial markers. This is very emphatic because both MREs are in the order of 0.4%, and their other error metrics are millimetric. In addition, the percentage change between **KMEA** and **LMEA** was computed and presented in the column % Change, where the former performed better in every error metric.

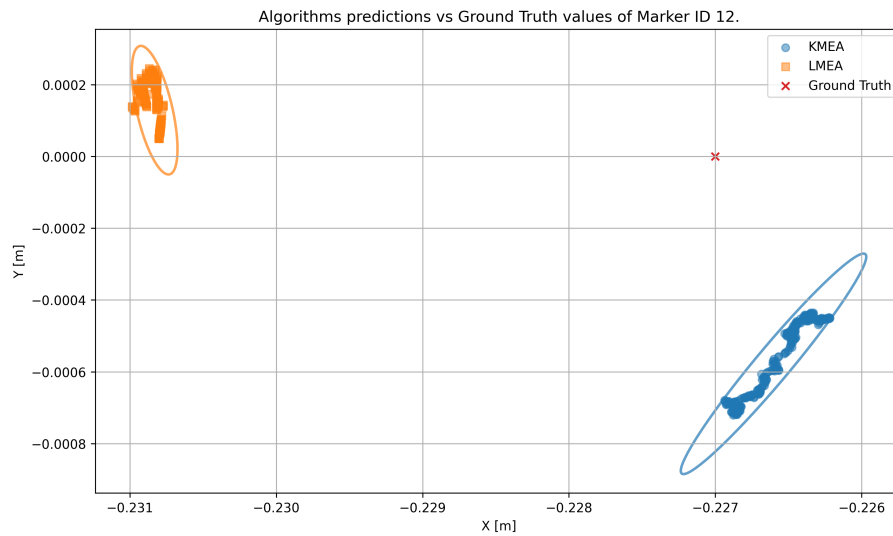


Figure 5.3: Stage I - Scatter plot of both algorithms and the ground truth position of marker ID 12

Table 5.1: Stage I - Comparison of estimation metrics. The percentage change computation is of **KMEA** relative to **LMEA**.

Metric	KMEA	LMEA	% Change
RMSE [m]	0.0078	0.0087	-10.3
MAE [m]	0.0075	0.0084	-11.2
MaxAE [m]	0.0104	0.0120	-13.4
MRE [%]	0.3605	0.4060	-11.2
STD [m]	0.0018	0.0019	-4.8

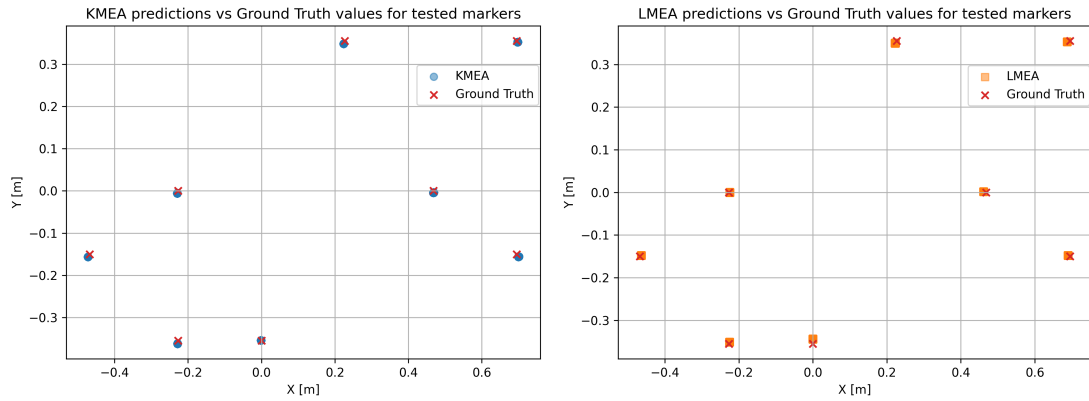
5.1.2 Stage II. Several Markers Estimated With a 20 cm Height Support

Stage II is similar to Stage I; however, a support, displayed in Figure 4.9a, was used to assess the position estimations within a 20 cm height relative to the factory floor since this is the height of the **AMRs** used in this scenario. Thus, as the distance between the center of the assessed marker and the optical center of the camera is decreased, it is expected, in theory, that the error metrics will be reduced compared to the Stage I test since all the sources of error already discussed in Chapter 4 will tend to be lower. In this reasoning, Figure 5.4 shows the scatter plots of both algorithms for the assessed markers.

Since these scatter plots give an overall view of the test, bar plots were also generated to assess the algorithms in the second part more thoroughly. Thus, Figure 5.5 shows all the five error metrics for the Stage II test.

As illustrated in Figure 5.5, all metrics show that the estimations became more precise, denoted by the **STD** being practically zero across all estimated markers, and more accurate, hinted by all the other error metrics that had their values reduced in general. For a more detailed analysis, the errors metrics' summary is presented in Table 5.2.

According to the table, the values were smaller in all metrics. Also, **KMEA** benefited more



(a) Scatter plot of the markers' positions estimations of the KMEA against their ground truth values (b) Scatter plot of the markers' positions estimations of the LMEA against their ground truth values

Figure 5.4: Stage II - Comparative scatter plots of markers' position estimations by KMEA and LMEA

Table 5.2: Stage II - Comparison of estimation metrics. The percentage change computation is of KMEA to LMEA.

Metric	KMEA	LMEA	% Change
RMSE [m]	0.0056	0.0062	-8.6
MAE [m]	0.0056	0.0062	-8.6
MaxAE [m]	0.0059	0.0063	-5.9
MRE [%]	0.2704	0.2960	-8.6
STD [m]	0.0001	0.0001	0

from the decreased marker distance to the camera's optical center. This is corroborated by the percentage change showing that the metrics of the KMEA are smaller than those of the LMEA metrics. However, these differences are negligible, and both algorithms have solid accuracy and precision for this case study.

5.1.3 Stage III. Several Markers Estimated With Orthonormal Basis Markers Distorted

As explained in Chapter 4, Stage III methodology and objective is to evaluate the robustness of both algorithms by exploiting a critical weakness of the LMEA. Because this algorithm considers only three markers to form its orthonormal basis to estimate all the other markers, its performance is tied to them. Therefore, if one or more of them are obstructed, damaged, worn out over time, or even wrongly assembled, it would significantly affect the accuracy and precision of the algorithm. If the obstruction happens for a limited amount of time, this can be mitigated with filtering and other approaches; however, with time, this would degenerate the error propagation. This is not the case for the KMEA, which can achieve the same or even better accuracy and precision with more robustness. This can be even more magnified with a proper error distribution study for adequate filtering and outlier rejection, proving an excellent choice for industrial and scalable applications.

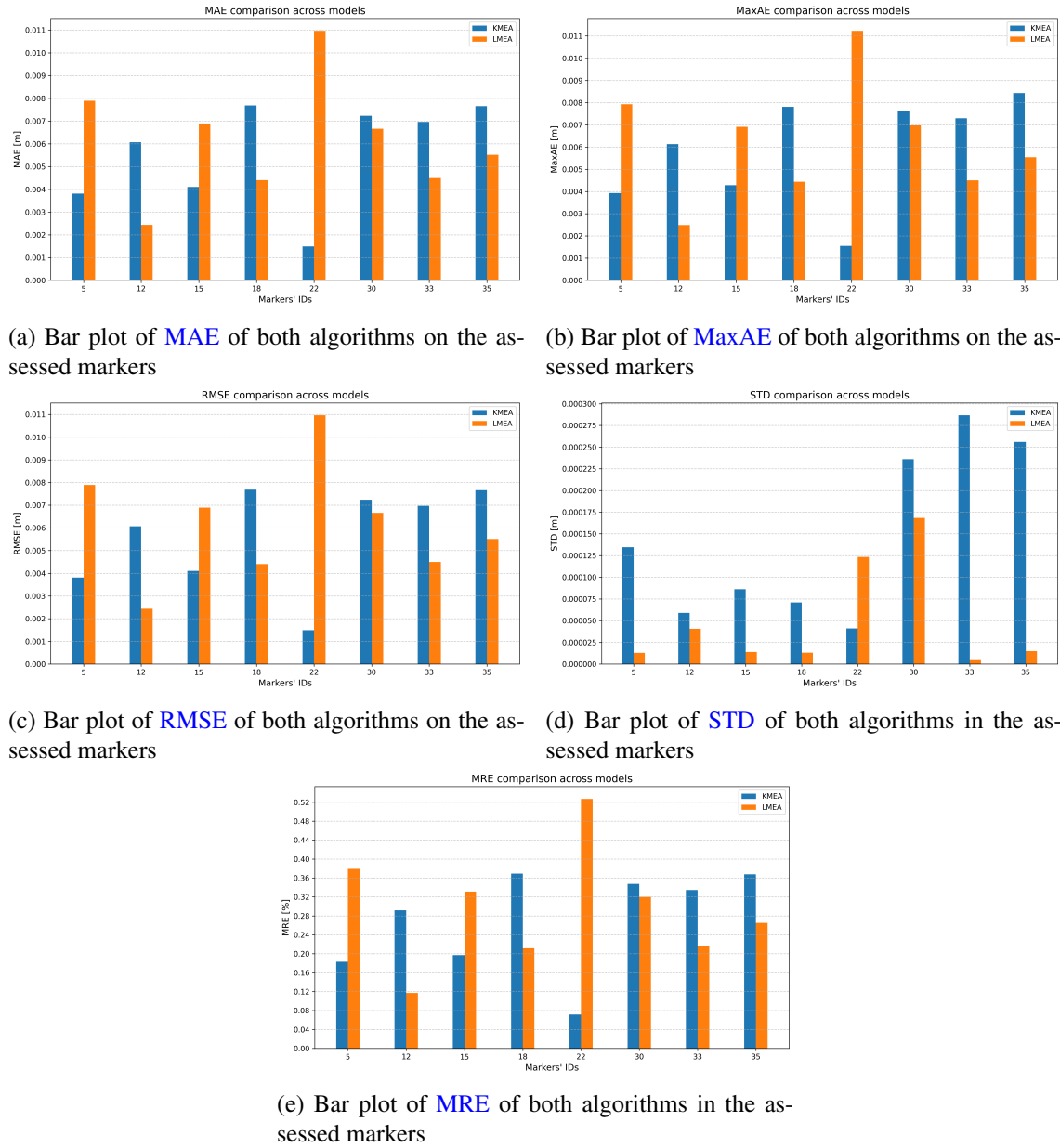
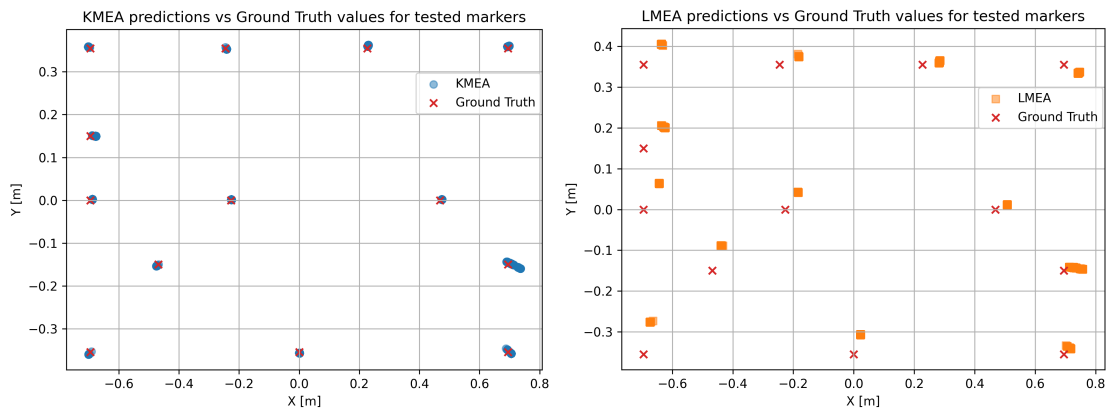


Figure 5.5: Stage II - Bar plots of error metrics of both algorithms described in the test methodology

In this sense, Figure 5.6 shows the scatter plots of the algorithms against the ground truth values for a holistic view of the test.

This illustration visually represents the effects of the distortion of the markers, which affects the accuracy of the approaches, albeit much more impactful in LMEA. This discrepancy is more detailed in Figure 5.7, which shows all the metrics results from the Stage III assessment.

So, after misaligning the markers responsible for the orthonormal basis of LMEA, it is clear that both KMEA and LMEA are affected. However, the latter becomes unfeasible to be considered for a localization system since an error of more than a couple of centimeters could impact the



(a) Scatter plot of the markers' positions estimations of the **KMEA** against their ground truth values (b) Scatter plot of the markers' positions estimations of the **LMEA** against their ground truth values

Figure 5.6: Stage III - Comparative scatter plots of markers' position estimations by **KMEA** and **LMEA**

parking operations in the warehouse mockup, as outlined by the error metrics reaching up to 8.8 cm of **MAE**, **MaxAE**, **RMSE**, and 4% of **MRE**. **KMEA**, despite three misaligned markers, can still yield reasonable results within this case study's constraints, with the **MAE**, **RMSE**, and **MaxAE** all below 1.6 cm. The worst-case scenario occurs with marker 35, where errors reached 4.0 cm, indicating an outlier since its estimation was significantly impacted in **STD** and **MaxAE** and moderately for the other metrics for both approaches. The **MRE** metric of **KMEA** also corroborates the outlier argument, where the metric stayed below 0.8% across all markers. Both algorithms remained precise, however, with a millimetric **STD**. This is expected since statically misaligning specific markers only jeopardizes the estimation bias. Finally, marker 35 showed a massive spike in the **STD** metric for both algorithms, with a 1 cm deviation, compared to the other markers predominantly in the sub-mm scale, strengthening the case that it is an outlier.

Figure 5.8 shows the scatter plot of both algorithms against the ground truth position of marker ID 34, where on it, it is possible to check the variance and bias of the data distributions more thoroughly in a zoomed-in scale by checking the confidence ellipses of three standard deviations. Again, the **KMEA** proved more reliable than **LMEA**.

Table 5.3: Stage III - Comparison of estimation metrics. The percentage change computation is of **KMEA** to **LMEA**.

Metric	KMEA	LMEA	% Change
RMSE [m]	0.0075	0.0607	-87.6
MAE [m]	0.0073	0.0606	-88.0
MaxAE [m]	0.0105	0.0639	-84.0
MRE [%]	0.3496	2.9103	-88.0
STD [m]	0.0013	0.0011	14.0

The data presented in Table 5.3 summarizes the error metrics from Stage III of the test, providing a more transparent analysis overview. This table demonstrates that **KMEA** outperforms

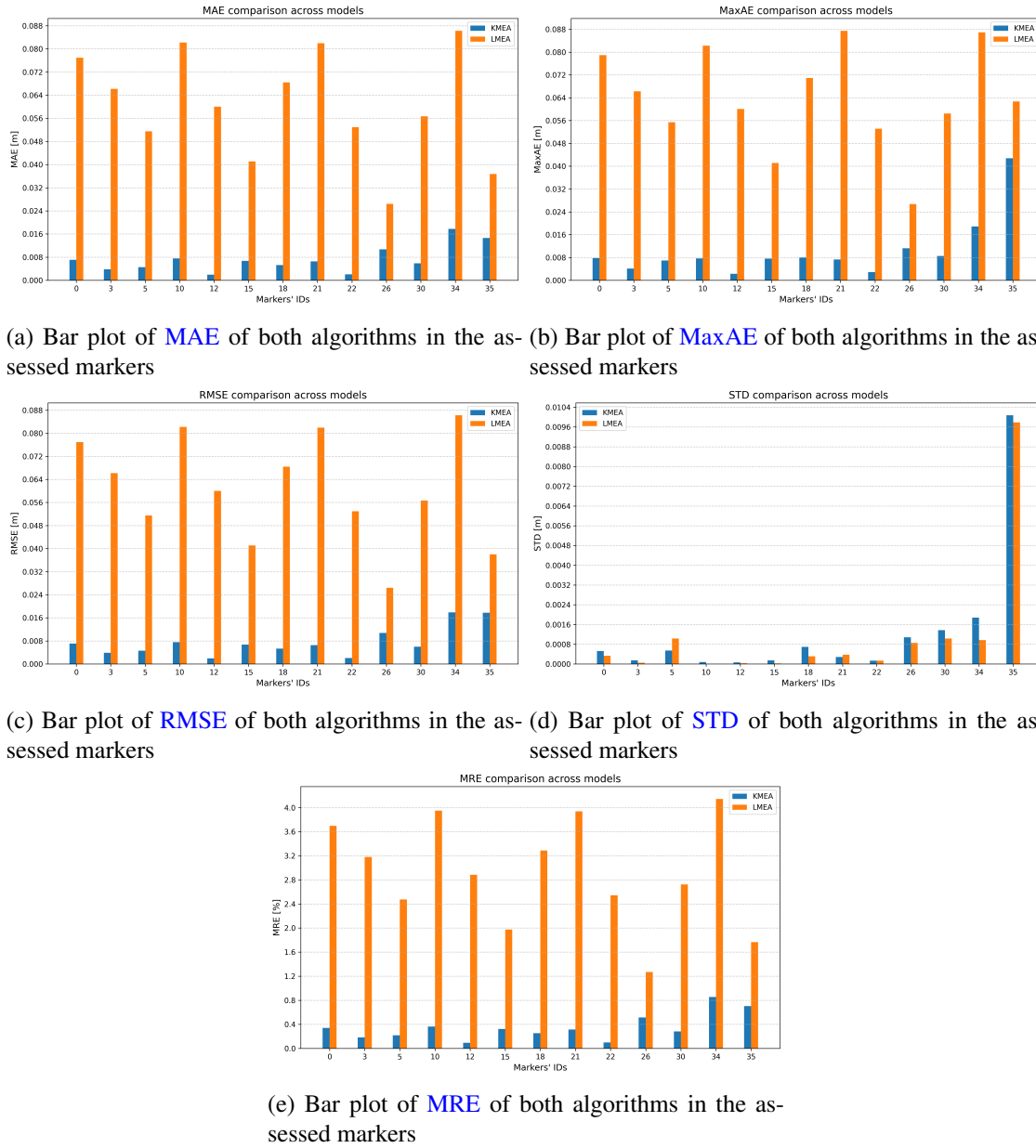


Figure 5.7: Stage III - Bar plots of error metrics of both algorithms described in the test methodology

LMEA in robustness and accuracy. Despite the challenge of having three misaligned markers in the testbed, **KMEA** managed to keep its error metrics within a remarkably low range, under 1 cm and an **MRE** of 0.35%. In contrast, **LMEA**'s performance significantly deteriorated, with errors averaging around 6 cm and an **MRE** of 2.9%. This is emphasized by analyzing **KMEA**'s **MAE**, **MaxAE**, **RMSE**, and **MRE** values, which are consistently lower than **LMEA**. Both approaches showed stable and predictable performance with a high level of error consistency indicated by the similarity of the **MaxAE** regarding the metrics of **RMSE** and **MAE**, highlighting their ability to limit the maximum error to a level near its average errors.

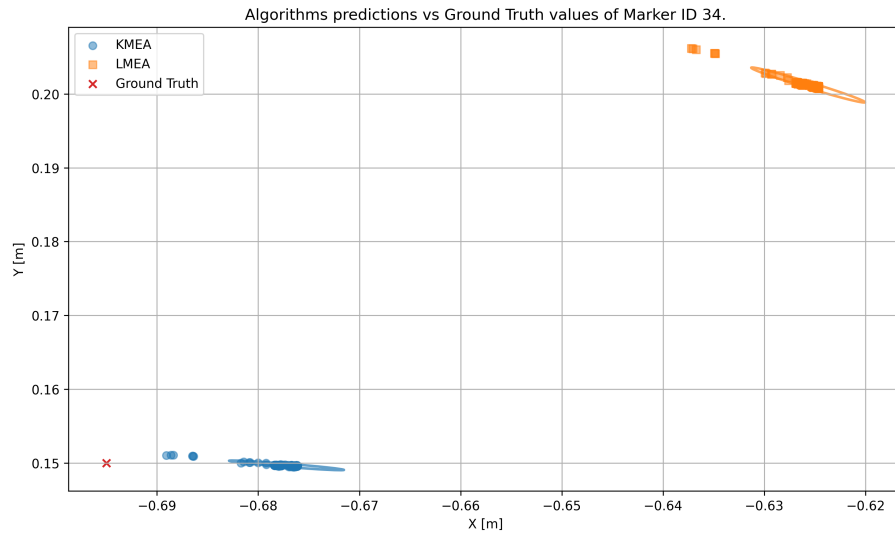


Figure 5.8: Stage III - Scatter plot of both algorithms and the ground truth position of marker ID 34

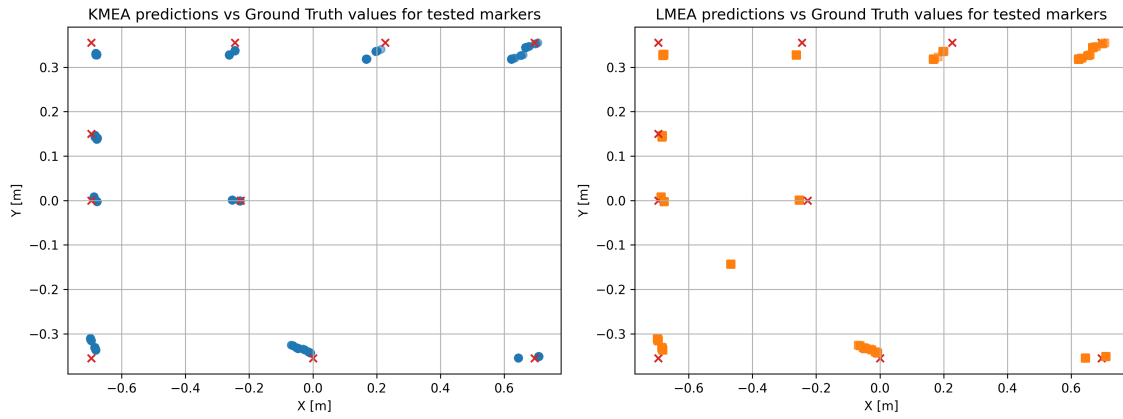
The column labeled % Change in Table 5.3 is particularly revealing. It shows that, except for the **STD**, **KMEA**'s error metrics are, on average, 86.9% lower than those of **LMEA**. This indicates a significant superiority in the accuracy and consistency of **KMEA**. The negligible difference in standard deviation between the two algorithms is consistent with previous observations and analyses discussed in this section.

5.1.4 Stage IV. Several Markers Estimated With Orthonormal Basis Markers Distorted, and the Camera's Pose Changed

Stage IV test had conditions similar to Stage III, with some exceptions. The main one was the camera's extrinsic parameters being changed. Since the algorithm does not need external calibration, the scenario change did not influence the results. In addition, the distortion of the orthonormal basis was much more substantial than in Stage III, stressing more both algorithms. Finally, the set of the assessed markers was also changed. In this sense, Figure 5.9 shows the scatter plots of both algorithms against the evaluated markers' ground truth values.

As illustrated in Figures 5.9a and 5.9b, the dispersion of values was increased, and the bias relative to the true values was also heavily affected, compared to previous tests. For a more thorough analysis, Figure 5.10 shows the five error metrics for the evaluated markers.

The effect on the performance of the algorithms was more intense, evidenced by the displayed bar plots in Figure 5.10. Considering the **MAE** analysis in Figure 5.10a, the error ranged from 15 to 60 cm for the **LMEA**. On the other hand, the **KMEA** still managed to maintain reasonable levels, given the scenario's extreme conditions, with the **MAE** ranging up to 5 cm of error. Analyzing Figure 5.10b, which displays the **MaxAE** error metrics, one can see that the maximum errors of **LMEA** reached absurd levels in the metric scale, indicating that the distortion made it useless in this scenario, whereas **KMEA** varied approximately in the range below of 10 cm. The results



(a) Scatter plot of the markers' positions estimations of the **KMEA** against their ground truth values (b) Scatter plot of the markers' positions estimations of the **LMEA** against their ground truth values

Figure 5.9: Stage IV - Comparative scatter plots of markers' position estimations by **KMEA** and **LMEA**

remained consistent in the other metrics, such as **MRE**, in Figure 5.10e, that **LMEA** errors ranged approximately from 3% to 30% and **KMEA**'s errors all were below 2.5% which is an acceptable metric given the circumstances. In addition, the **RMSE** results, displayed in Figure 5.10c, showed absurd errors for **LMEA**, where **KMEA** still manages to have adequate results in the centimetric scale. Finally, the **STD** metric, shown in Figure 5.10d, shows that some markers were not consistently detected in this specific test because of the camera's different pose and illumination. This caused **LMEA** to generate more dispersed estimations, increasing the **STD** metrics significantly with unacceptable values. **KMEA**, on the other hand, maintained its consistency in the reasonable, given the extreme scenario, centimetric scale.

Figure 5.11 shows a scatter plot of the ground truth position of tag 30 and estimations of both algorithms with confidence ellipses of three standard deviations. In this illustration, the elongated confidence ellipses were caused by the near-singular covariance matrices of both data distributions. This may be attributed to low data variability, seen by both algorithms' data distribution. Specifically, **LMEA**'s data distribution shows two clusters, potentially caused by quantization issues. Table 5.4 displays the average of the error metrics across all evaluated markers and the percentage change of **KMEA** relative to **LMEA**.

Table 5.4: Stage IV - Comparison of estimation metrics. The percentage change computation is of **KMEA** to **LMEA**.

Metric	KMEA	LMEA	% Change
RMSE [m]	0.0322	0.4242	-92.4
MAE [m]	0.0295	0.3000	-90.2
MaxAE [m]	0.0451	0.6451	-93.0
MRE [%]	1.4154	14.4180	-90.2
STD [m]	0.0117	0.2972	-96.1

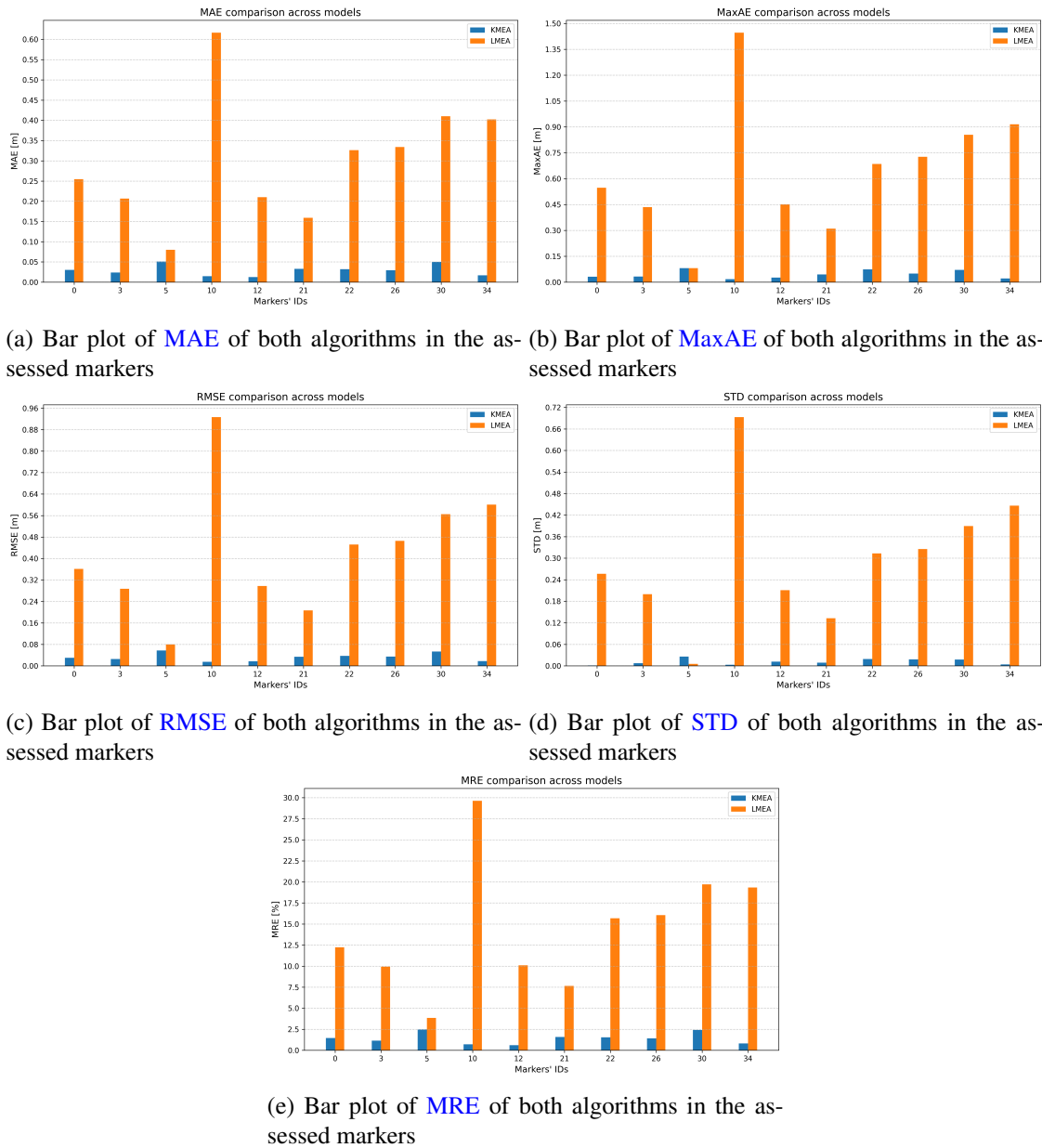


Figure 5.10: Stage IV - Bar plots of error metrics of both algorithms described in the test methodology

As described in Table 5.4, **KMEA** maintained acceptable metrics even in scenario two that provided absurd conditions. This is evidenced by average error metrics ranging from 1 cm to 4.5 cm and a **MRE** of approximately 1.4%. **LMEA**, on the other hand, had worse results ranging from 30 cm to 64 cm and an **MRE** of approximately 14.4%. Finally, the % Change column shows an average of 92% reduction in the error metrics of **KMEA** relative to **LMEA**.

This section shows that integrating the Kabsch Algorithm aligns detected markers with their ground truth positions, enhancing precision, robustness, and accuracy. This robustness is crucial in dynamic environments, such as industrial settings, where the movement of machinery and

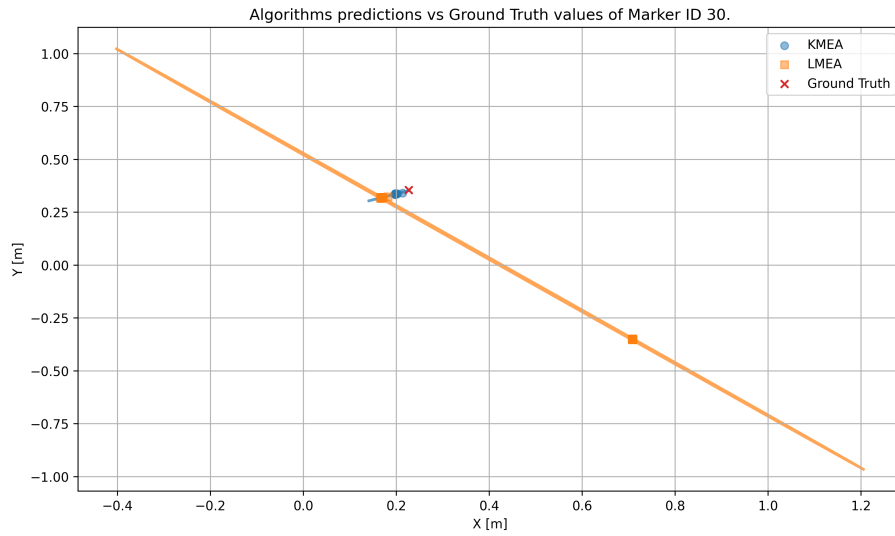


Figure 5.11: Stage IV - Scatter plot of both algorithms and the ground truth position of marker ID 30. Elongated confidence ellipses indicate near-singular covariance matrices.

personnel can lead to marker damage or obstruction. The **KMEA** was evaluated in an **I4.0** automated warehouse mockup, simulating the **RAF** competition environment. It outperformed the previous **LMEA** by using data from all visible markers rather than relying on a few select markers; it also provides the independence of the camera's extrinsic parameters and less sensitivity to marker distortion. These features address the previous system's limitations, which depended on specific markers, and significantly improve robustness and adaptability in dynamic environments where marker visibility might be compromised. In addition, **KMEA** achieved comparable or superior accuracy to the **LMEA** in tests under normal conditions (Stages I and II). This finding is notable, given the increased impact of radial distortion away from the image center, since **LMEA** uses markers near the camera center and **KMEA** uses all detected markers within a given frame. Furthermore, the **KMEA** proved robust even with marker misplacement and changing the camera extrinsic parameters. This was represented in Stages III and IV of the test methodology. **KMEA** maintained good accuracy and precision, whereas the **LMEA**'s performance declined drastically to unacceptable levels in scale terms. This was corroborated by an average percentage decrease of the assessed metrics of 86.9% in the Stage III test and 92% in the Stage IV test of **KMEA** relative to **LMEA**. This research confirmed the potential of a marker-based localization system for indoor **AMR** localization within the **I4.0** context. Its effectiveness, scalability, robustness, and broad applicability, extending to augmented reality and visual localization, make it a valuable asset in sectors requiring precise indoor localization.

5.1.5 Z-Correction Algorithm Results

After noticing the deviated estimations explained in Section 3.2.6, which happened while obtaining the results in this chapter, the "Z-Correction" algorithm was implemented in **KMEA**, and some tests were performed to assess its functionality. Particularly, these deviations were noticeable only

when estimations were far from the image's center. That is why it was not noticed during the four-stage testing, where some behaviors that could have been attributed to the parallax effect were attributed to quantization issues. They were only caught when tests with the robotic platform were performed. During these continuous dynamic tests, the deviations were highlighted when the robot moved far away from the image's center. Figures 5.12 and 5.13 illustrate the impact of parallax on the estimation accuracy before and after applying the compensation algorithm on KMEA. In these illustrations, several markers were assessed with the 20 cm height support on several markers' on the factory floor.

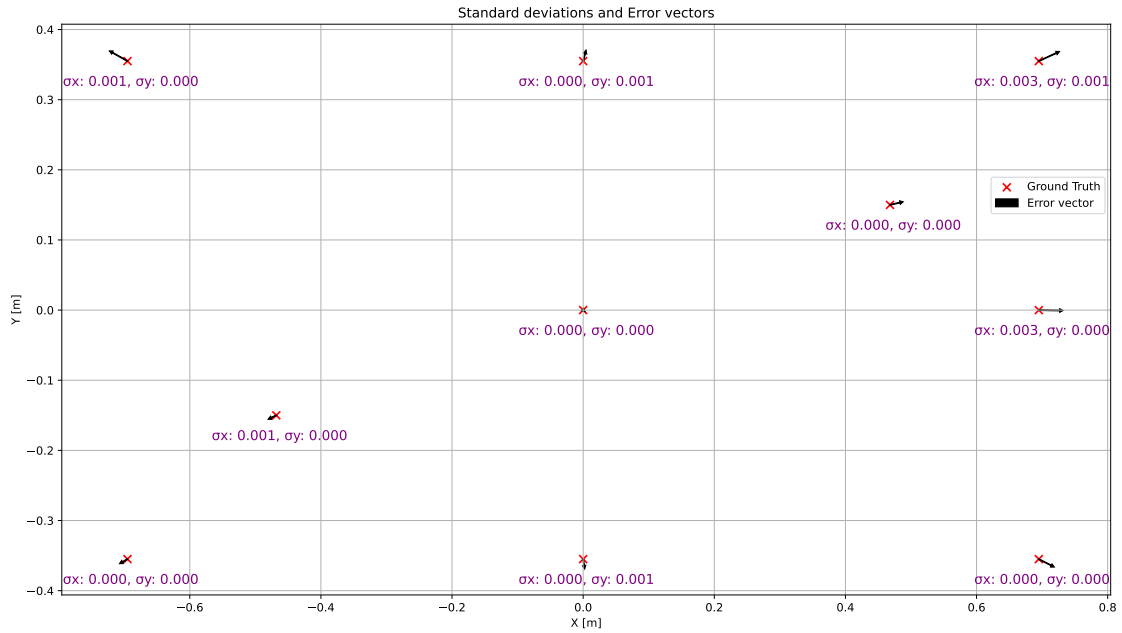


Figure 5.12: Illustration of the parallax effect on KMEA pose estimation without compensation. The STD of x and y are represented by σ_x and σ_y in meters, respectively. The black arrows represent the error vectors from the estimated to the true positions (red x symbols).

As shown in Figures 5.12 and 5.13, the plots show the error vectors from the estimated to the real positions of the markers before and after the parallax compensation algorithm, respectively. According to the illustrations, the effects of the parallax on the markers for pose estimation were practically nullified by the proposed compensation algorithm, considering the system's significant digits. This phenomenon is clearly shown by the norm of the negligible error vectors in Figure 5.13 and the STDs' suppression to zero. Further improvements are not feasible since the system seems to have reached its precision limits, and further errors are quantization or non-deterministic errors. Finally, to have a quantitative and detailed analysis, Table 5.5 displays the comparative metrics, described in Section 4.3.1, computed for these tests.

As seen, the “Z-Corrected” system significantly improves the error metrics across the metrics, practically reaching the low end of the millimeter scale for all the assessed tags dispersed throughout the factory floor. For a more summarized analysis, Table 5.6 displays the average percentage reduction of all the metrics across all the assessed tags for the normal and Z-corrected KMEA.

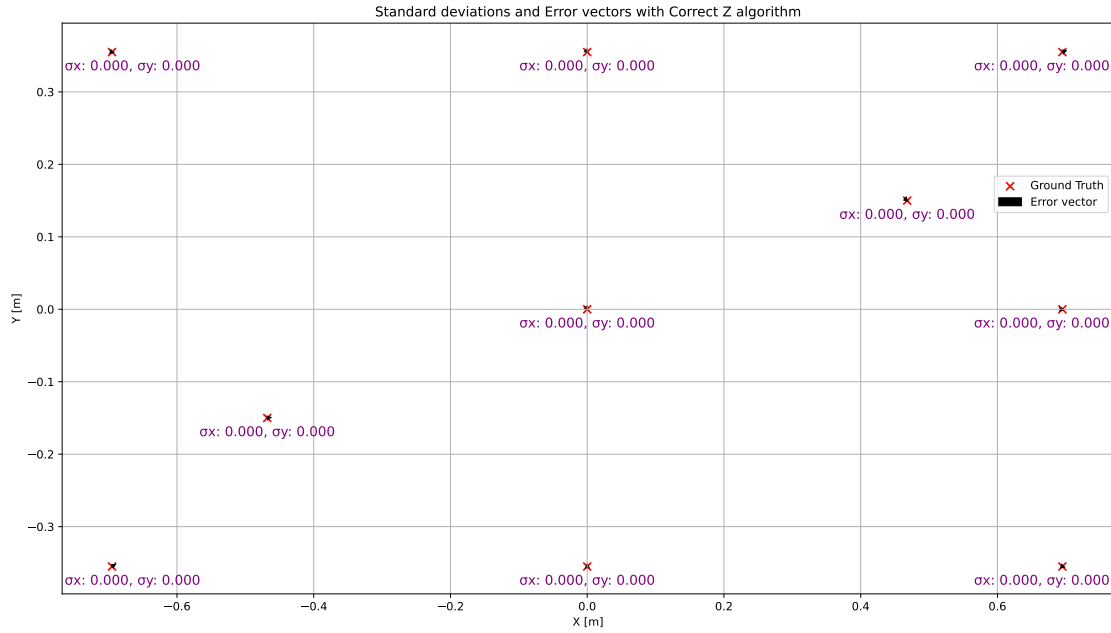


Figure 5.13: **KMEA** pose estimation after parallax compensation. The **STD** of x and y are represented by σ_x and σ_y in meters, respectively. The black arrows represent the error vectors from the estimated to the true positions (red x symbols).

Table 5.5: Statistical metrics for the normal and Z-corrected **KMEA** systems

Tag	MAE [m]		RMSE [m]		MaxAE [m]		STD [m]	
	Normal	Z-Corr	Normal	Z-Corr	Normal	Z-Corr	Normal	Z-Corr
Tag 18	0.014	0.005	0.014	0.005	0.016	0.005	0.00082	0.00003
Tag 26	0.026	0.003	0.026	0.003	0.028	0.003	0.00031	0.00008
Tag 5	0.034	0.006	0.034	0.006	0.038	0.006	0.00333	0.00013
Tag 0	0.031	0.003	0.031	0.003	0.032	0.003	0.00076	0.00012
Tag 22	0.014	0.003	0.014	0.003	0.020	0.003	0.00149	0.00010
Tag 21	0.014	0.006	0.014	0.006	0.014	0.006	0.00004	0.00004
Tag 8	0.021	0.001	0.021	0.001	0.021	0.001	0.00035	0.00015
Tag 4	0.016	0.002	0.016	0.002	0.020	0.002	0.00078	0.00033
Tag 16	0.037	0.002	0.037	0.002	0.038	0.002	0.00309	0.00023
Tag 13	0.004	0.001	0.004	0.002	0.004	0.002	0.00019	0.00016

Table 5.6: Metrics' average percentage reduction for the Z-Corrected **KMEA** system compared to the normal

	MAE (%)	RMSE (%)	MaxAE (%)	STD (%)
Average Reduction	79.8	79.8	80.2	67.3

The average percentage reduction in the metrics reached remarkable numbers from 67 to 80%.

5.2 AI-Enhanced Indoor Localization System Results

This section explores the performance of the AI-enhanced indoor localization system, **RoPM**, under several simulated and real-world conditions designed to evaluate its capabilities in challenging dynamic environments. The validation approach includes detailed case studies that expose **RoPM** to a range of challenging scenarios, highlighting its robustness and adaptability compared to the original **EKF**-based **PM**. Each case study tests the system's response to environmental outliers and dynamic changes, offering insights into its operational effectiveness. The following subsections will detail the simulation and real-world tests conducted and discuss their findings. It is important to note that for each case study, the statistical analyses in the results tables correspond to the "Region of interest" ranges marked in the respective test plots if they are present; if not, it considers all the data ranges from the plots.

5.2.1 Simulation

Several tests were made to validate the algorithm, and two case studies were selected for display. The first case is a static scenario where the robot is stopped, trying to localize itself with environmental outliers. Data logs of the original localization system and **RoPM** were collected. The outliers chosen for this case study were personnel since it is aligned with the scenario, and the default YOLOV8.1~8.2 model can already infer these classes with good precision. In the second case study, the robot performs a predefined path in the factory scenario where substantially more outliers are arbitrarily displaced in the scenario. It is necessary to restate/recall the **PM** loss definition from Chapter 3: the mean absolute Euclidean distance between all **LiDAR** beams and their nearest walls.

5.2.1.1 Case Study I

In this case study, the robot was placed in a challenging position to locate itself, where five personnel were in the middle of the scenario, jeopardizing the data for map matching. In this way, several samples were taken in this test, and the absolute errors of each system's state variables and the **PM** loss are shown in Figures 5.14 and 5.15. As shown in these illustrations, the absolute errors for all state variables and the **PM** loss were substantially higher in the original system than in **RoPM**. This happens because the outliers were present in the middle of the map, obstructing the static features where the original **PM** used to match the pose. Despite being high, the error converged to a particular value because it could use other walls without outliers to minimize the loss function, **PM** loss, shown in Figure 5.15. In contrast, **RoPM** successfully identified the outliers and rejected the flawed measurements. This is shown in the top-view map in Figure 4.12, where the **LiDAR** projected points of the original system (blue points) were hitting the outliers, whereas the **RoPM** (orange points) were not. This is also seen in the array of **LiDAR** points in the bottom-right wall of the map, where the array of blue points is rotated relatively to the wall (because the original system had a high error in θ). In contrast, the array of orange points is correctly displaced along the

bottom wall, demonstrating that **RoPM** had a very low error in the state variables. In the top-right window of Figure 4.12, it is possible to see bounding boxes drawn around the outliers in the image's plane perspective. In this window, the original system considered all points for localization, whereas **RoPM** only considered the valid (orange) ones.

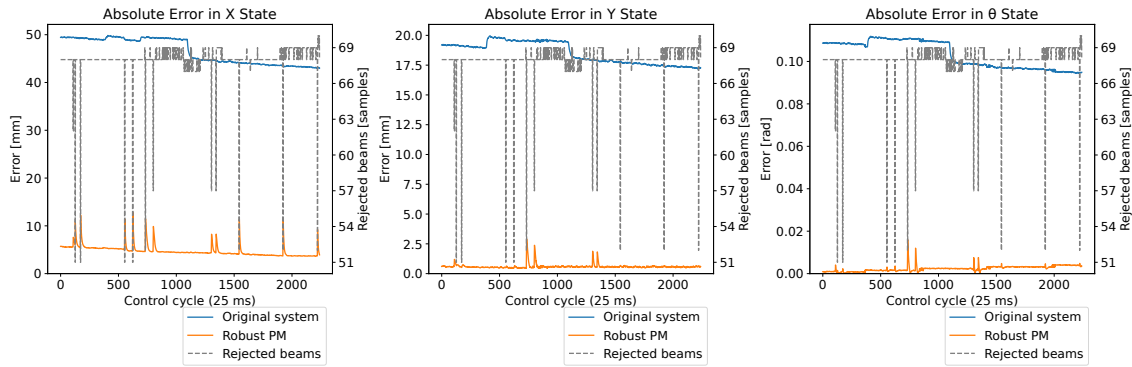


Figure 5.14: Case Study I: Robot state variables' absolute errors comparison of the original and **RoPM** systems. The left y-axis represents the absolute errors for both systems in mm, and the right y-axis represents the number of rejected beams at a given control cycle (dashed gray lines). The x-axis represents the control cycles, with samples taken every 25 ms. The valleys of the rejected beam lines, inversely correlated with the peaks in the **RoPM** algorithm, are caused by frames in which the **AI** did not detect any outliers.

Comparing both systems in Figure 5.14, it is clear that **RoPM** shows superior accuracy and robustness in dynamic environments. **RoPM** in the x state stays in the mid 5 mm whereas the original system varies from 45 ~ 50 mm; the same behavior is present in the y and θ state variables, where the **RoPM** stays in the lows 1.25 mm and 0.01 radians, respectively. In contrast, the original system stays in the 17.5 ~ 20.0 mm and 0.10 ~ 0.12 radians range of y and θ states. Another important feature to analyze is the second axes of the plots, which show the rejected beams at a given sample. During the test, **RoPM** rejected beams in the low 67 beams range. There are several moments where the number of rejected beams dropped for a few control cycles, and this is because of the reliability of detection of the **YOLO** model. This analysis shows a clear correlation between the localization performance in dynamic scenarios and the performance of the outlier-rejection **AI**.

As displayed in Figure 5.15, the **PM** loss is also correlated with the number of rejected beams. As explained in Chapter 3, the **PM** loss is the average of all the **LiDAR**'s beam measurements to the closest wall after applying the M-estimator. In this reasoning, if the projected pose of the **LiDAR** beam is not too high, the M-estimator will not reject it, and the minimizer of **PM** will try to adjust the robot's pose ($[x, y, \theta]$) to decrease the loss function, which by consequence, will correct the distance of this **LiDAR** beam to the closest wall. With all this in mind, this illustration shows that **RoPM**'s loss stays near the 0 mm mark given the precision shown in the plot, where only in some occurrences it increases a little in the mid 5 mm range, which is directly related to the **YOLO** model stopping detecting the outliers. The original system, however, varied in the range of 27 – 30 mm.

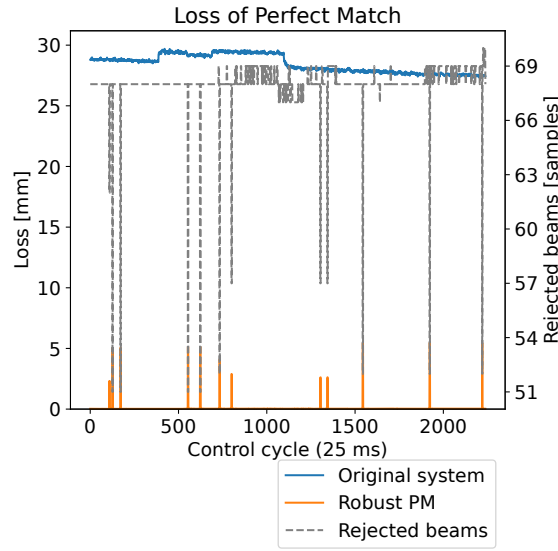


Figure 5.15: Case Study I: **PM** loss of the original and **RoPM** systems. The left y-axis represents the **PM** loss for both systems in mm, and the right y-axis represents the number of rejected beams at a given control cycle (dashed gray lines). The x -axis represents the control cycles, with samples taken every 25 ms. The valleys of the rejected beam lines, inversely correlated with the peaks in the **RoPM** algorithm, are caused by frames in which the **AI** did not detect any outliers.

Tables 5.7 and 5.8 show the computed statistical indicators, presented in Section 4.3.1, for both systems.

Table 5.7: Case Study I error metrics. Original system.

	MAE	STD	MaxAE	MinAE	RMSE
X [mm]	46.57	2.66	49.84	43.00	46.64
Y [mm]	18.53	0.92	19.94	17.18	18.55
θ [rad]	0.10	0.01	0.11	0.09	0.10
PM [mm]	28.46	0.74	29.70	27.32	28.47

Analyzing Table 5.7, the error metrics of the original system for the $[x, y]$ states variables lie approximately from 1.7 to 5 cm for all metrics except a slight standard deviation. Nevertheless, the error of five cm is not feasible for reliable operation, considering the relative size of the factory mockup.

Table 5.8: Case Study I error metrics. **RoPM** system.

	MAE	STD	MaxAE	MinAE	RMSE
X [mm]	4.78	1.16	12.80	3.64	4.92
Y [mm]	0.60	0.21	2.92	0.44	0.63
θ [rad]	0.00	0.00	0.02	0.00	0.00
PM [mm]	0.10	0.54	5.46	0.01	0.55

In contrast, in Table 5.8, **RoPM** maintained millimetric in all error metrics except **MaxAE**,

which was in the low centimetric scale, which is expected since outliers were not detected in every frame, culminating in this value. Even so, values in this scale can be worked with filtering techniques since most of the error data distribution is not in this range, as shown by the other metrics. These results show that **RoPM** can reliably operate under these conditions.

Table 5.9: Case Study I: Percentage difference of the **RoPM** relative to the original system

	D_{MAE}	D_{STD}	D_{MaxAE}	D_{MinAE}	D_{RMSE}
X [%]	89.74	56.39	74.33	91.54	89.46
Y [%]	96.77	77.49	85.36	97.42	96.58
θ [%]	97.69	79.83	85.72	99.99	97.37
PM [%]	99.66	26.09	81.62	99.96	98.06

Table 5.9 shows the percentage difference, D_{metric} computed as $|RoPM_{metric} - PM_{metric}| / PM_{metric}$ where *metric* is any of the statistical metrics evaluated. In all error metrics, **RoPM** significantly improved in the $[80, 99]\%$ range, approximately. Considering **MAE** and **RMSE**, for instance, it reduced approximately from 89 to 99%. Thus, it is possible to state that **RoPM** improved consistently upon the original system's accuracy, precision, and robustness. Finally, Figure 5.16 shows a pose comparison between the ground truth and the two localization systems to have a wider visual of where the estimates varied in the XY plane.

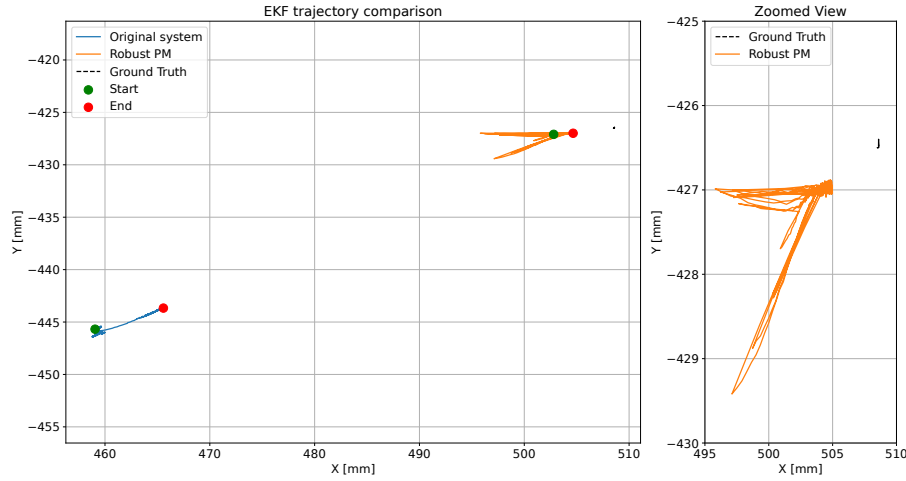


Figure 5.16: Case Study I: XY plot of the ground truth, the original, and the **RoPM** systems. The illustration shows the estimated pose of the robot during the stationary test. **RoPM** demonstrates lower bias with comparable low dispersion to the original system. The green and red circles indicate both systems' trajectories' start and endpoints.

As shown in Figure 5.16, the original system's bias was significant towards the true value, whereas the **RoPM** had a significantly lesser bias. Visually, the precision of both systems appears similar, corroborated in Tables 5.7 and 5.8, where the original system varied in the 2.5 mm range and **RoPM** in the 1 mm range for the x state.

5.2.1.2 Case Study II

For Case Study II, the **AMR** performed a predefined trajectory in an outlier-filled environment (7 personnel) in front of the walls, affecting both systems' performance. The **AMR** then started the path from the bottom-left corner of the warehouse's workspace and proceeded to go to the bottom-middle location of the warehouse. Figure 5.17 displays the trajectory performed by the robot and the two localization systems compared to the ground truth trajectory.

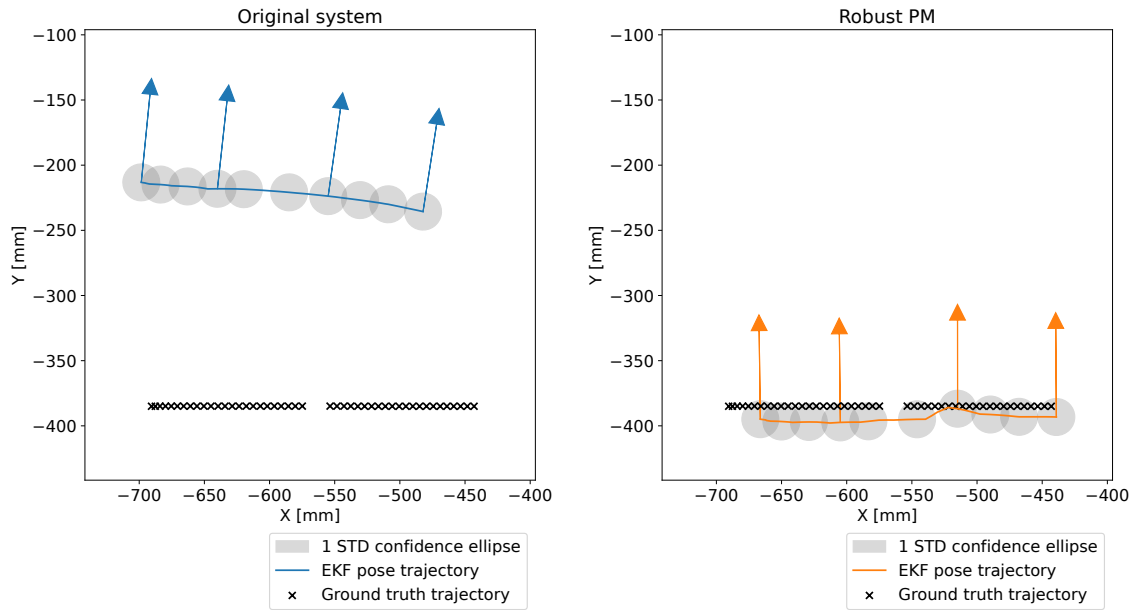


Figure 5.17: Case Study II: Trajectories plot of the ground truth, the original, and the **RoPM** systems. **RoPM** shows a superior performance compared to the original system. The periodic arrows along both systems' trajectories represent the robot's orientation. The discontinuity in the GT points was a network problem.

In Figure 5.17, the arrows are periodically plotted to show the robot's orientation during the trajectory performed. The ellipses are one **STD** confidence ellipses for bivariate data, the black x symbols show the ground truth positions, and the line trajectories show both systems **EKF**'s means interpolated by the plotting function. The arrows and the ellipses were plotted with a decimation factor. It is possible to note in the image that the original system had a much more significant bias relative to the true values, whereas **RoPM** had a more negligible bias, really close in the mm scale of the true values. A discontinuity in position data transmission from the ground truth system to the robot is visible as a gap between the x symbols. Figure 5.18 displays a trajectory comparison without clutter for better visualization.

Therefore, analyzing Figure 5.18, it is possible to perceive that the original's system performance is heavily degraded. In contrast, **RoPM**'s degradation is minimized or even reduced to insignificant magnitudes. The same error metrics analysis of Case Study I was also done for this case study. They are shown in Figures 5.19 and 5.20.

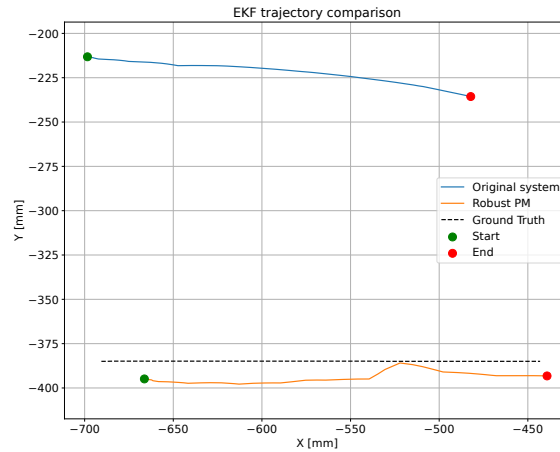


Figure 5.18: Case Study II: XY plot of the ground truth, the original, and the RoPM systems. The illustration shows the trajectory of both systems compared to the ground truth. The green and red circles indicate both systems' trajectories' start and endpoints.

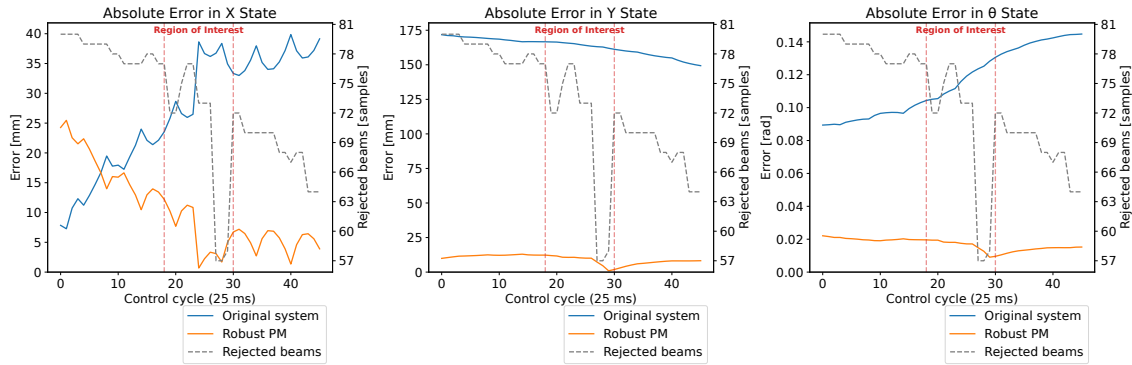


Figure 5.19: Case Study II: Robot state variables' absolute errors comparison of the original and the RoPM systems. The left y-axis represents the absolute errors for both systems in mm, and the right y-axis represents the number of rejected beams at a given control cycle (dashed gray lines). The x-axis represents the control cycles, with samples taken every 25 ms. The region of interest highlights the moment when the Machine A profile appears in the coverage of the robot's camera and LiDAR, obscuring the outliers. Consequently, the number of rejected beams plummets without causing an increase in errors.

In Figure 5.19, it is possible to see the same behavior as the first case study where the number of rejected beams is reversely correlated with the error metrics magnitudes. There is a moment during the test where the number of the rejected beams dropped; this happened when the robot's sensors coverage was obstructed by Machine A during the trajectory, displayed in Figure 4.6. Despite this occurrence, all state variables of RoPM had very low magnitudes except the x state since the robot was in a position with less localization data for the minimizer to correct this state. As soon the robot started moving closer to the middle of the map, new information to correct the x state appeared, and then the error decreased. The y and θ states had plenty of information since the beginning, obstructed by many personnel. However, the AI detected and acted accordingly to increase the performance of the localization. This is reflected by the plots of these state variables

where RoPM had minor errors. On the other hand, the original system had significant degradation in estimating the state variables where it reached unfeasible and impractical magnitudes. This effect was much more impactful in the y state because the personnel was at the warehouse entrance, and the robot was looking at them, projecting the correction of the LiDAR beams much more in the y state. The PM loss of this case study is displayed in Figure 5.20.

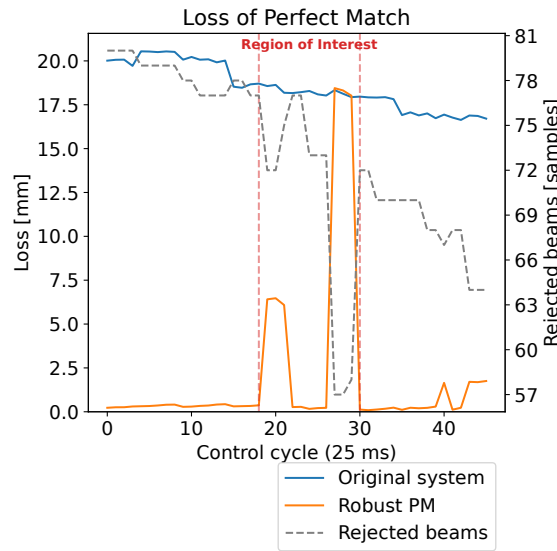


Figure 5.20: Case Study II: PM loss of the original and the RoPM systems. The left y-axis represents the PM loss for both systems in mm, and the right y-axis represents the number of rejected beams at a given control cycle (dashed gray line). x -axis represents the control cycles, with samples taken every 25 ms. The region of interest highlights the moment when the Machine A profile appears in front of the robot's camera and LiDAR, obscuring the outliers. Thus, the PM loss reacts accordingly by increasing its convergence point.

Figure 5.20 shows the same behavior as the previous case study. There are two moments where the number of rejected beams dropped, and the PM loss increased. The first one is because the YOLO model did not detect all outliers in the image, and the second instance, which is much more prominent, is the part where the robot sensors' coverage is obstructed by Machine A. The PM loss of the original system varied in the range of 17.5 – 20 mm and RoPM's stayed close to zero during the test despite the instances where the object detection model did not detect reliable all outliers. The magnitude of the PM loss of the original system decreased along the test because the robot's trajectory ended near the middle of the factory floor where the map-matching data is more abundant, diluting the outliers and decreasing the error. The quantitatively analysis is displayed in Tables 5.10, 5.11, and 5.12.

As shown in Table 5.10. For the x state, the original system had an MAE of 31.69 mm and RMSE of 32.14 mm, suggesting borderline positional accuracy, with deviations reflecting STD of 5.57 mm. The y coordinate shows a significantly higher MAE of 164.45 mm and an RMSE of 164.46 mm, indicating a substantial error and a tighter deviation with an STD of 1.82 mm. The orientation error is relatively low with an MAE and RMSE of 0.12 rad. Lastly, the PM loss

Table 5.10: Case Study II error metrics. Original system. The considered analysis range was the “Region of interest”.

	MAE	STD	MAX	MinAE	RMSE
X [mm]	31.69	5.57	38.64	23.54	32.14
Y [mm]	164.45	1.82	166.65	161.22	164.46
θ [rad]	0.12	0.01	0.13	0.10	0.12
PM [mm]	18.25	0.25	18.71	17.93	18.25

presents a lower variation in error with an **MAE** and **RMSE** of 18.25 mm and a minimal **STD** of 0.25 mm. The similar metrics of **MAE** and **RMSE** do not indicate the absence of outliers. Instead, it is because, in this scenario, the original system faced the outliers during the whole test.

Table 5.11: Case Study II error metrics. **RoPM** system. The considered analysis range was the “Region of interest”.

	MAE	STD	MAX	MinAE	RMSE
X [mm]	6.56	4.11	12.21	0.68	7.65
Y [mm]	8.70	3.83	12.29	0.94	9.45
θ [rad]	0.02	0.00	0.02	0.01	0.02
PM [mm]	5.80	7.55	18.45	0.13	9.29

On the other hand, the **RoPM** errors’ metrics are displayed in Table 5.11. All the metrics were reduced substantially, showing that the approach improves the algorithm’s overall performance even with a scenario filled with outliers. These results show that even with some limitations, the monocular camera and the default trained model, for instance, the approach showed significant improvements in all error metrics. For better overall analysis, the percentage difference of **RoPM** relative to the original system’s error metrics was also computed and shown in Table 5.12.

Table 5.12: Case Study II. Percentage difference of the **RoPM** relative to the original system.

	D_{MAE}	D_{STD}	D_{MAX}	D_{MinAE}	D_{RMSE}
X [%]	79.31	26.36	68.39	97.11	76.19
Y [%]	94.71	110.20	92.62	99.42	94.25
θ [%]	86.03	60.99	85.00	91.32	85.76
PM [%]	68.22	-	1.37	99.30	49.12

As it is possible to see, all error metrics showed remarkable percentage reduction corroborating the superiority of **RoPM** over the original system in terms of accuracy, precision, and robustness. The only metric that increased was the **STD** of the **PM** loss. However, this is expected because the **YOLO** model could not detect all outliers in all analyzed frames, and therefore, it changed the estimations over time, increasing the deviations. Still, the values are negligible, and the percentage difference analysis in this context does not add value.

5.2.2 Real-world

Following the simulation assessments, the original and the **RoPM** localization systems were subjected to real-world tests to evaluate their robustness and accuracy in dynamic environments. The transition from simulated environments, generally more controlled and predictable, to complex real-world scenarios, often presents unexpected challenges and can significantly affect performance metrics.

Baseline - Perfect Match Without Outliers

During this test, the original system was evaluated under normal conditions, i.e., without outliers on the factory floor, and it was submitted to a trajectory follow-up task. Therefore, the **EKF**-based **PM** algorithm's performance, as deployed on the factory floor and evidenced by the trajectory plot in Figure 5.21 and statistical metrics in Table 5.13, demonstrates high accuracy and reliability under ideal conditions in the real-world factory mockup. Figure 5.21 illustrates the close congruence between the estimated **EKF** pose trajectory and the ground truth, which is bounded within the one **STD** confidence ellipse, suggesting consistent positional accuracy and precision. This visual alignment is quantitatively supported by the low error values reported in the table.

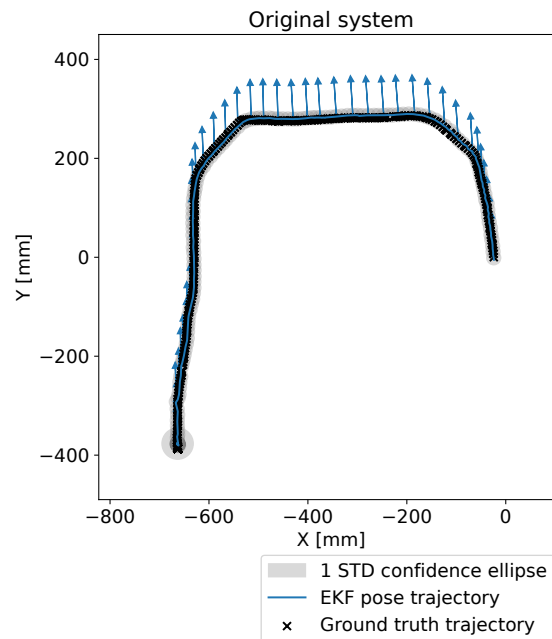


Figure 5.21: Real Scenario Baseline Study: Trajectory plot comparison of the original and the ground truth systems. The periodic blue arrows show the robot's orientation during the trajectory.

In Figure 5.22, the graphical representation of absolute errors in the x , y , and θ states provide further insight. The peaks in absolute errors align with the maximum errors reported in the table, where x , y , and θ recorded maximum values of 13.12 mm, 17.82 mm, and 0.04 rad, respectively. Despite these peaks, most data points remain within a lower error range, illustrating the algorithm's overall stability and precision. Figure 5.23 shows the **PM** loss during the test.

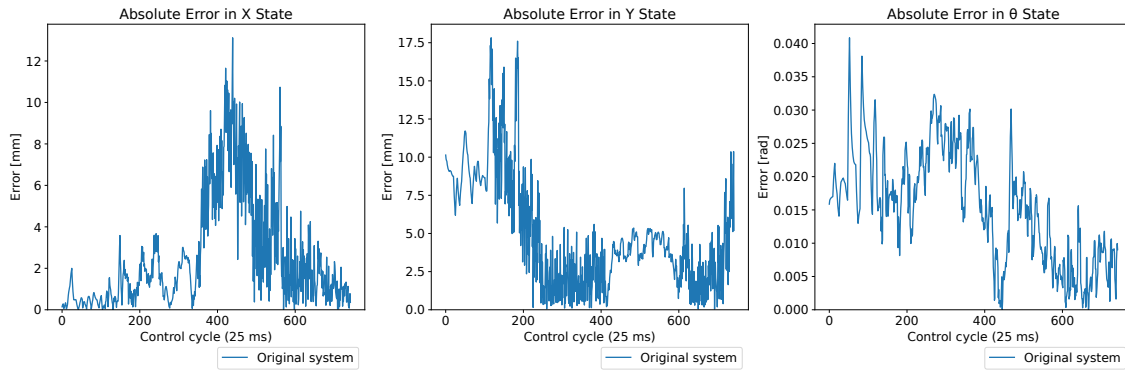


Figure 5.22: Real Scenario Baseline Study: Robot state variables' absolute errors of the original system

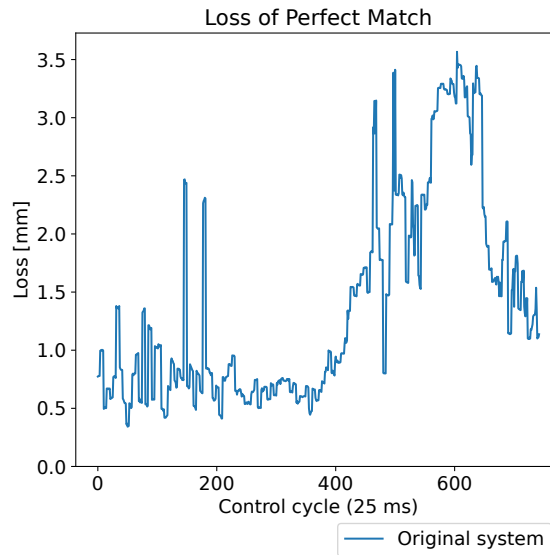


Figure 5.23: Real Scenario Baseline Study: **PM** loss of the original system

Figure 5.23 displays that the **PM** loss varied in the range of $0.3 \sim 3.5$ mm, shallow values showing that the algorithm was converging to local minima but not to a global minimum. The error magnitude increased slightly during the test's development, which is expected because as the test progressed, the robot reached positions on the factory floor with much more data to minimize (middle of the map), accumulating more errors despite being negligible. Nevertheless, it achieved remarkable accuracy and precision. Three more circumstances exist to consider regarding the **PM** loss reaching local minima. One, the first half of the test trajectory happened in the factory floor areas that were not abundant with map-matching data (the extremities of the map). In addition, the heuristic of the system for finishing an optimization cycle was to perform 20 iterations or run for 25 ms, whichever was the fastest. Third, the system achieved global minima in some instances, and these results are limited by the **LiDAR** sensor's measurement accuracy.

Table 5.13 provides a deeper statistical context to these observations by highlighting the **MAE** and **RMSE** across different metrics. The **MAE** for the x and y coordinates are remarkably low

Table 5.13: Real Scenario Baseline Study error metrics. Original system.

	MAE	STD	MAX	MinAE	RMSE
X [mm]	2.53	2.62	13.12	0.01	3.64
Y [mm]	4.89	3.60	17.82	0.05	6.07
θ [rad]	0.02	0.01	0.04	0.00	0.02
PM [mm]	1.38	0.89	3.57	0.34	1.64

at 2.53 mm and 4.89 mm, respectively, with **RMSEs** of 3.64 mm and 6.07 mm, emphasizing the algorithm’s ability to maintain tight control over localization errors under optimal conditions. Similarly, the low **MAE** of 1.38 mm and **RMSE** of 1.64 mm for the **PM** loss, as graphically depicted in Figure 5.23, validate the **PM** algorithm’s robustness and reliability, showing it was performing in the mm scale, almost converging to zero. Finally, for a better visualization without the plot clutter, Figure 5.24 shows the trajectory of the original system compared with the ground truth without the confidence ellipses and the orientation arrows.

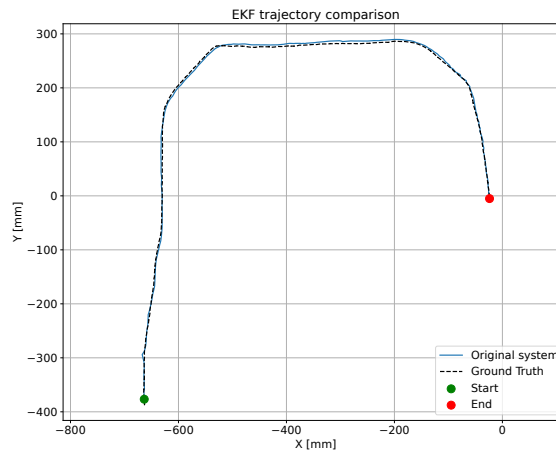


Figure 5.24: Real Scenario Baseline Study: XY plot of the ground truth and original systems. The illustration shows the trajectory of both systems compared to the ground truth. The green and red circles indicate the start and endpoints of the original system’s trajectory, respectively.

Test With Outliers in the Entrance of the Warehouse

The test conducted at the warehouse entrance with the presence of outliers provides a critical evaluation of the robustness and efficacy of the **RoPM** versus the original system. Figure 5.25 shows a trajectory comparison, highlighting significant deviations between the **EKF** pose trajectory of the original system and the ground truth, especially compared to the **RoPM**. The extended error deviations are graphically represented, indicating a broader spread and greater error magnitude in the original system than in the **RoPM**.

Quantitative data in Table 5.14 supports these observations, where the original system exhibits a **MAE** of 19.89 mm in x and 97.37 mm in y , considerably higher than the **RoPM**, which maintains a **MAE** of 5.32 mm in x and 6.50 mm in y as noted in Table 5.15. The improvements in accuracy

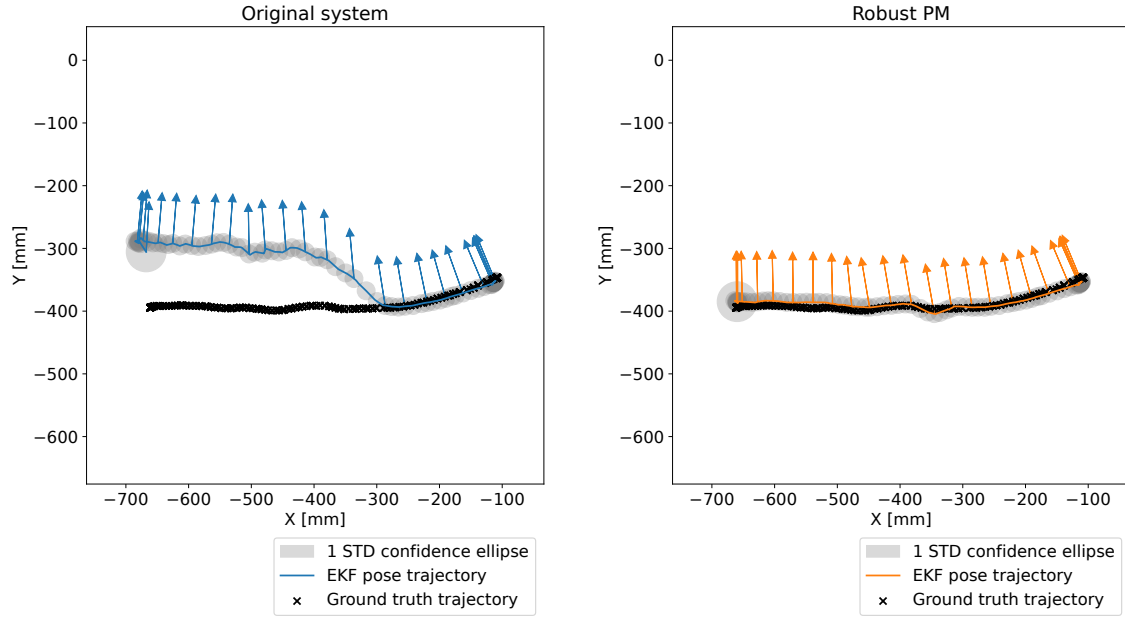


Figure 5.25: Real Scenario Case Study I: Trajectory comparison of the ground truth, the original, and the **RoPM** systems. **RoPM** shows a superior performance compared to the original system. The periodic arrows along both systems' trajectories represent the robot's orientation.

and consistency are further emphasized in the percentage differences shown in Table 5.16, with the **RoPM** showing improvements of over 70% in most metrics compared to the original system. The same reasoning can be applied to the other metrics since **RoPM** performed significantly better than the original system in all metrics.

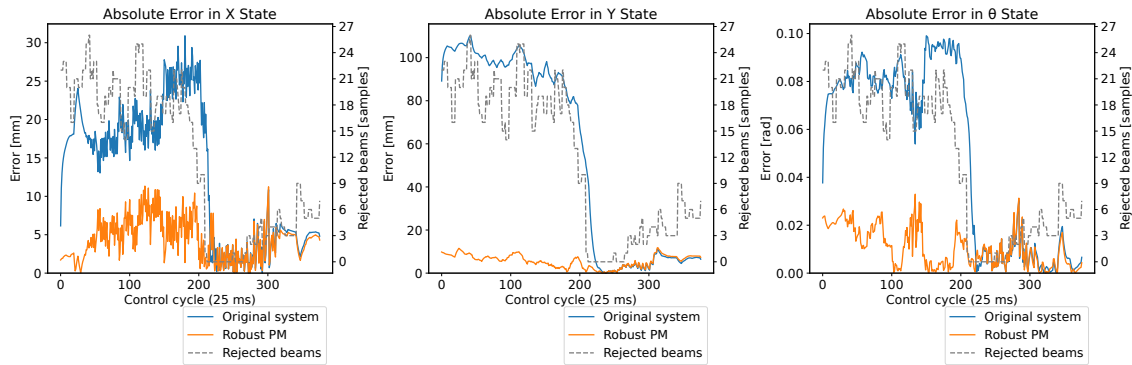


Figure 5.26: Real Scenario Case Study I: Robot state variables' absolute errors comparison of the original and the **RoPM** systems. The left y-axis represents the absolute errors for both systems in mm, and the right y-axis represents the number of rejected beams at a given control cycle (dashed gray lines). The x-axis represents the control cycles, with samples taken every 25 ms. The rejected beams and error of the original system decrease when the trajectory approaches control cycle 200 since Machine A appears in front of the outliers from the robot's sensors coverage perspective. From there, both systems behave similarly with low stationary errors.

In Figure 5.26, the absolute errors in x , y , and θ dimensions are plotted, illustrating more

stable and lower error rates in the RoPM system throughout the test, corroborated by lower STD and MaxAE values in Table 5.15. This stability is particularly evident in the PM loss, where the original system's PM loss is 11.87 mm MAE compared to 1.39 mm in the RoPM.

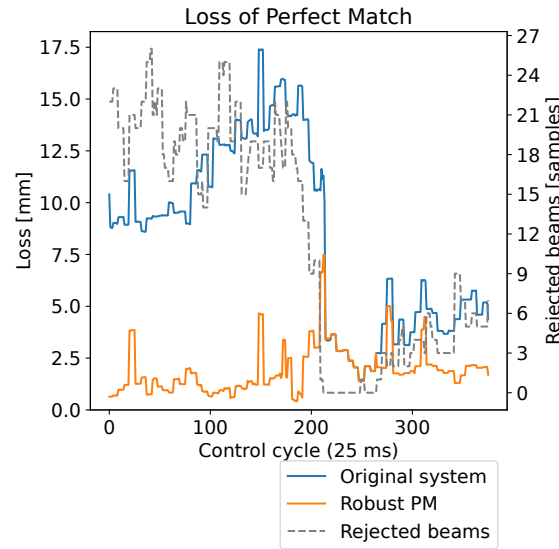


Figure 5.27: Real Scenario Case Study I: PM loss of the original and the RoPM systems. The left y-axis represents the absolute errors for both systems in mm, and the right y-axis represents the number of rejected beams at a given control cycle (dashed gray lines). The x-axis represents the control cycles, with samples taken every 25 ms. The rejected beams and loss of the original system decrease when the trajectory approaches control cycle 200 since Machine A appears in front of the outliers from the robot's sensors coverage perspective. From there, both systems behave similarly with low stationary errors.

Figure 5.27 and 5.28 visually and quantitatively underline the superior performance of the RoPM system in handling real-world disturbances. Figure 5.27 depict smaller error bands, whereas Figure 5.28 also shows tighter clustering around the ground truth, demonstrating the RoPM system's capability to deliver high accuracy and consistency even amid environmental outliers.

Table 5.14: Real Scenario Case Study I error metrics. Original system.

	MAE	STD	MAX	MinAE	RMSE
X [mm]	19.89	3.99	30.91	6.15	20.29
Y [mm]	97.37	7.33	110.15	78.69	97.64
θ [rad]	0.08	0.01	0.10	0.04	0.08
PM [mm]	11.87	2.44	17.41	8.59	12.12

Dynamic Personnel Entering in Front of the Robot

This test involved personnel unexpectedly crossing in front of the robot, providing a challenging environment to evaluate the comparative performance of the original and the RoPM localization systems. Since this test reversed the roles of the relative movement (the personnel are moving and

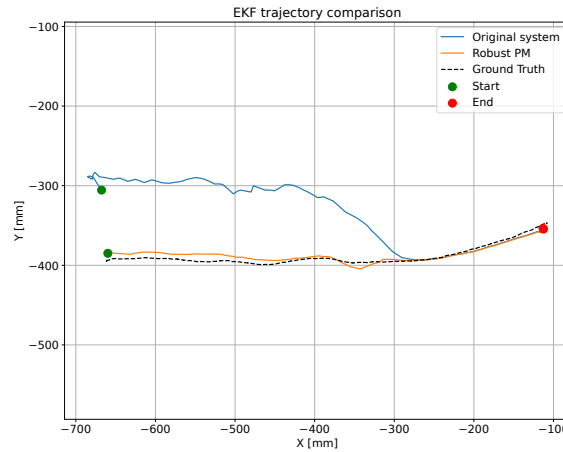


Figure 5.28: Real Scenario Case Study I. XY plot of the ground truth, the original, and the RoPM systems. The illustration shows the trajectory of both systems compared to the ground truth. The green and red circles represent both systems' start and endpoints.

Table 5.15: Real Scenario Case Study I error metrics. RoPM system.

	MAE	STD	MAX	MinAE	RMSE
X [mm]	5.32	2.68	11.34	0.04	5.95
Y [mm]	6.50	2.40	11.51	0.55	6.92
θ [rad]	0.01	0.01	0.03	0.00	0.02
PM [mm]	1.39	0.86	4.67	0.42	1.63

Table 5.16: Real Scenario Case Study I. Percentage difference of the RoPM relative to the original system.

	MAE	STD	MAX	MinAE	RMSE
X [%]	73.26	32.90	63.32	99.32	70.66
Y [%]	93.33	67.22	89.55	99.30	92.91
θ [%]	83.22	14.28	66.74	99.60	80.44
PM [%]	88.29	64.54	73.19	95.15	86.51

not the robot), it was considered that the trajectory plot with the confidence ellipses would not add value to the analysis.

Figure 5.29 and Tables 5.17 and 5.18 illustrate the absolute errors for the x , y , and θ states. The original system displays significant variability and higher error peaks, represented by **MaxAE**, particularly noticeable in the y coordinate, where errors soared up to 118.95 mm. Conversely, the RoPM system maintained a more contained error peak profile, never exceeding 24.38 mm for the y coordinate, as detailed in Table 5.18. The much lower standard deviations in the RoPM system indicate remarkable consistency and precision under dynamic conditions. Related to **MAE** and **RMSE**, RoPM also had a much more constrained and controller error profile in the millimetric scale with the y state variable with the highest magnitude of 17.19 mm, which is still a remarkable result in this highly-dynamic case study. On the other hand, the original system had these metrics

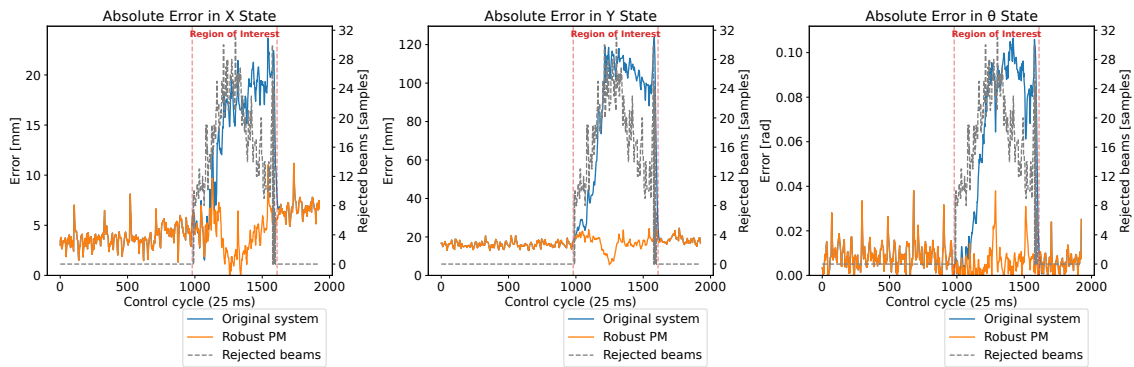


Figure 5.29: Real Scenario Case Study II: Robot state variables' absolute errors comparison of the original and the RoPM systems. The left y-axis represents the absolute errors for both systems in mm, and the right y-axis represents the number of rejected beams at a given control cycle (dashed gray lines). The x-axis represents the control cycles, with samples taken every 25 ms. The rejected beams and error of the original system increase when the trajectory approaches the region of interest when personnel appears in front of the robot's sensors coverage. During the test, RoPM maintains low asymptotic error values.

reaching up to approximately 9 cm of error in the y state variable, which, for this factory floor and working scenario, is not feasible to employ. The y state variable was the most affected because the robot faced the warehouse entrance, which has map-matching data to correct for the y state if the robot is orthogonal to it, considering the minimizer.

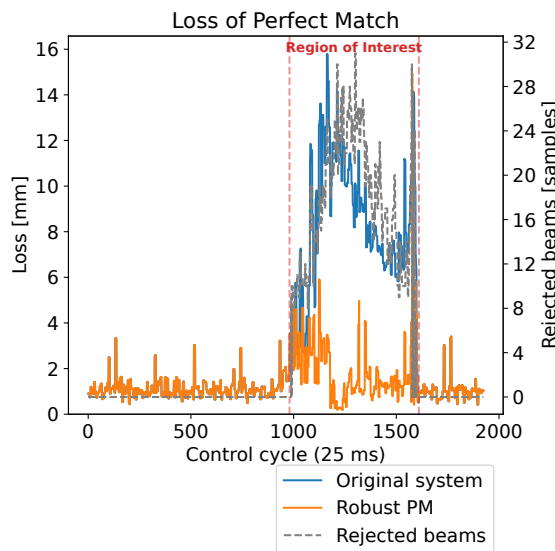


Figure 5.30: Real Scenario Case Study II. PM loss comparison of the original and the RoPM systems. The left y-axis represents the absolute errors for both systems in mm, and the right y-axis represents the number of rejected beams at a given control cycle (dashed gray lines). The x-axis represents the control cycles, with samples taken every 25 ms. The rejected beams and PM loss of the original system increase when the trajectory approaches the region of interest when personnel appears in front of the robot's sensors coverage. During the test, RoPM maintains low asymptotic loss values.

The **PM** loss shown in Figure 5.30 and quantified in Tables 5.17 and 5.18 reveal a similar trend. The original system's **PM** reached errors as high as 15.80 mm, whereas the **RoPM** system limited this to 5.92 mm. Despite the challenging scenario, the **RoPM** system maintained a lower mean error, indicating its resilience to dynamic disturbances.

Table 5.17: Real Scenario Case Study II. Original system. The considered analysis range was the “Region of interest”.

	MAE	STD	MAX	MinAE	RMSE
X [mm]	13.93	5.86	23.68	1.52	15.11
Y [mm]	83.26	33.35	118.95	19.34	89.68
θ [rad]	0.06	0.04	0.11	0.00	0.07
PM [mm]	8.38	2.92	15.80	2.10	8.88

Table 5.18: Real Scenario Case Study II. **RoPM** system. The considered analysis range was the “Region of interest”.

	MAE	STD	MAX	MinAE	RMSE
X [mm]	3.87	1.94	11.04	0.03	4.33
Y [mm]	16.70	4.08	24.38	5.56	17.19
θ [rad]	0.01	0.01	0.04	0.00	0.01
PM [mm]	1.76	1.18	5.92	0.18	2.12

Table 5.19 provides a stark quantitative comparison, showing percentage improvements in error metrics with the **RoPM** system. The error reductions are profound, with a 72.22% decrease in **MAE** for the x coordinate, 79.94% improvement in the y coordinate, 89.94% reduction in the θ variable, and a 78.95% improvement in the **PM** loss. The same reasoning can be applied to the other statistical metrics where all of them **RoPM** showed significant improvements. These statistics underscore the substantial enhancements the **RoPM** system achieves in handling dynamic obstacles, showing a more consistent, accurate, precise, and robust error profile.

Table 5.19: Real Scenario Case Study II. Percentage difference of the **RoPM** relative to the original system.

	MAE	STD	MAX	MinAE	RMSE
X [%]	72.22	66.80	53.37	98.00	71.34
Y [%]	79.94	87.77	79.50	71.28	80.83
θ [%]	89.94	81.96	64.39	95.25	87.57
PM [%]	78.95	59.62	62.54	91.61	76.09

Lastly, Figure 5.31 visually confirms the superior trajectory adherence of the **RoPM** to the ground truth compared to the original system, emphasizing the **RoPM**'s advanced capability to correct itself swiftly in response to dynamic changes, thus ensuring more accurate and reliable localization even in the presence of potential disruptions. This is highly noticeable in the trajectory comparison in the plot, where the original system diverged to a higher bias and precision as soon

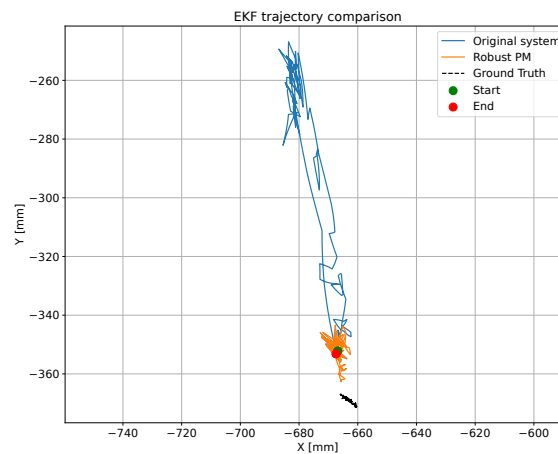


Figure 5.31: Real Scenario Case Study II. XY plot of the ground truth, the original, and the **RoPM** systems. The illustration shows the trajectory of both systems compared to the ground truth. The green and red circles represent both systems' start and endpoints. **RoPM** showed much lower bias and dispersion than the original system.

as the personnel entered the scenario. **RoPM**, on the other hand, maintained its bias and precision profile consistent and close to the true value. The endpoints are the same as the starting points because the personnel were removed from the sensors' coverage as soon as the testing finished.

Chapter 6

Conclusion and Future Work

This chapter summarizes a comprehensive study to improve localization systems for [AMR](#) in indoor dynamic [I4.0](#) environments. Through integrating object detection [AI](#), this research enhances localization robustness, accuracy, and precision, addressing gaps in current low-cost technology and setting a new benchmark for cost-effective, scalable solutions. This research introduces a novel [AI](#)-enhanced localization system with economically effective hardware that diverges from traditional static models by employing dynamic object detection, significantly enhancing adaptability and performance in unpredictable environments. In this reasoning, the remainder of this chapter will conclude the overall study performed in a structured way, going through all the achievements acquired while validating the thesis's hypothesis. Thus, this chapter will provide the final discussion and conclusion of the following topics:

- Summary of findings.
- Development and validation of the robotic platform.
- Development and validation of the ground truth system, [KMEA](#).
- Definition of the study case and testing scenarios.
- Validation of the [AI](#)-enhanced indoor localization system - [RoPM](#).

After the final discussion on the mentioned topics, the limitations and challenges faced during this research are stated in Section [6.7](#). Then, the future directions with research potential will be described.

6.1 Principal Outcomes

This research validated the hypothesis that integrating object detection [AI](#) with [AMRs](#)' localization systems enhances robustness, accuracy, and precision in dynamic [I4.0](#) warehouse environments. This enhancement was achieved by developing and implementing an [AI](#)-enhanced localization

system. The core of this system is a uniquely adapted map-matching algorithm entitled Robust Perfect Match (RoPM).

The **AI** component uses cameras to dynamically perceive the environment, effectively recognizing and evaluating outliers that impact localization. This capability allows the system to make informed decisions about the utility of sensed data, thereby significantly improving the decision-making process in real-time localization operations. Such enhancements are critical in dynamic settings where static assumptions fail and can lead to significant localization errors. The low-cost hardware of this system was possible due to the advancements in CNNs over recent years.

Comprehensive testing in both simulated and real-world scenarios demonstrated marked improvements in localization performance. **RoPM** algorithm, integrated with an **EKF** using the robot's odometry as the process model, exhibited superior performance metrics. Key improvements were noted in accuracy, precision, and robustness, with the system showing resilience against common dynamic environmental challenges such as relative moving obstacles and temporary occlusions.

Validation efforts were quantified through several statistical metrics, including but not limited to **MAE**, **RMSE**, and **STD** under varying conditions. The results underscored the system's capability to maintain high performance even in a complex industrial environment setting represented by the developed **I4.0** factory mockup. These improvements in localization performance demonstrate technical success and translate to a solid foundation and groundwork to increase operational efficiency and flexibility in dynamic warehouse environments, directly aligning with the strategic goals of **I4.0**.

6.2 Development and Validation of the Robotic Platform

A systematic approach was employed to validate the hypothesis posited in this thesis, starting with the conceptualization and development of the bespoke robotic platform. This platform was central to testing and demonstrating the practicality and effectiveness of the **RoPM** developed through this research.

Several strategic considerations drove the decision to develop a custom robotic platform:

- **Economic Feasibility:** Off-the-shelf robotic platforms, while readily available, are often expensive. By designing a custom platform, the research minimized cost and tailored the design to fit the specific requirements of the **RoPM**.
- **Full control:** A purchased platform, even if open-source, would not provide complete control over all components and processes. Developing a custom platform ensured that every aspect, from sensor selection to hardware configuration, was aligned with the research needs.
- **Educational Value:** Instead of focusing on just one part, by engaging in the design and assembly of the platform, there was an invaluable opportunity for hands-on learning and

a deep understanding of the integration between hardware components and software algorithms, essential for troubleshooting and optimizing the system. In other words, the opportunity to implement from the hardware and low-level firmware to the high-level software was deemed essential.

The developed platform was a 4-wheeled mecanum robotic platform chosen for its ability to maneuver with high agility in complex warehouse environments. From the ground up, the platform was equipped with cost-effective hardware supporting advanced localization tasks.

The mechanical framework was crafted and served as the subject of a master's thesis co-supervised by the author, ensuring the integration of academic rigor and practical design strategies.

The platform's electronics and firmware were custom-developed to manage sensors, actuators, and motor control, thus providing a robust foundation for the robotic platform and localization system.

At the software level, the system included path and trajectory planning algorithms, robust localization modules, and advanced communication protocols—all designed to optimize the performance and reliability of the localization system in dynamic environments with low-cost hardware. Except for one or two scripts, the whole robot was developed in C++, aiming for computational efficiency.

Further validation of the platform's capabilities was achieved through participation in a national I4.0 logistics warehouse competition, RAF, where the platform, equipped with a newly developed localization system tailored for the competition, secured a commendable third place by tying with the first and second places, losing just for the time of performing the tasks. Therefore, it was considered a successful test of the robotic platform, as it performed all tasks consistently and reliably, just penalized by flaws in the rules of the competition. This localization system and the developed robotic platform were published in a renowned journal and internationally indexed conference, showing its unique setting of employing fiducial markers for localization incorporated in an EKF-based approach with two filters for observations, the Euclidean and Mahalanobis filters. Further information can be found in publications [204, 214, 215].

6.3 Development and Validation of Ground Truth System

Validating the hypothesis of this thesis required a reliable ground truth system to accurately assess the performance of the RoPM system under real-world conditions. Given the limitations of simulated environments in fully replicating all possible variables in a real scenario and being unable to model non-deterministic behavior, developing an in-situ ground truth system was imperative to test in the real world.

The need for a bespoke ground truth system arose from the same practical and economic considerations that led to the development of the custom robotic platform. By employing a SOTA algorithm with fiducial markers, the ground truth system could efficiently and accurately locate these markers within a dynamic I4.0 factory setting, providing a robust basis for performance validation.

The ground truth system was designed using cost-effective, readily available hardware to ensure scalability and accessibility. The core components included:

- **Raspberry Pi 5:** Chosen for its balance of performance, affordability, and community support, serving as the computational backbone of the system.
- **Raspberry Pi Cam Module V2:** Used for its integration ease with the Raspberry Pi 5 and sufficient resolution to capture detailed fiducial markers within the environment.

The ground truth algorithm allowed the system to operate without needing the camera's extrinsic parameters, distinguishing it from other [SOTA](#) approaches. This feature significantly simplifies the setup and reduces the time required for system deployment.

The ground truth system underwent extensive testing in real-world scenarios to ensure accuracy and reliability. The performance metrics, validated against systematic testing, demonstrated that the system could achieve millimeter-scale accuracy and precision with standard calibration procedures. This level of precision is particularly noteworthy given the system's cost constraints and hardware simplicity. The robustness and accuracy of the ground truth system were further validated through its publication in [216]. After implementing the *Z Correction Algorithm*, the parallax effect from the monocular vision was nullified. Consequently, the estimation errors and deviations, which increased with distance from the image's center towards the edges, were minimized to practically zero.

[KMEA](#)'s development and successful implementation have provided a reliable means to quantify the improvements brought about by the [RoPM](#) system, ensuring that the findings are grounded in verifiable data. This ground truth system has set different branches for future research, demonstrating that high accuracy and robustness can be achieved without incurring prohibitive costs or overly complex setups. It also establishes the foundational work of a reliable and scalable multi-robot/multi-marker localization system for indoor scenarios. Further information on the contributions of the ground truth system can be found in [214, 216]

6.4 Study Case and Testing Scenario

The validation of the [RoPM](#) system was further supported by a designed study case and testing scenario, which closely mimicked an in-scale [I4.0](#) warehouse environment. This setup was developed to demonstrate the system's real-world applicability and robustness in a laboratory setting.

The testing scenario was constructed to reflect the complex nature of modern industrial warehouses, incorporating several structural and operational elements typical of [I4.0](#) settings. For this reason, a scale replica of a factory floor was assembled in the laboratory, complete with garages, an entrance and an exit, and designated areas for storing and retrieving boxes. This mockup was designed based on the dimensions and layouts used in the national robotic competition, [RAF](#), ensuring the scenario was as realistic as possible. Alongside the physical mockup, a simulated factory model was created in the robotics simulation software SimTwo to allow for testing before transitioning to the real scenario, speeding the development process. This simulated environment

included exact replicas of the physical layout, enhancing the consistency and validity of the testing results.

Obstacles were introduced to test the localization system's capability to handle real-world unpredictability. Human figures were 3D printed and strategically placed within the test environment to act as relative dynamic outliers. These figures could be moved or repositioned to simulate changing human activity in the warehouse, challenging the system's ability to adapt its localization strategies. The entire setup and each testing phase were thoroughly documented to provide a comprehensive overview of the system's performance under several conditions. The developed study case effectively bridged the gap between theoretical research and practical application, demonstrating the localization system's viability in real I4.0 warehouses by replicating a real-world industrial scenario within a simulated and physical environment. In this reasoning, further information about the testing scenario, RAF, can be found in [214–216].

6.5 System Testing and Hypothesis Validation

The RoPM and the original system underwent testing within the designed study cases and developed testbed. Testing encompassed several scenarios in simulated and real-world environments to ensure the localization system could reliably perform under diverse conditions.

The performance of the localization systems was quantified using several statistical metrics described in the thesis and widely used in the literature. Measurements of pose errors and deviations of RoPM demonstrated significant improvements in accuracy and precision over the original localization system. The RoPM system's ability to maintain performance metrics under varying conditions, including dynamic outliers and environmental changes, was recorded and analyzed. In addition to quantitative metrics, qualitative assessments were also part of the evaluation process. Visual inspection of entire test runs was generated by doing posterior offline analysis with online-recorded data. The results validate the research hypothesis and pave the way for future applications and development in autonomous mobile robotics. All the result analysis concludes that the RoPM system performed superiorly in all statistical metrics and visual assessment, maintaining a low error and high-consistent profile in all challenging tests, demonstrating reliability and robustness.

6.6 Contributions to Indoor Localization Systems

The contributions of this thesis address the prevalent gap identified in the existing literature regarding the dependency on static environment assumptions in localization systems with accessible hardware and inexpensive computational solutions. Through a comprehensive literature review, this research identified the limitations of indoor localization systems that do not operate under the static-environment assumption. They primarily rely on high-cost, high-performance hardware and sophisticated computational algorithms, rendering them impractical for widespread deployment.

This PhD research developed the **RoPM**, a novel adaptation of the map-matching algorithm **PM**. By integrating efficient, lightweight deep-learning models, the system performs real-time dynamic feature discrimination and adaptation, essential for high-precision localization without extensive computational overhead. This system's design aligns with the economic and practical demands of **I4.0**, making it a scalable solution for dynamic indoor environments. Unlike existing localization solutions, which often depend on static world assumptions and high resource demands, the **RoPM** enables effective localization in dynamic settings using standard cameras and minimalist hardware. Although it cannot compete with the computationally expensive solutions in the **SOTA**, it achieves comparable accuracy while reducing costs and resource consumption. This research addresses a pressing challenge in robotics by developing a robust, accurate, and economically feasible localization system suitable for unpredictable environments. The insights and methodologies developed here extend beyond the immediate application, offering a foundation for future innovations in autonomous mobile robotics and smart warehousing in industry and academia. Further information about the contributions can be found in [217].

6.7 Limitations and Challenges

The primary limitation of this research was network interference, as the study was conducted using the university's WiFi network, which was prone to overload and interruptions. This led to significant performance issues due to packet loss and delays. Testing the systems with all the robot's components was done at night to reduce these problems. However, in doing so, the research faced problems with illumination and personnel detection by the **YOLO** model.

Budget constraints also posed a significant challenge; despite aiming for a cost-effective solution, the affordability of robotics hardware limited the choices available, affecting the overall setup. However, this limitation proved crucial in validating the hypothesis that effective localization could be achieved with even the most budget-friendly hardware available, highlighting a key finding of this work. However, with more budget, a low-to-mid-cost embedded computer could be bought, and the object inference could run locally, avoiding the entirety of the network issues.

Additionally, the use of plastic mecanum wheels introduced mechanical reliability issues. These wheels often skidded and slid, particularly during movements where unbalanced friction conflicted with the ideal kinematic model, which affected the robot's trajectory and adherence to planned paths. Battery autonomy further restricted the extent of testing, as frequent recharging was necessary. Time constraints, exacerbated by delays due to the global COVID-19 pandemic, also impacted the research timeline and the amount of testing that could be conducted.

6.8 Future Directions

Future improvements should focus on several key areas to enhance the system's effectiveness and reliability. Developing a scientific method for estimating the covariance of the **PM** algorithm in the factory environment would provide a more physically meaningful measure of the **EKF**'s

covariance, currently defined by a sensitive analysis only for filter convergence speed, as this was not the scope of the research. Also, scientific, systematic, and analytical estimation methods for the covariance of the **EKF**'s sensors are a prevalent gap in the literature.

In addition, improving the synchronization model of the robot would also be beneficial. Data is processed as soon as it arrives, but integrating timestamps and registering them with the **EKF** could allow for faster robot operational speeds. This could start with the **LiDAR** registration; however, the current sensor does not support measurement timestamps.

Additionally, implementing a permanent, omnidirectional camera system would ensure thorough visibility, allowing the robot to detect outliers in all directions and significantly enhance the **RoPM**'s performance. Currently, a prototype was done to validate the work's hypothesis, which is done by multiplexing four low-cost cameras. However, these four cameras do not provide a 360° **FoV** for the robot, and the ESP32-CAMs are unreliable for consistent robot operation.

A custom-trained state-of-the-art object detection model specifically designed for the factory setting would also be advantageous, focusing on maximizing inference speed and accuracy while minimizing false positives and negatives. Currently, the default model from Ultralytics YOLOv8.2¹ is used.

Lastly, integrating better wheels and designing an advanced trajectory controller would address the mechanical limitations currently experienced, improving the robot's maneuverability and precision in the industrial environment, taking full advantage of the **RoPM** remarkable accuracy and precision.

¹<https://github.com/ultralytics/assets/releases>

Bibliography

- [1] IFR. Executive summary - wr service robots 2023, 2023. URL https://ifr.org/img/worldrobotics/Executive_Summary_WR_Service_Robots_2023.pdf. Accessed: May 19, 2024.
- [2] IFR. Wr service robots 2023 - sources and methods, 2023. URL https://ifr.org/img/worldrobotics/WR_Service_Robots_2023_Sources_and_Methods.pdf. Accessed: May 19, 2024.
- [3] Agnieszka A. Tubis and Juni Rohman. Intelligent Warehouse in Industry 4.0—Systematic Literature Review. *Sensors*, 23(8):4105, January 2023. ISSN 1424-8220. doi: 10.3390/s23084105. URL <https://www.mdpi.com/1424-8220/23/8/4105>. Number: 8 Publisher: Multidisciplinary Digital Publishing Institute.
- [4] Maarten van Geest, Bedir Tekinerdogan, and Cagatay Catal. Design of a reference architecture for developing smart warehouses in industry 4.0. *Computers in Industry*, 124: 103343, January 2021. ISSN 0166-3615. doi: 10.1016/j.compind.2020.103343. URL <https://www.sciencedirect.com/science/article/pii/S0166361520305777>.
- [5] Marco Lombardi, Francesco Pascale, and Domenico Santaniello. Internet of Things: A General Overview between Architectures, Protocols and Applications. *Information*, 12(2): 87, February 2021. ISSN 2078-2489. doi: 10.3390/info12020087. URL <https://www.mdpi.com/2078-2489/12/2/87>. Number: 2 Publisher: Multidisciplinary Digital Publishing Institute.
- [6] Enoch Oluwademilade Sodiya, Uchenna Joseph Umoga, Olukunle Oladipupo Amoo, Akoh Atadoga, Enoch Oluwademilade Sodiya, Uchenna Joseph Umoga, Olukunle Oladipupo Amoo, and Akoh Atadoga. AI-driven warehouse automation: A comprehensive review of systems. *GSC Advanced Research and Reviews*, 18(2):272–282, 2024. ISSN 2582-4597, 2582-4597. doi: 10.30574/gscarr.2024.18.2.0063. URL <https://gsconlinepress.com/journals/gscarr/content/ai-driven-warehouse-automation-comprehensive-review-systems>. Last Modified: 2024-02-19T16:26+05:30 Number: 2 Publisher: GSC Advanced Research and Reviews.
- [7] Jonas Felipe Pereira de Queiroz. *An Agent-based Approach to Design Decentralized Cloud-Edge Data Analysis in Cyber-Physical System*. Ph.d. dissertation, University of Porto, Porto, Portugal, 2022. URL <https://hdl.handle.net/10216/141507>.
- [8] Giuseppe Fragapane, Dmitry Ivanov, Mirco Peron, Fabio Sgarbossa, and Jan Ola Strandhagen. Increasing flexibility and productivity in Industry 4.0 production networks with autonomous mobile robots and smart intralogistics. *Annals of Operations Research*, 308 (1):125–143, January 2022. ISSN 1572-9338. doi: 10.1007/s10479-020-03526-7. URL <https://doi.org/10.1007/s10479-020-03526-7>.

- [9] Mary B Alatise and Gerhard P Hancke. A review on challenges of autonomous mobile robot and sensor fusion methods. *IEEE Access*, 8:39830–39846, 2020.
- [10] H Durrant Whyte. Simultaneous localisation and mapping (slam): Part i the essential algorithms. *Robotics and Automation Magazine*, 2006.
- [11] Mina Henein, Jun Zhang, Robert Mahony, and Viorela Ila. Dynamic slam: The need for speed. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2123–2129. IEEE, 2020.
- [12] Linhui Xiao, Jing Wang, Xiaosong Qiu, Zheng Rong, and Xudong Zou. Dynamic-slam: Semantic monocular visual localization and mapping based on deep learning in dynamic environment. *Robotics and Autonomous Systems*, 117:1–16, 2019.
- [13] Ajantha Vijayakumar and Subramaniaswamy Vairavasundaram. Yolo-based object detection models: A review and its applications. *Multimedia Tools and Applications*, pages 1–40, 2024.
- [14] Muhamad Risqi U Saputra, Andrew Markham, and Niki Trigoni. Visual slam and structure from motion in dynamic environments: A survey. *ACM Computing Surveys (CSUR)*, 51(2): 1–36, 2018.
- [15] Qi Zhang and Changdi Li. Semantic slam for mobile robots in dynamic environments based on visual camera sensors. *Measurement Science and Technology*, 34(8):085202, 2023.
- [16] Pedro Azevedo and Vítor Santos. Comparative analysis of multiple yolo-based target detectors and trackers for adas in edge devices. *Robotics and Autonomous Systems*, 171: 104558, 2024. ISSN 0921-8890. doi: <https://doi.org/10.1016/j.robot.2023.104558>. URL <https://www.sciencedirect.com/science/article/pii/S0921889023001975>.
- [17] Nishtha Prakash, Areeba Atiq, Mohammad Shahid, Jyoti Rani, and Srishti Dikshit. Merging minds and machines: The role of advancing ai in robotics. *EAI Endorsed Transactions on Internet of Things*, 10, 2024.
- [18] Bharat Mahaur, Navjot Singh, and KK Mishra. Road object detection: a comparative study of deep learning-based algorithms. *Multimedia Tools and Applications*, 81(10):14247–14282, 2022.
- [19] Ken Peffers, Tuure Tuunanen, Marcus A Rothenberger, and Samir Chatterjee. A design science research methodology for information systems research. *Journal of management information systems*, 24(3):45–77, 2007.
- [20] Md AK Niloy, Anika Shama, Ripon K Chakraborty, Michael J Ryan, Faisal R Badal, Zinat Tasneem, Md H Ahamed, Sumaya I Moyeen, Sajal K Das, Md F Ali, et al. Critical design and control issues of indoor autonomous mobile robots: A review. *IEEE Access*, 9:35338–35370, 2021.
- [21] Sushil Kumar Sahoo and Bibhuti Bhusan Choudhury. A review of methodologies for path planning and optimization of mobile robots. *Journal of process management and new technologies*, 11(1-2):122–140, 2023.
- [22] Lixing Liu, Xu Wang, Xin Yang, Hongjie Liu, Jianping Li, and Pengfei Wang. Path planning techniques for mobile robots: Review and prospect. *Expert Systems with Applications*, page 120254, 2023.

- [23] Hejia Gao, Dongliang Liu, and Juqi Hu. A survey on path planning for mobile robot systems. In *2023 IEEE 12th Data Driven Control and Learning Systems Conference (DDCLS)*, pages 1176–1181. IEEE, 2023.
- [24] Nour Abujabal, Raouf Fareh, Saif Sinan, Mohammed Baziyad, and Maamar Bettayeb. A comprehensive review of the latest path planning developments for multi-robot formation systems. *Robotica*, pages 1–26, 2023.
- [25] Antonio Savio Silva Oliveira, Marcello Carvalho dos Reis, Francisco Alan Xavier da Mota, Maria Elisa Marciano Martinez, and Auzuir Ripardo Alexandria. New trends on computer vision applied to mobile robot localization. *Internet of Things and Cyber-Physical Systems*, 2:63–69, 2022.
- [26] Andrea Motroni, Alice Buffi, and Paolo Nepa. Mobile-robots indoor tracking and navigation: perspectives for rfid technology. In *2022 7th International Conference on Smart and Sustainable Technologies (SpliTech)*, pages 1–6. IEEE, 2022.
- [27] Prabin Kumar Panigrahi and Sukant Kishoro Bisoy. Localization strategies for autonomous mobile robots: A review. *Journal of King Saud University-Computer and Information Sciences*, 34(8):6019–6039, 2022.
- [28] Yi Kiat Tee and Yi Chiew Han. Lidar-based 2d slam for mobile robot in an indoor environment: A review. In *2021 International Conference on Green Energy, Computing and Sustainable Technology (GECOST)*, pages 1–7. IEEE, 2021.
- [29] Michail Kalaitzakis, Brennan Cain, Sabrina Carroll, Anand Ambrosi, Camden Whitehead, and Nikolaos Vitzilaos. Fiducial markers for pose estimation. *Journal of Intelligent & Robotic Systems*, 101(4):1–26, 2021.
- [30] Ricardo B Sousa, Héber M Sobreira, and António Paulo Moreira. A systematic literature review on long-term localization and mapping for mobile robots. *Journal of Field Robotics*, 40(5):1245–1322, 2023.
- [31] Aishwarya A Panchpor, Sam Shue, and James M Conrad. A survey of methods for mobile robot localization and mapping in dynamic indoor environments. In *2018 Conference on Signal Processing And Communication Engineering Systems (SPACES)*, pages 138–144. IEEE, 2018.
- [32] Francisco Rubio, Francisco Valero, and Carlos Llopis-Albert. A review of mobile robots: Concepts, methods, theoretical framework, and applications. *International Journal of Advanced Robotic Systems*, 16(2):1729881419839596, 2019.
- [33] Noor Abdul Khaleq Zghair and Ahmed S Al-Araji. A one decade survey of autonomous mobile robot systems. *International Journal of Electrical and Computer Engineering*, 11(6):4891, 2021.
- [34] Anca Morar, Alin Moldoveanu, Irina Mocanu, Florica Moldoveanu, Ion Emilian Radoi, Victor Asavei, Alexandru Gradinaru, and Alex Butean. A comprehensive survey of indoor localization methods based on computer vision. *Sensors*, 20(9):2641, 2020.
- [35] Saeid Nahavandi, Roohallah Alizadehsani, Darius Nahavandi, Shady Mohamed, Navid Mohajer, Mohammad Rokonuzzaman, and Ibrahim Hossain. A comprehensive review on autonomous navigation. *arXiv preprint arXiv:2212.12808*, 2022.

- [36] Nian Ding, Chunrong Peng, Min Lin, and Celimuge Wu. A comprehensive review on automatic mobile robots: applications, perception, communication and control. *Journal of Circuits, Systems and Computers*, 31(08):2250153, 2022.
- [37] Jacob Fraden. *Handbook of Modern Sensors*. Springer, Cham, 2016. ISBN 978-3-319-19302-1. doi: <https://doi.org/10.1007/978-3-319-19303-8>.
- [38] Roland Siegwart, Illah Reza Nourbakhsh, and Davide Scaramuzza. *Introduction to autonomous mobile robots*. MIT press, 2011.
- [39] G Balamurugan, J Valarmathi, and V P S Naidu. Survey on uav navigation in gps denied environments. In *2016 International Conference on Signal Processing, Communication, Power and Embedded System (SCOPEs)*, pages 198–204, 2016. doi: 10.1109/SCOPEs.2016.7955787.
- [40] Yongjun Liu, Rui Chen, Yan Zhou, Mengjie Liu, Yilong Hui, and Nan Cheng. Rfid-based vehicle localization using virtual wideband multi-frequency continuous wave. *IEEE Journal of Radio Frequency Identification*, 7:222–232, 2023. doi: 10.1109/JRFID.2023.3287327.
- [41] Yihe Chen, Osama Abdul Hafez, Boris Pervan, and Matthew Spenko. Landmark augmentation for mobile robot localization safety. *IEEE Robotics and Automation Letters*, 6(1): 119–126, 2020.
- [42] Mengyuan Duan, Xiaokang Lei, Zhongxing Duan, and Zhicheng Zheng. A minimalist self-localization approach for swarm robots based on active beacon in indoor environments. *Sensors*, 23(10):4926, 2023.
- [43] Muhamad Risqi U Saputra, Chris Xiaoxuan Lu, Pedro Porto B de Gusmao, Bing Wang, Andrew Markham, and Niki Trigoni. Graph-based thermal–inertial slam with probabilistic neural networks. *IEEE Transactions on Robotics*, 38(3):1875–1893, 2021.
- [44] Zixian Zang, Haotian Shen, Lizhi Yang, and Avidesh Zakhori. Sensor-aware frontier exploration and mapping with application to thermal mapping of building interiors. *Electronic Imaging*, 34:1–5, 2022.
- [45] Malte Riechmann, Matthias König, and Jan Rexilius. 3d-multi-layer-multi-representation-maps for short-and long-term mapping and navigation. In *2022 27th International Conference on Automation and Computing (ICAC)*, pages 1–6. IEEE, 2022.
- [46] Kai-Tai Song and Li-Ren Kang. Object-oriented navigation with a multi-layer semantic map. In *2022 13th Asian Control Conference (ASCC)*, pages 1386–1391. IEEE, 2022.
- [47] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: part I. *IEEE robotics & automation magazine*, 13(2):99–110, 2006.
- [48] Tim Bailey and Hugh Durrant-Whyte. Simultaneous localization and mapping (SLAM): Part II. *IEEE Robotics & Automation Magazine*, 13(3):108–117, 2006.
- [49] Hamzah Ahmad and Toru Namerikawa. Extended Kalman filter-based mobile robot localization with intermittent measurements. *Systems Science & Control Engineering: An Open Access Journal*, 1(1):113–126, 2013.

- [50] Kaiyu Wang, Guihua Xia, Qidan Zhu, Yongtao Yu, Yebin Wu, and Yan Wang. The SLAM algorithm of mobile robot with omnidirectional vision based on EKF. In *2012 IEEE International Conference on Information and Automation*, pages 13–18. IEEE, 2012.
- [51] Chris Burbridge, Libor Spacek, Joan Condell, and Ulrich Nehmzow. Monocular Omnidirectional Vision Based Robot Localisation and Mapping. *TAROS 2008—Towards Autonomous Robotic Systems 2008, The University of Edinburgh, United Kingdom*, pages 135–141, 2008.
- [52] Sunhyo Kim and Se-Young Oh. SLAM in indoor environments using omni-directional vertical and horizontal line features. *Journal of Intelligent and Robotic Systems*, 51(1): 31–43, 2008.
- [53] Hayrettin Erturk, Gurkan Tuna, Tarik Veli Mumcu, and Kayhan Gulez. A Performance Analysis of Omnidirectional Vision Based Simultaneous Localization and Mapping. In *International Conference on Intelligent Computing*, pages 407–414. Springer, 2012.
- [54] Luigi D’Alfonso, Andrea Griffo, Pietro Muraca, and Paolo Pugliese. A SLAM algorithm for indoor mobile robot localization using an Extended Kalman Filter and a segment based environment mapping. In *2013 16th International Conference on Advanced Robotics (ICAR)*, pages 1–6. IEEE, 2013.
- [55] Javier Civera, Andrew J Davison, and JM Martinez Montiel. Inverse depth parametrization for monocular SLAM. *IEEE Transactions on Robotics*, 24(5):932–945, 2008.
- [56] Fernando A Auat Cheein, Juan M Toibero, Fernando di Sciascio, Ricardo Carelli, and F Lobo Pereira. Monte Carlo uncertainty maps-based for mobile robot autonomous SLAM navigation. In *2010 IEEE International Conference on Industrial Technology*, pages 1433–1438. IEEE, 2010.
- [57] Alejandro Rituerto, Luis Puig, and J Guerrero. Comparison of omnidirectional and conventional monocular systems for visual slam. *10th OMNIVIS with RSS*, 2010.
- [58] Zolghadr Javad, Yuanli Cai, and Yekkehfallah Majid. Comparing EKF and SPKF Algorithms for Simultaneous Localization and Mapping (SLAM). *J. Robot. Netw. Artif. Life*, 3(4):217–220, 2017.
- [59] David Valiente, Arturo Gil, Lorenzo Fernández, and Óscar Reinoso. A comparison of EKF and SGD applied to a view-based SLAM approach with omnidirectional images. *Robotics and Autonomous Systems*, 62(2):108–119, 2014.
- [60] Simon J Julier and Jeffrey K Uhlmann. A counter example to the theory of simultaneous localization and map building. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*, volume 4, pages 4238–4243. IEEE, 2001.
- [61] Shoudong Huang and Gamini Dissanayake. Convergence and consistency analysis for extended Kalman filter based SLAM. *IEEE Transactions on Robotics*, 23(5):1036–1049, 2007.
- [62] Juan Andrade-Cetto, Teresa Vidal-Calleja, and Alberto Sanfeliu. Unscented transformation of vehicle states in SLAM. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 323–328. IEEE, 2005.

- [63] Ruben Martinez-Cantin and José A Castellanos. Unscented SLAM for large-scale outdoor environments. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3427–3432. IEEE, 2005.
- [64] Won-Seok Choi, Jeong-Gwan Kang, and Se-Young Oh. Measurement noise estimator assisted extended Kalman filter for SLAM problem. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2077–2082. IEEE, 2009.
- [65] Mohammad Bozorg, Masoud S Bahraini, and Ahmad B Rad. A new adaptive ukf algorithm to improve the accuracy of slam. *International Journal of Robotics, Theory and Applications*, 5(1):35–46, 2019.
- [66] Feng Yang, Mengting Yan, Bo Jin, and Litao Zheng. Slam based on double layer unscented kalman filter. In *2019 5th International Conference on Control, Automation and Robotics (ICCAR)*, pages 663–668, 2019. doi: 10.1109/ICCAR.2019.8813755.
- [67] Chao Tang, Dajian Zhou, Lihua Dou, and Chaoyang Jiang. A 3d range-only slam algorithm based on improved derivative ukf. *Electronics*, 11(7):1109, 2022.
- [68] D. Hahnel, W. Burgard, D. Fox, and S. Thrun. An efficient fastSLAM algorithm for generating maps of large-scale cyclic environments from raw laser range measurements. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, volume 1, pages 206–211 vol.1, 2003. doi: 10.1109/IROS.2003.1250629.
- [69] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters. *IEEE Transactions on Robotics*, 23(1): 34–46, 2007.
- [70] Dirk Hahnel, Wolfram Burgard, Dieter Fox, and Sebastian Thrun. An efficient FastSLAM algorithm for generating maps of large-scale cyclic environments from raw laser range measurements. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*, volume 1, pages 206–211. IEEE, 2003.
- [71] Giorgio Grisetti, Gian Diego Tipaldi, Cyrill Stachniss, Wolfram Burgard, and Daniele Nardi. Fast and accurate SLAM with Rao-Blackwellized particle filters. *Robotics and Autonomous Systems*, 55(1):30–38, 2007.
- [72] Guillem Vallicrosa and Pere Ridao. H-slam: Rao-blackwellized particle filter slam using hilbert maps. *Sensors*, 18(5):1386, 2018.
- [73] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. Nonlinear Constraint Network Optimization for Efficient Map Learning. *IEEE Transactions on Intelligent Transportation Systems*, 10(3):428–439, 2009.
- [74] Edwin Olson, John Leonard, and Seth Teller. Fast Iterative Alignment of Pose Graphs with Poor Initial Estimates. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 2262–2269. IEEE, 2006.
- [75] Michael Kaess, Hordur Johannsson, Richard Roberts, Viorela Ila, John J Leonard, and Frank Dellaert. iSAM2: Incremental smoothing and mapping using the Bayes tree. *The International Journal of Robotics Research*, 31(2):216–235, 2012.

- [76] Rainer Kümmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. g2o: A General Framework for Graph Optimization. In *2011 IEEE International Conference on Robotics and Automation*, pages 3607–3613. IEEE, 2011.
- [77] Luca Carlone and Frank Dellaert. Duality-based verification techniques for 2D SLAM. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4589–4596. IEEE, 2015.
- [78] Luca Carlone, Giuseppe C Calafiore, Carlo Tommolillo, and Frank Dellaert. Planar Pose Graph Optimization: Duality, Optimal solutions, and Verification. *IEEE Transactions on Robotics*, 32(3):545–565, 2016.
- [79] Luca Carlone and Giuseppe C Calafiore. Convex Relaxations for Pose Graph Optimization With Outliers. *IEEE Robotics and Automation Letters*, 3(2):1160–1167, 2018.
- [80] Vadim Indelman, Erik Nelson, Nathan Michael, and Frank Dellaert. Multi-Robot Pose Graph Localization and Data Association from Unknown Initial Relative Poses via Expectation Maximization. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 593–600. IEEE, 2014.
- [81] Jorge Fuentes-Pacheco, José Ruiz-Ascencio, and Juan Manuel Rendón-Mancha. Visual simultaneous localization and mapping: a survey. *Artificial Intelligence Review*, 43(1): 55–81, 2015.
- [82] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. ORB: An efficient alternative to SIFT or SURF. In *2011 International Conference on Computer Vision*, pages 2564–2571. Ieee, 2011.
- [83] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European Conference on Computer vision*, pages 404–417. Springer, 2006.
- [84] David G Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.
- [85] Liang Hong, Zhou Yangfan, Li Lihua, He Xiaofeng, Chen Junxi, and Chen Lingna. Monocular SLAM Based on Semi-Direct method. In *Proceedings of the 2020 3rd International Conference on Robot Systems and Applications*, pages 21–24, 2020.
- [86] Lauer, Martin and Lange, Sascha and Riedmiller, Martin. Calculating the Perfect Match: An Efficient and Accurate Approach for Robot Self-localization. In Bredenfeld, Ansgar and Jacoff, Adam and Noda, Itsuki and Takahashi, Yasutake, editor, *RoboCup 2005: Robot Soccer World Cup IX*, pages 142–153, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. ISBN 978-3-540-35438-3.
- [87] Sobreira, Héber and Rocha, Luis and Costa, Carlos and Lima, Jose and Costa, Paulo and Moreira, A. Paulo. 2D Cloud Template Matching - A Comparison between Iterative Closest Point and Perfect Match. In *2016 International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pages 53–59, 2016. doi: {10.1109/ICARSC.2016.13}.
- [88] Sobreira, Héber and Pinto, Miguel and Moreira, António Paulo and Costa, Paulo Gomes and Lima, José. Robust Robot Localization Based on the Perfect Match Algorithm. In Moreira, António Paulo and Matos, Aníbal and Veiga, Germano, editor, *CONTROLO’2014 – Proceedings of the 11th Portuguese Conference on Automatic Control*, pages 607–616, Cham, 2015. Springer International Publishing. ISBN 978-3-319-10380-8.

- [89] Andry M Pinto, António P Moreira, and Paulo G Costa. A localization method based on map-matching and particle swarm optimization. *Journal of Intelligent & Robotic Systems*, 77:313–326, 2015.
- [90] Martin Lauer, Sascha Lange, and Martin Riedmiller. Calculating the Perfect Match: an Efficient and Accurate Approach for Robot Self-Localization. In *Robot Soccer World Cup*, pages 142–153. Springer, 2005.
- [91] Héber Sobreira, Carlos M Costa, Ivo Sousa, Luis Rocha, José Lima, PCMA Farias, Paulo Costa, and A Paulo Moreira. Map-matching algorithms for robot self-localization: a comparison between perfect match, iterative closest point and normal distributions transform. *Journal of Intelligent & Robotic Systems*, 93(3):533–546, 2019.
- [92] Miguel Pinto, António Paulo Moreira, Aníbal Matos, Héber Miguel Sobreira, and Filipe Neves Santos. Fast 3D map matching localisation algorithm. *Journal of Automation and Control Engineering*, 2013.
- [93] Manuel Gouveia, António Paulo Moreira, Paulo Costa, Luís Paulo Reis, and Marcos Ferreira. Robustness and Precision Analysis in Map-Matching based Mobile Robot Self-Localization. In *EPIA, 14th Portuguese Conference on Artificial Intelligent*, pages 243–253, 2009.
- [94] Andry M Pinto, António P Moreira, and Paulo G Costa. A Localization Method Based on Map-Matching and Particle swarm Optimization. *Journal of Intelligent & Robotic Systems*, 77(2):313–326, 2015.
- [95] Peter Biber and Wolfgang Straßer. The normal distributions transform: A new approach to laser scan matching. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*, volume 3, pages 2743–2748. IEEE, 2003.
- [96] Martin Magnusson, Achim Lilienthal, and Tom Duckett. Scan registration for autonomous mining vehicles using 3D-NDT. *Journal of Field Robotics*, 24(10):803–827, 2007.
- [97] Martin Magnusson. *The Three-Dimensional Normal-Distributions Transform: an Efficient Representation for Registration, Surface Analysis, and Loop Detection*. PhD thesis, Örebro universitet, 2009.
- [98] Ramazan Havangi. Mobile robot localization based on PSO estimator. *Asian Journal of Control*, 21(4):2167–2178, 2019.
- [99] Ramazan Havangi, Mohammad Ali Nekoui, and Mohammad Teshnehlab. A Multi Swarm Particle Filter for Mobile Robot Localization. *International Journal of Computer Science*, 7(3):15–22, 2010.
- [100] Ali R Vahdat, Naser NourAshrafoddin, and Saeed Shiry Ghidary. Mobile robot global localization using differential evolution and particle swarm optimization. In *2007 IEEE Congress on Evolutionary Computation*, pages 1527–1534. IEEE, 2007.
- [101] Junfei Huo, Liling Ma, Yuanlong Yu, and Junzheng Wang. Hybrid algorithm based mobile robot localization using DE and PSO. In *Proceedings of the 32nd Chinese Control Conference*, pages 5955–5959. IEEE, 2013.

- [102] Hossein Vahid Dastjerdi, Mohammad Bagher Menhaj, and Saeideh Shataei. Self-localization of humanoid robots using particle swarm optimization algorithm. In *2016 Artificial Intelligence and Robotics (IRANOPEN)*, pages 98–102. IEEE, 2016.
- [103] Sabita Mali. Mobile Robot Localization Using Multi-Objective Optimization. *International Journal of Latest Technology in Engineering, Management & Applied Science-IJLTEMAS*, 4:21–25, 2015.
- [104] Haijiang Wang, Wenhong Liu, Fugui Zhang, Simon X Yang, and Lin Zhang. A GA-fuzzy logic based extended Kalman filter for mobile robot localization. In *2015 12th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, pages 319–323. IEEE, 2015.
- [105] Shreedhar D Rajurkar, Aniket K Kar, Subhajit Goswami, and Nishchal K Verma. Optimal path estimation and tracking for an automated vehicle using GA optimized fuzzy controller. In *2016 11th International Conference on Industrial and Information Systems (ICIIS)*, pages 365–370. IEEE, 2016.
- [106] Luis Moreno, Jose M Armingol, Santiago Garrido, Arturo De La Escalera, and Miguel A Salichs. A Genetic Algorithm for Mobile Robot Localization Using Ultrasonic Sensors. *Journal of Intelligent and Robotic Systems*, 34(2):135–154, 2002.
- [107] Fernando Martín, Luis Moreno, Santiago Garrido, and Dolores Blanco. Kullback-Leibler divergence-based differential evolution Markov Chain filter for global localization of mobile robots. *Sensors*, 15(9):23431–23458, 2015.
- [108] Michał Lisowski, Zhun Fan, and Ole Ravn. Differential evolution to enhance localization of mobile robots. In *2011 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2011)*, pages 241–247. IEEE, 2011.
- [109] Amitava Chatterjee. Differential evolution tuned fuzzy supervisor adapted extended Kalman filtering for SLAM problems in mobile robots. *Robotica*, 27(3):411–423, 2009.
- [110] Rafael Marques Claro, Diogo Brandão Silva, and Andry Maykol Pinto. Artuga: A novel multimodal fiducial marker for aerial robotics. *Robotics and Autonomous Systems*, 163: 104398, 2023.
- [111] Pablo García-Ruiz, Rafael Muñoz-Salinas, Rafael Medina-Carnicer, and Manuel J Marín-Jiménez. Object localization with multiplanar fiducial markers: Accurate pose estimation. In *Iberian Conference on Pattern Recognition and Image Analysis*, pages 454–465. Springer, 2023.
- [112] Francesca Digiacomo, Francesco Bologna, Francesco Inglese, Cesare Stefanini, and Mario Milazzo. Mechatag: A mechanical fiducial marker and the detection algorithm. *Journal of Intelligent and Robotic Systems: Theory and Applications*, 103(3), 2021. doi: 10.1007/s10846-021-01507-x. Cited by: 2; All Open Access, Green Open Access, Hybrid Gold Open Access.
- [113] Mi Zhang, Songshan Han, Xing Liu, Shihan Wang, Liang Han, and Jiayang Zhao. Visual-marker-inertial fusion localization system using sliding window optimization. pages 3566–3572, Kunming, China, 2021. IEEE. ISBN 978-1-6654-3129-3. doi: 10.1109/CCDC52312.2021.9602344.

- [114] Goran Popović, Igor Cviić, Ivan Marković, and Ivan Petrović. Human localization in warehouse environments based on a wearable camera sensor suite and dynamic ultra-wide band nodes. pages 818–823, Ljubljana, Slovenia, 2021. IEEE. ISBN 978-1-6654-3685-4. doi: 10.1109/ICAR53236.2021.9659480.
- [115] Ali Alabbas, Miguel Altamirano Cabrera, Oussama Alyounes, and Dzmitry Tsetserukou. Arucoglide: a novel wearable robot for position tracking and haptic feedback to increase safety during human-robot interaction. pages 1–8, Sinaia, Romania, 2023. IEEE. ISBN 979-8-3503-3992-5. doi: 10.1109/ETFA54631.2023.10275727.
- [116] Eric Sean Kesuma, Pranoto Hidayat Rusmin, and Devira Anggi Maharani. Pallet detection and distance estimation with yolo and fiducial marker algorithm in industrial forklift robot. In *2023 International Conference on Artificial Intelligence in Information and Communication (ICAIIIC)*, pages 270–275. IEEE, 2023.
- [117] Joao Braun, Alexandre O. Junior, Guido Berger, Vitor H. Pinto, Ines N. Soares, Ana Pereira, I, Jose Lima, and Paulo Costa. A robot localization proposal for the robotatfactory 4.0: A novel robotics competition within the industry 4.0 concept. *FRONTIERS IN ROBOTICS AND AI*, 9, NOV 15 2022. ISSN 2296-9144. doi: 10.3389/frobt.2022.1023590.
- [118] C. Romero, C. Naufal, J. Meza, and A.G. Marrugo. A validation strategy for a target-based vision tracking system with an industrial robot. volume 1547, 2020. doi: 10.1088/1742-6596/1547/1/012018. Cited by: 6; All Open Access, Gold Open Access.
- [119] Wei Zhang, Liang Gong, Yefeng Sun, Bishu Gao, Chenrui Yu, and Chengliang Liu. Precise visual positioning of agricultural mobile robots with a fiducial marker reprojection approach. *Measurement Science and Technology*, 34(11), 2023. doi: 10.1088/1361-6501/ace8b0. Cited by: 1.
- [120] Jieqingxin Zhang, Hui Zhang, Xidong Zhou, Bo Chen, Xiangchuan Wang, and Tao Xu. A graph-based slam method assisted by visual marker in the degenerate scenes. page 1 – 6, 2023. doi: 10.1109/ICARM58088.2023.10218772. Cited by: 0.
- [121] Qiangqiang Huang, Joseph Degol, Victor Fragoso, Sudipta N. Sinha, and John J. Leonard. Optimizing fiducial marker placement for improved visual localization. *IEEE Robotics and Automation Letters*, 8(5):2756 – 2763, 2023. doi: 10.1109/LRA.2023.3260700. Cited by: 0; All Open Access, Green Open Access.
- [122] Manman Yang and Erfu Yang. Two-stage multi-sensor fusion positioning system with seamless switching for cooperative mobile robot and manipulator system. *International Journal of Intelligent Robotics and Applications*, 7(2):275 – 290, 2023. doi: 10.1007/s41315-023-00276-0. Cited by: 0; All Open Access, Green Open Access, Hybrid Gold Open Access.
- [123] Qiangqiang Huang, Joseph DeGol, Victor Fragoso, Sudipta N. Sinha, and John J. Leonard. Optimizing fiducial marker placement for improved visual localization. *IEEE Robotics and Automation Letters*, 8(5):2756–2763, 2023. doi: 10.1109/LRA.2023.3260700.
- [124] Roman Adámek, Martin Brabc, Patrik Vávra, Barnabás Dobossy, Martin Formánek, and Filip Radil. Analytical models for pose estimate variance of planar fiducial markers for mobile robot localisation. *Sensors*, 23(12), 2023. doi: 10.3390/s23125746. Cited by: 0; All Open Access, Gold Open Access, Green Open Access.

- [125] Jiahao Huang, Steffen Junginger, Hui Liu, and Kerstin Thurow. Indoor positioning systems of mobile robots: A review. *Robotics*, 12(2), 2023. doi: 10.3390/robotics12020047. Cited by: 4; All Open Access, Gold Open Access.
- [126] David Jurado-Rodriguez, Rafael Muñoz-Salinas, Sergio Garrido-Jurado, and Rafael Medina-Carnicer. Planar fiducial markers: a comparative study. *Virtual Reality*, pages 1–17, 2023.
- [127] Luan C. Klein, João Braun, João Mendes, Vítor H. Pinto, Felipe N. Martins, Andre Schneider de Oliveira, Heinrich Wörtche, Paulo Costa, and José Lima. A machine learning approach to robot localization using fiducial markers in robotatfactory 4.0 competition. *Sensors*, 23(6), 2023. doi: 10.3390/s23063128. Cited by: 2; All Open Access, Gold Open Access, Green Open Access.
- [128] Taeho Kim, Haneul Jeon, and Donghun Lee. A multi-layered 3d ndt scan-matching method for robust localization in logistics warehouse environments. *Sensors*, 23(5), 2023. doi: 10.3390/s23052671. Cited by: 0; All Open Access, Gold Open Access, Green Open Access.
- [129] Murat Ekici, Ahmet Çağdaş Seçkin, Ahmet Özek, and Ceyhun Karpuz. Warehouse drone: Indoor positioning and product counter with virtual fiducial markers. *Drones*, 7(1), 2023. doi: 10.3390/drones7010003. Cited by: 3; All Open Access, Gold Open Access.
- [130] Andre Kirsch, Malte Riechmann, and Matthias Koenig. Assisted localization of mavs for navigation in indoor environments using fiducial markers. 2023. doi: 10.1109/ECMR59166.2023.10256424. Cited by: 0.
- [131] Andrea Motroni, Paolo Nepa, Valerio Magnago, Alice Buffi, Bernardo Tellini, Daniele Fontanelli, and David Macii. SAR-based indoor localization of UHF-RFID tags via mobile robot. In *2018 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pages 1–8. IEEE, 2018.
- [132] Andrea Motroni, Paolo Nepa, Paolo Tripicchio, and Matteo Unetti. A multi-antenna SAR-based method for UHF RFID tag localization via UGV. In *2018 IEEE International Conference on RFID Technology & Application (RFID-TA)*, pages 1–6. IEEE, 2018.
- [133] Qian Yang, David G Taylor, and Gregory D Durgin. Kalman filter based localization and tracking estimation for HIMR RFID systems. In *2018 IEEE International Conference on RFID (RFID)*, pages 1–5. IEEE, 2018.
- [134] Jian Zhang, Yibo Lyu, Justin Patton, Senthilkumar CG Periaswamy, and Thaddeus Roppel. BFVP: A probabilistic UHF RFID tag localization algorithm using Bayesian filter and a variable power RFID model. *IEEE Transactions on Industrial Electronics*, 65(10):8250–8259, 2018.
- [135] Emidio DiGiampaolo and Francesco Martinelli. A robotic system for localization of passive UHF-RFID tagged objects on shelves. *IEEE Sensors Journal*, 18(20):8558–8568, 2018.
- [136] Haibing Wu, Bo Tao, Zeyu Gong, Zhouping Yin, and Han Ding. A fast UHF RFID localization method using unwrapped phase-position model. *IEEE Transactions on Automation Science and Engineering*, 16(4):1698–1707, 2019.
- [137] Jahanzaib Shabbir and Tarique Anwer. A Survey of Deep Learning Techniques for Mobile Robot Applications. *arXiv preprint arXiv:1803.07608*, 2018.

- [138] Sonia Setia, Akshat Shukla, Amartya Raj, and Abhimanyu Rathore. A detailed review on object detection algorithms. In *2023 7th International Conference on Trends in Electronics and Informatics (ICOEI)*, pages 1521–1528. IEEE, 2023.
- [139] Simranjit Bhumbla, D K Gupta, and Nisha. A review: Object detection algorithms. In *2023 Third International Conference on Secure Cyber Computing and Communication (IC-SCCC)*, pages 827–832, 2023. doi: 10.1109/ICSCCC58608.2023.10176865.
- [140] Ayoub Benali Amjoud and Mustapha Amrouch. Object detection using deep learning, cnns and vision transformers: a review. *IEEE Access*, 2023.
- [141] Youzi Xiao, Zhiqiang Tian, Jiachen Yu, Yinshu Zhang, Shuai Liu, Shaoyi Du, and Xuguang Lan. A review of object detection based on deep learning. *Multimedia Tools and Applications*, 79(33):23729–23791, 2020.
- [142] P. Viola and M. Jones. Rapid Object Detection using a Boosted Cascade of Simple Features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1, pages I–I, 2001. doi: 10.1109/CVPR.2001.990517.
- [143] Navneet Dalal and Bill Triggs. Histograms of Oriented Gradients for Human Detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893. Ieee, 2005.
- [144] Pedro Felzenszwalb, David McAllester, and Deva Ramanan. A Discriminatively Trained, Multiscale, Deformable Part Model. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008. doi: 10.1109/CVPR.2008.4587597.
- [145] Sanjif Rajaratnam. Development of a Real-time Vision Framework for Autonomous Mobile Robots in Human-centered Environments. Master’s thesis, University of Toronto (Canada), 2019.
- [146] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2014.
- [147] Ross Girshick. Fast R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1440–1448, 2015.
- [148] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. *Advances in Neural Information Processing Systems*, 28:91–99, 2015.
- [149] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature Pyramid Networks for Object Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2117–2125, 2017.
- [150] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2961–2969, 2017.
- [151] Zhaowei Cai and Nuno Vasconcelos. Cascade R-CNN: Delving into high quality object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6154–6162, 2018.

- [152] Lile Cai, Bin Zhao, Zhe Wang, Jie Lin, Chuan Sheng Foo, Mohamed Sabry Aly, and Vijay Chandrasekhar. MaxpoolNMS: getting rid of NMS bottlenecks in two-stage object detectors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9356–9364, 2019.
- [153] Anima Pramanik, Sankar K Pal, J Maiti, and Pabitra Mitra. Granulated RCNN and multi-class deep sort for multi-object detection and tracking. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2021.
- [154] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013.
- [155] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016.
- [156] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. SSD: Single Shot MultiBox Detector. In *European Conference on Computer Vision*, pages 21–37. Springer, 2016.
- [157] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal Loss for Dense Object Detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2980–2988, 2017.
- [158] Shifeng Zhang, Longyin Wen, Xiao Bian, Zhen Lei, and Stan Z. Li. Single-shot refinement neural network for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.
- [159] Qijie Zhao, Tao Sheng, Yongtao Wang, Zhi Tang, Ying Chen, Ling Cai, and Haibin Ling. M2Det: A Single-Shot Object Detector based on Multi-Level Feature Pyramid Network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 9259–9266, 2019.
- [160] Joseph Redmon and Ali Farhadi. YOLOv3: An Incremental Improvement. *arXiv preprint arXiv:1804.02767*, 2018.
- [161] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. YOLOv4: Optimal Speed and Accuracy of Object Detection. *arXiv preprint arXiv:2004.10934*, 2020.
- [162] Chien-Yao Wang, I-Hau Yeh, and Hong-Yuan Mark Liao. You Only Learn One Representation: Unified Network for Multiple Tasks. *arXiv preprint arXiv:2105.04206*, 2021.
- [163] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. YOLOx: Exceeding yolo series in 2021. *arXiv preprint arXiv:2107.08430*, 2021.
- [164] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7464–7475, 2023.
- [165] Aabidah Nazir and Mohd Arif Wani. You only look once-object detection models: a review. In *2023 10th International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 1088–1095. IEEE, 2023.

- [166] Hetal Bhaidasna and Zubin Bhaidasna. Object Detection Using Machine Learning : A Comprehensive Review. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, pages 248–255, May 2023. ISSN 2456-3307. doi: 10.32628/CSEIT2390215. URL <https://ijsrcseit.com/CSEIT2390215>.
- [167] Syamimi Abdul-Khalil, Shuzlina Abdul-Rahman, Sofianita Mutalib, SI Kamarudin, and S Sakira Kamaruddin. A review on object detection for autonomous mobile robot. *IAES Int. J. Artif. Intell.(IJ-AI)*, 12(3):1033–1043, 2023.
- [168] Ravpreet Kaur and Sarbjeet Singh. A comprehensive review of object detection with deep learning. *Digital Signal Processing*, 132:103812, 2023.
- [169] Xuxin Lin, Jianwen Gan, Chaohao Jiang, Shuai Xue, and Yanyan Liang. Wi-fi-based indoor localization and navigation: A robot-aided hybrid deep learning approach. *Sensors*, 23(14), 2023. ISSN 1424-8220. doi: 10.3390/s23146320. URL <https://www.mdpi.com/1424-8220/23/14/6320>.
- [170] Takashi Shimoda, Shoya Koga, and Kazuya Sato. Autonomous motion control of a mobile robot using marker recognition via deep learning in gps-denied environments. *Journal of Robotics and Mechatronics*, 35(1):136–144, 2023. doi: 10.20965/jrm.2023.p0136.
- [171] Shushuai Li, Christophe De Wagter, and Guido C. H. E. De Croon. Self-supervised monocular multi-robot relative localization with efficient deep neural networks. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 9689–9695, 2022. doi: 10.1109/ICRA46639.2022.9812150.
- [172] Huijun Li, Ying Mao, Wei You, Bin Ye, and Xinyi Zhou. A neural network approach to indoor mobile robot localization. In *2020 19th International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES)*, pages 66–69, 2020. doi: 10.1109/DCABES50732.2020.00026.
- [173] Zhaojian Li, Yang Gao, Shuqi Wang, and Jiang Liu. Localizability of laser slam robot based on deep learning. In *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 364–369, 2019. doi: 10.1109/ROBIO49542.2019.8961679.
- [174] Zhihuang Zhang, Jintao Zhao, Changyao Huang, and Liang Li. Learning visual semantic map-matching for loosely multi-sensor fusion localization of autonomous vehicles. *IEEE Transactions on Intelligent Vehicles*, 8(1):358–367, 2023. doi: 10.1109/TIV.2022.3173662.
- [175] Hongyi Hao. Researches advanced in visual simultaneously localization and mapping based on deep learning. In *Proceedings of the 2022 4th International Conference on Robotics, Intelligent Control and Artificial Intelligence*, pages 1241–1246, 2022.
- [176] Huijun Li, Yu Zhang, Bin Ye, and Hailong Zhao. A monocular vision positioning and tracking system based on deep neural network. *The Journal of Engineering*, 2023(3):e12246, 2023.
- [177] Xinyu Zeng, Ying He, F. Richard Yu, and Guang Zhou. A Novel Visual SLAM System for Autonomous Vehicles in Dynamic Environments. In *2023 IEEE 98th Vehicular Technology Conference (VTC2023-Fall)*, pages 1–5, Hong Kong, Hong Kong, October 2023. IEEE. ISBN 9798350329285. doi: 10.1109/VTC2023-Fall60731.2023.10333654. URL <https://ieeexplore.ieee.org/document/10333654/>.

- [178] Qamar Ul Islam, Haidi Ibrahim, Pan Kok Chin, Kevin Lim, Mohd Zaid Abdullah, and Fatemeh Khozaei. ARD-SLAM: Accurate and robust dynamic SLAM using dynamic object identification and improved multi-view geometrical approaches. *Displays*, 82: 102654, April 2024. ISSN 01419382. doi: 10.1016/j.displa.2024.102654. URL <https://linkinghub.elsevier.com/retrieve/pii/S0141938224000180>.
- [179] Hanxiao Rong, Alex Ramirez-Serrano, Lianwu Guan, and Xiaodan Cong. Robust RGB-D SLAM for Dynamic Environments Based on YOLOv4. In *2020 IEEE 92nd Vehicular Technology Conference (VTC2020-Fall)*, pages 1–6, Victoria, BC, Canada, November 2020. IEEE. ISBN 978-1-72819-484-4. doi: 10.1109/VTC2020-Fall49728.2020.9348738. URL <https://ieeexplore.ieee.org/document/9348738/>.
- [180] Jianbo Zhang, Liang Yuan, Teng Ran, Song Peng, Qing Tao, Wendong Xiao, and Jianping Cui. A dynamic detection and data association method based on probabilistic models for visual SLAM. *Displays*, 82:102663, April 2024. ISSN 01419382. doi: 10.1016/j.displa.2024.102663. URL <https://linkinghub.elsevier.com/retrieve/pii/S0141938224000271>.
- [181] Lei Cheng, Tian Sun, Yaqi Hu, and Yong Liu. Construction of Semantic Point Cloud Based on Dynamic Object Recognition. In *2023 5th International Conference on Robotics and Computer Vision (ICRCV)*, pages 193–197, Nanjing, China, September 2023. IEEE. ISBN 9798350326369. doi: 10.1109/ICRCV59470.2023.10329076. URL <https://ieeexplore.ieee.org/document/10329076/>.
- [182] Jianbo Zhang, Liang Yuan, Teng Ran, Song Peng, Qing Tao, Wendong Xiao, and Jianping Cui. A dynamic detection and data association method based on probabilistic models for visual SLAM. *Displays*, 82:102663, April 2024. ISSN 0141-9382. doi: 10.1016/j.displa.2024.102663. URL <https://www.sciencedirect.com/science/article/pii/S0141938224000271>.
- [183] Lei Cheng, Tian Sun, Yaqi Hu, and Yong Liu. Construction of Semantic Point Cloud Based on Dynamic Object Recognition. In *2023 5th International Conference on Robotics and Computer Vision (ICRCV)*, pages 193–197, September 2023. doi: 10.1109/ICRCV59470.2023.10329076. URL <https://ieeexplore.ieee.org/abstract/document/10329076>.
- [184] Mina Henein, Jun Zhang, Robert Mahony, and Viorela Ila. Dynamic SLAM: The Need For Speed. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2123–2129, May 2020. doi: 10.1109/ICRA40945.2020.9196895. URL <https://ieeexplore.ieee.org/abstract/document/9196895>. ISSN: 2577-087X.
- [185] Mingrui Li, Yiming Zhou, Guangan Jiang, Tianchen Deng, Yangyang Wang, and Hongyu Wang. DDN-SLAM: Real-time Dense Dynamic Neural Implicit SLAM, March 2024. URL <http://arxiv.org/abs/2401.01545>. arXiv:2401.01545 [cs].
- [186] Seunghyeop Nam, Changseok Woo, Sinkyu Kang, Tuan Anh Nguyen, and Dugki Min. iNAV-driSLAM: An Improved Indoor Self-Driving Framework for Mobile Robots using Deep Reinforcement Learning integrated with SLAM. In *2023 15th International Conference on Advanced Computational Intelligence (ICACI)*, pages 1–8, May 2023. doi: 10.1109/ICACI58115.2023.10146173. URL <https://ieeexplore.ieee.org/abstract/document/10146173>.
- [187] Sergio Cebollada, Luis Payá, David Valiente, Xiaoyi Jiang, and Oscar Reinoso. An Evaluation between Global Appearance Descriptors based on Analytic Methods and Deep

- Learning Techniques for Localization in Autonomous Mobile Robots:. In *Proceedings of the 16th International Conference on Informatics in Control, Automation and Robotics*, pages 284–291, Prague, Czech Republic, 2019. SCITEPRESS - Science and Technology Publications. ISBN 978-989-758-380-3. doi: 10.5220/0007837102840291. URL <http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0007837102840291>.
- [188] Yu-Jen Li, Hsin-Hung Chen, and Chih-Hung G. Li. Enhancement of Visual Place Recognition for Robot Localization Subject to Pedestrian Occlusion. In *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*, pages 348–353, August 2021. doi: 10.1109/CASE49439.2021.9551536. URL <https://ieeexplore.ieee.org/abstract/document/9551536>. ISSN: 2161-8089.
- [189] Aolei Yang, Yanling Chen, Hongzhou Jin, Ling Chen, Renjun Chen, and Yulin Xu. CNN based Relocalization Method for Indoor Moving Robot with Planar State Constraint. In *2021 40th Chinese Control Conference (CCC)*, pages 4296–4301, July 2021. doi: 10.23919/CCC52363.2021.9550606. URL <https://ieeexplore.ieee.org/abstract/document/9550606>. ISSN: 1934-1768.
- [190] Hanxiao Rong, Alex Ramirez-Serrano, Lianwu Guan, and Xiaodan Cong. Robust RGB-D SLAM for Dynamic Environments Based on YOLOv4. In *2020 IEEE 92nd Vehicular Technology Conference (VTC2020-Fall)*, pages 1–6, November 2020. doi: 10.1109/VTC2020-Fall49728.2020.9348738. URL <https://ieeexplore.ieee.org/abstract/document/9348738>. ISSN: 2577-2465.
- [191] Qamar Ul Islam, Haidi Ibrahim, Pan Kok Chin, Kevin Lim, Mohd Zaid Abdullah, and Fatemeh Khozaei. ARD-SLAM: Accurate and robust dynamic SLAM using dynamic object identification and improved multi-view geometrical approaches. *Displays*, 82: 102654, April 2024. ISSN 0141-9382. doi: 10.1016/j.displa.2024.102654. URL <https://www.sciencedirect.com/science/article/pii/S0141938224000180>.
- [192] Xinyu Zeng, Ying He, F. Richard Yu, and Guang Zhou. A Novel Visual SLAM System for Autonomous Vehicles in Dynamic Environments. In *2023 IEEE 98th Vehicular Technology Conference (VTC2023-Fall)*, pages 1–5, Hong Kong, Hong Kong, October 2023. IEEE. ISBN 9798350329285. doi: 10.1109/VTC2023-Fall60731.2023.10333654. URL <https://ieeexplore.ieee.org/document/10333654/>.
- [193] Hesheng Yin, Shaomiao Li, Yu Tao, Junlong Guo, and Bo Huang. Dynam-slam: An accurate, robust stereo visual-inertial slam method in dynamic environments. *IEEE Transactions on Robotics*, 39(1):289–308, 2023. doi: 10.1109/TRO.2022.3199087.
- [194] Jiaming He, Mingrui Li, Yangyang Wang, and Hongyu Wang. Ovd-slam: An online visual slam for dynamic environments. *IEEE Sensors Journal*, 23(12):13210–13219, 2023. doi: 10.1109/JSEN.2023.3270534.
- [195] Qi Zhang, Siyuan Gou, and Wenbin Li. Visual perception system for autonomous driving, 2023.
- [196] Yuansong Cai and Shiqian Wu. A robust slam for highly dynamic environments. In *2022 IEEE 17th Conference on Industrial Electronics and Applications (ICIEA)*, pages 780–785, 2022. doi: 10.1109/ICIEA54703.2022.10006235.

- [197] Teng Ran, Liang Yuan, Jianbo Zhang, Dingxin Tang, and Li He. Rs-slam: A robust semantic slam in dynamic environments based on rgb-d sensor. *IEEE Sensors Journal*, 21(18): 20657–20664, 2021. doi: 10.1109/JSEN.2021.3099511.
- [198] Micha Hei, Jakob Grimm Hansen, Dengyun Li, Micha Kozowski, and Erdal Kayacan. Predictiveslam - robust visual slam through trajectory-aware object masking. In *2023 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1256–1261, 2023. doi: 10.1109/ICUAS57906.2023.10155806.
- [199] Jaafar Mahmoud and Andrey Penkovskiy. Dynamic environments and robust slam: Optimizing sensor fusion and semantics for wheeled robots. In *2023 33rd Conference of Open Innovations Association (FRUCT)*, pages 185–191, 2023. doi: 10.23919/FRUCT58615.2023.10143050.
- [200] Kais Baidi. Development and control of an omnidirectional robot for competition purposes. Master’s thesis, Polytechnic Institute of Bragana, 2024.
- [201] Hamid Taheri, Bing Qiao, and Nurallah Ghaeminezhad. Kinematic model of a four mecanum wheeled mobile robot. *International journal of computer applications*, 113(3): 6–9, 2015.
- [202] G. Bradski. The OpenCV Library. *Dr. Dobb’s Journal of Software Tools*, 2000.
- [203] Roman Admek, Martin Brable, Patrik Vvra, Barnabs Dobossy, Martin Formnek, and Filip Radil. Analytical models for pose estimate variance of planar fiducial markers for mobile robot localisation. *Sensors*, 23(12):5746, 2023.
- [204] Joo Braun, Kas Baidi, Luciano Bonzatto, Guido Berger, Milena Pinto, Rebeca B Kalbermatter, Luan Klein, Vinicius Grilo, Ana I Pereira, Paulo Costa, et al. Design and development of an omnidirectional mecanum platform for the robotatfactory 4.0 competition. In *Climbing and Walking Robots Conference*, pages 114–125. Springer, 2023.
- [205] Dengxing Zhang, Chen Chen, and Guanyu Zhang. Agv path planning based on improved a-star algorithm. In *2024 IEEE 7th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, volume 7, pages 1590–1595, 2024. doi: 10.1109/IAEAC59436.2024.10503919.
- [206] Jim Lawrence, Javier Bernal, and Christoph Witzgall. A purely algebraic justification of the kabsch-umeyama algorithm. *Journal of research of the National Institute of Standards and Technology*, 124:1, 2019.
- [207] Peter Sturm. *Pinhole Camera Model*, pages 610–613. Springer US, Boston, MA, 2014. ISBN 978-0-387-31439-6. doi: 10.1007/978-0-387-31439-6_472. URL https://doi.org/10.1007/978-0-387-31439-6_472.
- [208] Paulo Costa, Jos Gonalves, Jose Lima, and Paulo Malheiros. Simtwo realistic simulator: A tool for the development and validation of robot software. *Theory and Applications of Mathematics & Computer Science*, 1(1):17–33, 2011.
- [209] Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature methods*, 17(3):261–272, 2020.

- [210] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.
- [211] João Braun, Alexandre Oliveira Júnior, Guido S. Berger, José Lima, Ana I. Pereira, and Paulo Costa. Robotatfactory 4.0: a ros framework for the simtwo simulator. In *2022 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pages 205–210, 2022. doi: 10.1109/ICARSC55462.2022.9784794.
- [212] Timothy O Hodson. Root mean square error (rmse) or mean absolute error (mae): When to use them or not. *Geoscientific Model Development Discussions*, 2022:1–10, 2022.
- [213] John Tabak. *Probability and statistics: The science of uncertainty*. Infobase Publishing, 2014.
- [214] João Braun, Alexandre O Júnior, Guido Berger, Vítor H Pinto, Inês N Soares, Ana I Pereira, José Lima, and Paulo Costa. A robot localization proposal for the robotatfactory 4.0: A novel robotics competition within the industry 4.0 concept. *Frontiers in Robotics and AI*, 9:1023590, 2022.
- [215] João Braun, Alexandre Oliveira Júnior, Guido S Berger, José Lima, Ana I Pereira, and Paulo Costa. Robotatfactory 4.0: a ros framework for the simtwo simulator. In *2022 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pages 205–210. IEEE, 2022.
- [216] João Braun, José Lima, Ana I Pereira, and Paulo Costa. Kabsch marker estimation algorithm, a multi-robot marker-based localization algorithm within the industry 4.0 context. *IEEE Access*, 2024.
- [217] João Braun, João Mendes, Ana I Pereira, José Lima, and Paulo Costa. Object detection for indoor localization system. In *International Conference on Optimization, Learning Algorithms and Applications*, pages 788–803. Springer International Publishing Cham, 2022.