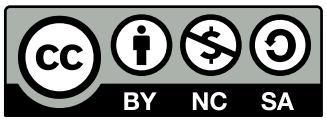
Aalto-yliopisto
Aalto-universitetet
Aalto University

Master's Programme in Mechanical Engineering

Developing a Tracking Based Autonomous Navigation System for Intralogistics in Factories

Muhammad Anis

Master's Thesis
2026



Aalto-yliopisto
Aalto-universitetet
Aalto University

Author Muhammad Anis

Title Developing a Tracking Based Autonomous Navigation System for Intralogistics in Factories

Degree programme Mechanical Engineering

Major Mechatronics

Supervisor Prof. Risto Ojala

Advisor Hari Prasanth S.M. (MSc)

Date 07 July 2026 **Number of pages** 63 **Language** English

Abstract

Intralogistics operations in factories are increasingly being automated using mobile robots. However, most of the currently implemented factory automation solutions rely on fixed paths and predefined target locations, where the robot is commanded to navigate to a predefined position to complete the task. This limits the flexibility of the system and makes it less adaptable to changes in the environment. This is a significant limitation in factory environments, where layouts and object positions can change due to human intervention or operational requirements. To address this limitation, this thesis presents a tracking based autonomous navigation framework in which the robot detects the location of the target object using multiple sensors and then performs the required navigation and intralogistics task based on the detected target position.

The proposed system uses a hybrid target localization approach. UWB measurements are first used to estimate the coarse position of the trolley and guide the robot towards a stand off location. From this position, LiDAR data is used to detect the trolley more accurately by extracting geometric features from the scan data based on the known trolley geometry. The combination of UWB and LiDAR sensing helps to compensate for the limitations of each individual sensing method. After the trolley has been localized, the robot aligns itself with the trolley using an LQR based controller. Successful alignment is followed by docking, attachment, and final transportation of the trolley to the desired goal location.

The system was evaluated in different simulated environments under varying operating conditions, including different robot and trolley positions, trolley orientations, and noise levels in both UWB and LiDAR measurements. The results show that the proposed method is capable of completing the autonomous navigation and docking task in multiple scenarios. The average docking time across the experiments was approximately 22.64 s, while the final longitudinal, lateral, and orientation errors remained small throughout the tests. Overall, the results demonstrate that combining UWB based coarse localization, LiDAR based refinement, and LQR based alignment provides a feasible approach for automation of factories intralogistics. By eliminating dependence on predefined target location, the proposed method can make intralogistics automation more flexible and reliable, as the robot can localize and track the target object even when its position changes.

Keywords Mobile Robots, Intralogistics, UWB, LiDAR, LQR

Preface

This master's thesis was written as part of the "Twin Flow Project". I would like to express my sincere gratitude to them for making this work possible.

I would like to thank Prof. Risto Ojala for supervising this thesis. His expertise, guidance, and attention to detail have greatly contributed to my academic and professional development.

I would also like to thank Mr. Hari Prasanth S.M. for advising my thesis. His valuable insights, continuous support, and guidance throughout the thesis have helped a lot in shaping the direction of this work and contributed significantly to its improvement.

Lastly, I would like to thank all my colleagues at the Autonomous Mobility Lab for creating such a warm, pleasant, and friendly atmosphere, which made working here enjoyable and motivating.

Otaniemi, 07 July 2026

Muhammad Anis

Contents

Symbols and abbreviations

Symbols

p_{uwb}^{map}	UWB position in map frame
$R(\theta_{tf})$	frame rotation matrix
t_{tf}	frame translation vector
p_i	LiDAR point position
$d_{cluster}$	clustering distance threshold
N	number of points in a cluster
c	cluster centroid
$\bar{p}_i$	centered point position
s_{xx}, s_{yy}, s_{xy}	cluster spread terms
tr, det	trace and determinant
λ_i	cluster eigenvalue
v_i	cluster direction vector
E_i	cluster extent
E_{major}, E_{minor}	major and minor extents
r_{line}	line residual
ρ	compactness ratio
τ_i	trolley dimension tolerance
p_j	trolley leg position
e_j	edge vector between trolley legs
$\hat{e}_j$	normalized edge vector
w_p, L_p	width and length parallelism measures
C_t	detected trolley centre
$\hat{w}, \hat{l}$	trolley width and length directions
z_w, z_l	projected cluster positions
W, L	detected trolley width and length
e_x, e_y, e_θ	docking frame errors
x_f	desired final longitudinal position
Δt	controller sampling time
x_k	LQR state vector
u_k	LQR control input
A, B	state and input matrices
Q, R	state and control weighting matrices
J	quadratic cost function
P	Riccati equation solution
K	LQR feedback gain
q_y, q_θ, r_ω	LQR penalty weights
k_x	longitudinal proportional gain
g	speed gate factor
$k_{g,y}, k_{g,\theta}$	speed gate gains
v_{cmd}	commanded linear velocity

Abbreviations

AGV	automated guided vehicle
AMR	autonomous mobile robot
SLAM	simultaneous localization and mapping
LiDAR	light detection and ranging
LQR	linear quadratic regulator
UWB	ultra-wideband
RGB	red, green, and blue
PD	proportional-derivative
PID	proportional-integral-derivative
ROS	robot operating system
TF	transform frame

1 Introduction

The advancement of modern industries and the increasing digitalization of manufacturing sectors have driven the need for smart factories capable of supporting rapid production cycles . As technology has evolved, factories have undergone significant transformations toward automation and digitalization . To sustain high service levels while minimizing costs, efficient work processes are essential. Such efficiency can be achieved by automating operations and reducing reliance on manual labor. One of the most important processes contributing to increase in operational efficiency in industries is automated logistics [1]. Traditionally, raw materials and finished products were handled manually using forklifts or operated by crane operators. However, advancements in robotics and automation have led to a rise in the adoption of mobile robots for intralogistics in factories across the globe. As per International Federation of Robotics [2], nearly 14 percent annual growth has been observed in the number of installed robots in transportation and logistics industry in 2024 and the number is expected to grow even more. A market report published by Ground View Research [3] reported that the global logistic robot market is expected to grow at a compound annual growth rate of 15.9% from 2025 to 2030.

Conventional industrial mobile robots commonly operate by following fixed paths and moving between predefined locations. While this approach can be effective in highly structured environments, it limits the flexibility of the system when operating conditions change [4] . For example, if the target object is moved from its predefined position, the robot may not be able to navigate to the new location without manual reconfiguration. This results in a limitation in their usage especially in factory environments, where these objects are frequently moved to different locations based on production needs. As a result, the dependence on fixed target locations reduces the adaptability of the robot and can impact operational efficiency, especially when timely material transport is required to maintain a smooth manufacturing process.

To overcome the limitations associated with fixed target locations, the robot must be able to localize the target object within the operating environment. For this purpose, several camera based approaches have been explored, including marker based detection, feature based detection, RGB-D sensing, and deep-learning based object detection. These methods can be effective for identifying a target object however, they require the target to be within the camera's field of view. In congested factory environments, the target object may be partially or fully occluded by other objects, which can make reliable detection difficult or, in some cases, impossible. In addition, camera-based detection can be affected by changes in lighting conditions on the factory floor. LiDAR based target detection has also been proposed as a possible solution. In this approach, the geometric features of the object are extracted from scan data and compared with the known geometry of the target object. This can provide useful information for estimating the location of the target object. However, LiDAR based detection also has limitations, since the target object must be within the sensing range and visible to the LiDAR for its geometric features to be detected. UWB based localization provides an alternative approach where a UWB tag is placed on the target object and the UWB anchors are placed in the environment. UWB does not rely on

the visual appearance or geometric features of the target object but they do not provide the accurate location of the target object rather they provide an estimate location of the target object and their measurement usually contain alot of noise. UWB also has a limited operating range, therefore, multiple anchors are required to obtain target pose estimates.

These limitations show that relying on a single sensor is not sufficient for target localization in a factory environment. Instead, the output of multiple sensors must be combined so that the limitation of one sensor can be complimented by the other. Combining camera and LiDAR data can improve local perception, but both sensors still require the target object to be within their sensing range and sufficiently visible. Therefore, this combination may still be affected by occlusion in a factory environments. Similarly, combining UWB with a camera can provide coarse target localization and visual confirmation, but the camera remains sensitive to lighting conditions and visibility constraints.Alternatively, combining UWB with LiDAR appears to be a suitable approach for the target localization because UWB can provide a rough estimate of the target location without requiring direct visual detection as long as the target is in the range of anchors, while LiDAR can be used to detect the accurate location of the target object from this estimate.

In this thesis, a simulation framework is presented for an autonomous navigation system designed for factory intralogistics. The system uses an autonomous mobile robot to transport a trolley from one location to another. To locate the trolley, a hybrid sensing approach is used, where UWB provides a coarse estimate of the trolley position, while LiDAR data is used for more accurate local detection. After estimating the trolley location, the robot navigates from its initial position to the target area using Nav2, which enables collision free path planning and motion execution. Once near the trolley, an LQR based controller is used to accurately align the robot with the trolley before coupling. The framework is developed in simulation, but the system is designed in a way that allows future implementation on real robotic hardware. The obtained results demonstrate that the proposed system can complete the intralogistics sequence, including target detection, autonomous navigation, accurate alignment, coupling, and transportation of the trolley in different simulated factory environments.

2 Related Work

2.1 AGV and AMR Based Intralogistics

Factories intralogistics has traditionally relied on Automated Guided Vehicles (AGVs) for repetitive material transport tasks in structured environments. AGVs are effective when the operating area is well defined and transport routes remain relatively unchanged, as their movement is based on predefined paths and deterministic control strategies [5]. This has made them reliable for fixed route material handling and has reduced the need for manual transport in factory intralogistics. However, the same dependence on predetermined routes limits their adaptability. When production layouts change, new transport routes are required, or temporary obstacles appear in the working area, AGVs often require reconfiguration or infrastructure modifications [6].

Kumbhar et al. [7] reviewed the use of AGVs in small manufacturing enterprises and identified the inflexibility of navigable paths as one of the main limitations affecting their implementation. The authors explained that AGVs can improve material handling in structured manufacturing environments, however, their dependence on predefined navigation paths reduces their adaptability when production layouts, transport routes, or material flow requirements change. Therefore, although AGVs are effective for repetitive factory transport tasks, their flexibility remains limited in manufacturing environments where operational conditions are frequently modified. These limitations have motivated the development of more flexible mobile robotic systems capable of perceiving and reacting to changing industrial environments.

Autonomous Mobile Robots (AMRs) have emerged as a response to the rigidity of conventional AGV based transport. Instead of relying only on predefined routes, AMRs use onboard sensors, localization, mapping, path planning, and obstacle avoidance to navigate in dynamic environments [8]. This enables them to operate in factory settings where human workers, movable objects, and layout modifications are common [9]. The transition from AGVs to AMRs therefore represents a shift from fixed path automation toward environment aware autonomy. This shift is particularly important in intralogistics applications where the robot is not only required to move between different locations, but also to interact with transport equipment such as carts, racks, crates, or trolleys.

The autonomy of AMRs is built on three fundamental capabilities: locomotion, perception, and navigation [10]. Locomotion determines how the robot physically moves, perception provides information about the robot and its surroundings, and navigation determines how the robot reaches a desired position or state. Although mobile robots can use a variety of locomotion mechanisms, wheeled locomotion is the dominant solution in indoor industrial environments because factory floors are typically flat and material transport requires efficient movement of loads . Wheeled AMRs offer high energy efficiency, mechanical simplicity, stability, relatively low cost, and reduced maintenance requirements [11]. These characteristics make them suitable for continuous operation in intralogistics tasks where the robot must move reliably between workstations, storage areas, and loading or unloading points.

Perception is a central requirement for autonomous operation because the robot

must detect objects, extract environmental information, and estimate its relationship to surrounding features. For perception the robot requires sensors. The sensors used in mobile robots are commonly categorized as proprioceptive or exteroceptive, and as active or passive. Proprioceptive sensors measure the internal state of the robot, such as wheel velocity, joint position, or motor feedback, whereas exteroceptive sensors provide information about the external environment, such as distances to nearby objects or environmental landmarks. Active sensors, such as LiDARs, emit energy and measure the returned signal, while passive sensors, such as cameras, rely on naturally available information from the environment [10]. In AMRs, perception is commonly achieved by combining several sensing modalities, including LiDARs, cameras, wheel encoders, and inertial measurement units. Sensor fusion improves robustness and supports localization, obstacle detection, environment understanding, and target tracking.

Navigation combines perception, localization, decision making, and motion control into a coordinated autonomy framework [11]. Perception provides information about the surrounding environment, localization estimates the robot pose, decision making selects the required action, and motion control converts this action into motor commands. In conventional AMR applications, navigation is often formulated as the problem of moving from an initial pose to a target pose while avoiding obstacles. In trolley-based intralogistics, the navigation problem becomes more specific because the target may be defined relative to another object rather than only as a fixed position in a map. The robot must estimate the relative pose of the trolley, approach it from a suitable direction, maintain alignment during motion, and execute controlled docking. As a result, general autonomous navigation methods must be complemented by perception and control strategies that support tracking-based interaction with load-carrying equipment.

Hercik et al. in [12] implemented an AMR for transporting manufactured products between production lines in a smart factory. The authors used map based navigation for general movement and positioning markers to improve localization near the production line. Their results showed that normal map based navigation alone was not sufficient for precise final positioning, while marker based positioning improved the approach accuracy to approximately ±3 mm. However, the study also indicated that accurate final docking remained dependent on additional localization support near the target. This shows that AMR-based factory transport requires not only autonomous navigation, but also reliable target localization for accurate interaction with production equipment.

2.2 Target Location Estimation for Autonomous Navigation

For factory intralogistics, one key requirement is to localize the object of interest, such as a trolley before the robot can approach, dock, and transport it. In this context, target localization is different from general robot localization. General localization estimates the pose of the robot within a global map, whereas target localization estimates the position or pose of the target object relative to the robot or to a local target frame. This relative target pose is important because the robot must not only reach the approximate area of the object, but also approach it from a suitable direction and align with it for

docking.

Camera-based localization is one of the common approaches for estimating the target pose because visual sensors provide rich information about object appearance, shape, texture, and identity. In marker-based methods, a known visual pattern is attached to the target object or docking station, and the detected marker is used to estimate the relative position and orientation between the camera and the target. Richter et al. in [13] used AprilTag-based pose estimation for precise docking of a mobile system at assembly stations. Their work focused on improving the detection of tag edges so that the robot could repeatedly estimate its pose relative to the station with high accuracy. This type of method is useful when the docking target can be instrumented with markers, but it also means that the target must remain visible to the camera and the marker must be placed in a suitable location.

Abbas et al. in [14] analyzed the accuracy and precision of AprilTag based state estimation and showed that marker-based localization is affected by factors such as camera calibration, marker position, viewing angle, and distance from the camera. Their study is important because it shows that fiducial markers can provide reliable relative pose information, but the estimated pose is not equally accurate in all directions or viewing conditions. Therefore, marker-based target localization can be effective for docking, but its performance still depends on the quality of the visual observation and the geometric configuration between the camera and the marker.

Volden et al. in [15] presented a stereo vision based positioning system for autonomous docking, where a learning based object detector was used to identify the target and 3D reconstruction was used to estimate its position. Their method combined data driven object detection with geometric reconstruction, which allowed the system to estimate the target position without relying only on a fiducial marker. The study shows that stereo vision can provide direct depth information for docking tasks, but robust detection remains challenging in complex environments where the target appearance, background, lighting, and viewing conditions can change.

RGB-D sensing extends camera based localization by combining color images with depth measurements. He et al. in [16] proposed a 3D object detection and pose estimation pipeline using RGB-D images. Their method first used template matching to generate candidate object detections, then grouped spatially similar detections into object hypotheses, applied scoring and non-maximum suppression to remove false or duplicate detections, and finally used point cloud processing to compute the 3D object pose. This shows that RGB-D sensors can estimate target pose more directly than RGB cameras because the depth channel relates image detections to 3D space. However, the need for hypothesis generation, scoring, and verification also shows that cluttered scenes can produce false detections unless additional filtering is applied. Camera and RGB-D methods are therefore useful when the target has distinctive visual features, when markers can be attached, or when learning-based detectors can reliably identify the object. Their main advantage is that they can provide both target identity and target pose from visual information. However, their performance depends on the target being visible in the camera's field of view.

LiDAR-based localization provides a different approach because it uses geometric range measurements rather than visual appearance. In 2D LiDAR-based methods, the

target is usually detected from structural features that appear in the laser scan, such as legs, corners, straight edges, or repeated support points. Zhao et al. in [17] proposed a 2D LiDAR based method for autonomous shelf recognition and docking. Their method detected shelf legs from laser scan data using geometric feature extraction and known shelf dimensions. It then used multi shelf model screening and nonlinear least-squares fitting to estimate the shelf center for docking. This type of method is suitable for factory objects such as shelves, carts, and trolleys because these objects often contain repeated geometric structures that can be detected in a planar laser scan.

Fagundes Jr. et al. in [18] presented an analytical approach for object identification and localization using 2D LiDAR data. Their work addressed the lack of a common representation for LiDAR based object localization and proposed a formal way to represent scan data and extract object information. This kind of geometric representation is useful when the target object can be described using measurable features in the scan, such as line segments, corners, or compact clusters. For trolley localization, this supports the idea of using LiDAR scan geometry to detect structural components rather than relying on visual texture or color.The advantage of LiDAR-based target localization is that it can provide accurate distance and shape information, and it is less affected by lighting changes than cameras. However, LiDAR does not naturally provide object identity. A laser scan can show that an object with a certain shape is present, but if several similar trolleys or shelves are in the environment, geometric information alone may not be sufficient to determine which object is the intended target. LiDAR-based methods also require the target structure to be visible in the scan. If the trolley legs are occluded, outside the LiDAR range, or too close to the sensor blind zone, the geometric features needed for localization may not be detected reliably.

UWB based localization provides another alternative because it is based on radio ranging rather than visual or geometric observation. In UWB target localization, tags can be attached to the target object and anchors can be placed in the environment. The measured ranges are then used to estimate the target position. Xu et al. in [19] reviewed UWB based localization for mobile autonomous machines and showed that UWB is widely used for indoor positioning because it can provide range measurements in environments where GNSS is unavailable. This makes UWB useful for industrial target localization because the target can still be identified through its tag even when it is not visually distinctive.

Cao et al. in [20] studied relative localization using multiple UWB ranging nodes. Their work showed that UWB can provide useful relative distance measurements, but estimating a complete relative pose from ranges alone is difficult. To improve the estimate, they equipped robots with multiple UWB nodes, minimized the residual error of the range measurements, and used odometry constraints through sliding window optimization. Their method shows that UWB can support relative localization, but it also indicates that additional constraints or motion information are needed when the goal is to estimate a stable relative pose rather than only a distance.

Morón et al. in [21] presented a benchmark for UWB based infrastructure free positioning and multi-robot relative localization. Their dataset includes UWB ranging together with inertial and odometry measurements, allowing different fusion methods

to be evaluated under varied configurations. Their analysis showed that UWB can provide useful relative localization information, but the ranging error can vary across different setups and operating conditions. This supports the use of UWB as a coarse localization source, while also showing why it may not be sufficient by itself for precise final docking. UWB is therefore useful for identifying and ranging the target object, especially when the object may be hidden or not visually observable. However, UWB measurements are affected by non-line-of-sight propagation, multipath reflections, antenna placement, and obstruction between the transmitter and receiver. These effects can introduce noise and bias into the estimated range. For this reason, UWB is better suited for coarse target localization or identity-aware ranging than for direct final alignment, where the robot needs a more accurate target pose.

Because each sensing method has different strengths and limitations, several studies combine multiple sensors for target localization. Shalihan et al. in [22] proposed a moving object localization method based on the fusion of UWB and LiDAR. In their approach, LiDAR was treated as an accurate but anonymous optical sensor, while UWB was treated as an identity based radio sensor. The method first detected a moving object from LiDAR scans, associated the detected object with UWB range information, rejected suspicious estimates, and then fused LiDAR, UWB, and odometry measurements using pose graph optimization. This shows that UWB and LiDAR complement each other: UWB helps identify the tagged target, while LiDAR provides geometric position information when the object is visible.

Song et al. in [23] proposed UWB and LiDAR fusion for cooperative range only SLAM. Their method fused peer to peer UWB range measurements with 2D LiDAR scan information to localize the robot, estimate the positions of UWB beacons, and build a map. The study showed that LiDAR can improve UWB only localization by providing geometric information about the environment, while UWB can reduce the drift that accumulates in LiDAR based SLAM. This demonstrates the broader benefit of combining radio based ranging with laser based geometry in indoor environments.

Dai et al. in [24] proposed a multi sensor fusion framework for autonomous mobile robot docking by integrating YOLOv8-based AprilTag detection, depth aided 3D localization, and LiDAR based orientation correction. Their method was designed for docking in industrial environments where lighting changes, motion blur, and occlusion can affect purely vision based localization. By combining visual detection, depth information, and LiDAR correction, the system improved the robustness of target localization for docking. This shows that multi-sensor localization is useful when one sensor alone cannot provide reliable target pose information under all operating conditions.

Li et al. in [25] proposed a LiDAR and UWB object tracking method in which LiDAR based 3D object information was fused with UWB measurements to maintain target identity and improve tracking accuracy. Their method addresses a key limitation of LiDAR only localization: the difficulty of maintaining object identity when multiple similar objects are present. By using UWB for identity information and LiDAR for spatial geometry, the system reduces ambiguity during object tracking. This type of fusion is important for factory intralogistics because the robot may operate around several similar trolleys, carts, or racks.

Oh et al. in [26] proposed a regression based docking system that used synchronized monocular camera and 2D LiDAR data during training to estimate docking related quantities such as depth, orientation, and lateral offset. During inference, the trained model used camera input to predict the geometric quantities needed for docking. This shows another way of combining sensing modalities, where LiDAR information can be used to support the learning of target geometry even if the final system relies mainly on vision. Such approaches are useful when the target pose must be estimated from limited onboard sensing, but their performance depends on the quality and coverage of the training data.

2.3 Motion Control for Alignment

After reliable detection of the object of interest, accurate alignment is required for successful latching, coupling, or load transfer. At this stage, the problem is no longer only global navigation or target localization, because the robot must regulate its pose with respect to the target object. Therefore, docking and alignment are commonly formulated using errors relative to target object, such as longitudinal offset, lateral offset, and heading error.

Wang et al. in [27] presented autonomous target docking for non-holonomic mobile robots using vision-based relative pose measurements of the target. Their method uses a stereo camera to obtain the relative range and orientation of the target, while fiducial markers are attached to the target to simplify detection. The docking process is divided into an approach phase and an autonomous docking phase. During the approach phase, the robot moves toward a switching region around the target, after which the autonomous docking controller is activated. Since visual target observations can be noisy and intermittent, an Extended Kalman Filter is used to estimate the target pose in the local reference frame of the robot. In the final docking stage, motion planning is combined with a nonlinear tracking controller that controls the robot motion by tracking a generated path toward the target, allowing the robot to reach the docking point with zero lateral offset and zero impact angle. This shows how docking can be treated as a local target relative control problem after the robot has reached the approximate target region.

Mateos et al. in [28] presented an autonomous docking system for aligning a robotic boat with a docking station. The system used LiDAR for long range navigation and a depth sensing camera for close range target estimation. During the alignment stage, the relative pose of the boat with respect to the docking station was represented using longitudinal, lateral, and yaw errors. Three independent PD controllers were then used to reduce these errors during docking. The method also included a recovery behavior, where the robot reversed, recomputed the target pose, and repeated the docking maneuver if the final alignment condition was not satisfied. Their results show that simple error based controllers can be used for docking when the target relative pose is available, but also that recovery behavior is needed when the final pose cannot be reached in a single attempt.

Fernandez et al. in [29] proposed an alignment and control system for robotic vessels using a multilayered control approach. Their method plans the motion in the

local coordinate frame of the docking target, so the trajectory is defined relative to the object rather than only in a global frame. An AprilTag based perception module estimates the target relative pose, and the resulting longitudinal, lateral, and heading errors are minimized using a Nonlinear Model Predictive Tracking Controller. This formulation shows how docking can be handled as a target frame trajectory tracking problem, where the controller continuously reduces pose errors with respect to the docking object.

Feng et al. in [30] studied autonomous alignment and docking control for a self reconfigurable modular mobile robotic system. Their approach separates the docking task into an optimization based path planning stage and a close range precision control stage. The path planning stage moves the robot module toward the target region, while the final alignment stage is handled by a Lyapunov function based precision controller. The precision controller uses the pose error between the robot module and the desired docking pose to generate stabilizing velocity commands near the docking mechanism. This structure shows how coarse approach planning and local pose stabilization can be combined for accurate final docking.

More recent studies have also considered docking as a constrained control problem, where the robot must satisfy motion limits, visibility constraints, and safety requirements while reducing the target relative pose error. Xiao et al. in [31] proposed an autonomous trolley collection system in which the close range docking motion was controlled using nonlinear model predictive control. The controller was formulated for a non-holonomic mobile robot base and included control barrier functions to satisfy constraints during the docking phase. In particular, the controller was designed to avoid obstacles while keeping the trolley inside the sensor field of view during the final approach. Their experiments on trolley collection tasks show that accurate docking requires simultaneous pose error reduction, obstacle avoidance, and target observability.

Xie et al. in [32] developed a multiple trolley collection system using a non-holonomic mobile robot and proposed a vision based docking controller based on Control Lyapunov Functions and Control Barrier Functions. The control problem was solved online as a quadratic programming problem. In this formulation, the Lyapunov function drives the robot toward the desired trolley relative docking pose, while the barrier function enforces constraints during the approach. Their real world experiments showed improved accuracy and efficiency in trolley collection, demonstrating that trolley docking requires a controller that can handle non-holonomic motion while maintaining accurate target-relative alignment.

Pang et al. in [33] proposed a disturbance aware visual predictive control scheme for autonomous trolley collection. Their method uses active infrared markers for robust visual feature extraction and formulates the docking as a predictive control problem. The controller explicitly considers non holonomic kinematics and visibility constraints, while an extended state observer is used to compensate for disturbances during trolley pushing. Their study shows that trolley docking becomes more challenging when the robot has to maintain alignment while interacting physically with the trolley, because external disturbances can affect the final motion.

Chen et al. in [34] proposed a nonlinear feedback controller for docking a non

holonomic mobile robot to a target pose represented by a virtual landmark. The desired docking pose was estimated using an RGB-D camera, and the controller was designed to drive the robot toward this pose while keeping the target inside the camera field of view. The authors also derived a feasible set of initial conditions from which convergence to the docking pose could be guaranteed. Their experiments showed that the robot could converge to the virtual landmark while satisfying the field of view constraint, showing that final docking control must consider both pose stabilization and target visibility.

Besides docking specific controllers, LQR and Linear Quadratic Gaussian (LQG) methods have also been applied to mobile robot motion control because they provide a systematic way to regulate tracking errors while penalizing control effort. Zhang et al. in [35] proposed an iterative LQR controller for trajectory tracking of a wheeled mobile robot. Their method linearized the robot kinematic model around the reference trajectory and used an iterative LQR formulation to compute local tracking commands. The controller was tested in simulation and experiments on a differential drive robot and was compared with a conventional LQR controller. Their results showed that the iterative method improved the control sequence and produced slightly better tracking performance.

Cornejo et al. in [36] compared polar coordinate control and LQR control for trajectory tracking of a differential wheeled mobile robot. In their formulation, the LQR controller was designed using the robot error model and generated control inputs to reduce the tracking error along the desired trajectory. Their comparison shows that LQR can be used as an error regulation method for differential drive robots, where the control objective is to reduce pose error while respecting the non holonomic motion characteristics of the platform.

Nourizadeh et al. in [37] designed a LQG controller for trajectory tracking of a non-holonomic wheeled mobile robot. The authors first identified a discrete linear model of the robot using experimental data and then designed an LQG controller with a Kalman observer to estimate unmeasured states. Their results showed that linear optimal control can be applied to a non-holonomic mobile robot when the system is represented around a suitable local operating condition. This is important for local alignment control because final docking is usually performed near a target pose where a local error model can be used.

Chatterjee et al. in [38] proposed an optimal cascade sliding mode controller for trajectory tracking of a non holonomic mobile robot. In their control structure, sliding mode control was used in the outer loop and PID control was used in the inner loop, while LQR was applied to optimize the control effort. Their simulations compared the controller performance with and without the optimal LQR component. The results showed that the LQR component reduced unnecessary control action while maintaining trajectory tracking performance, showing how optimal control can be combined with other feedback methods to improve smoothness and control efficiency.

Kim in [39] studied vision based docking of a wheeled rover using linear-quadratic control under sensing uncertainty. The work compared a certainty equivalent LQG controller with a dual control approach that considered both control performance and state estimation uncertainty. The results showed that a straight approach generated

by the certainty equivalent controller could make the range state poorly observable, which affected docking consistency. This shows that docking performance depends not only on the feedback controller, but also on the quality of the target relative state estimate during the approach.

2.4 Research Gap

In this thesis, an autonomous navigation framework for factory intralogistics is presented. The work addresses the limitation of conventional navigation systems that rely on predefined target locations by introducing a more flexible approach in which the target object is localized during runtime. Instead of assuming that the trolley is always located at a fixed and known position, the proposed framework uses a tracking-based strategy to estimate its location in the operating environment. For coarse localization, a UWB tag is attached to the trolley, while UWB anchors are placed in the environment. This allows the system to obtain an approximate estimate of the trolley position, which can be used to guide the robot toward the general target area.

To obtain the accurate trolley pose required for docking, the robot uses its onboard LiDAR once it reaches the vicinity of the trolley. The LiDAR scan data is processed to extract geometric features corresponding to the trolley structure, and these features are verified against the known trolley geometry. This enables the robot to estimate the precise location and orientation of the trolley in the environment. After the trolley pose has been detected, an LQR-based controller is used to align the robot with the trolley and perform the docking . This alignment stage is critical because the robot must correct its position and orientation before going under the trolley in order to avoid collisions with the trolley and complete the docking process smoothly. Overall, the framework combines coarse UWB-based target localization with accurate LiDAR-based pose estimation and feedback control to enable flexible trolley docking without relying on predefined target locations.

3 Methods

3.1 Navigation

Autonomous navigation is one of the main tasks in factory intralogistics automation, and its complexity depends on both the required task and the characteristics of the operating environment. In many intralogistics applications, mobile robots navigate toward predefined target locations, however, this can limit flexibility when the position of the target object changes. To make the system more flexible and adaptable to such changes, the robot should not rely only on predefined target locations, but should instead be able to continuously localize and track the target object during operation. This capability is implemented in the proposed system by combining UWB based coarse localization with LiDAR based pose refinement. In this work, the navigation task is carried out in several stages. First, the robot moves from its initial position to a stand off location using the coarse trolley position obtained from UWB measurements. From this stand off location, LiDAR data is used to detect the trolley more accurately and refine its pose. Based on this refined trolley pose, the robot then navigates to a staging location closer to the trolley. From the staging location, the robot performs the alignment process and continues moving until it reaches the docking position underneath the trolley. Once the docking position is reached, the robot is coupled with the trolley and navigates to the final target location.

The overall navigation process is shown in Figure 1.

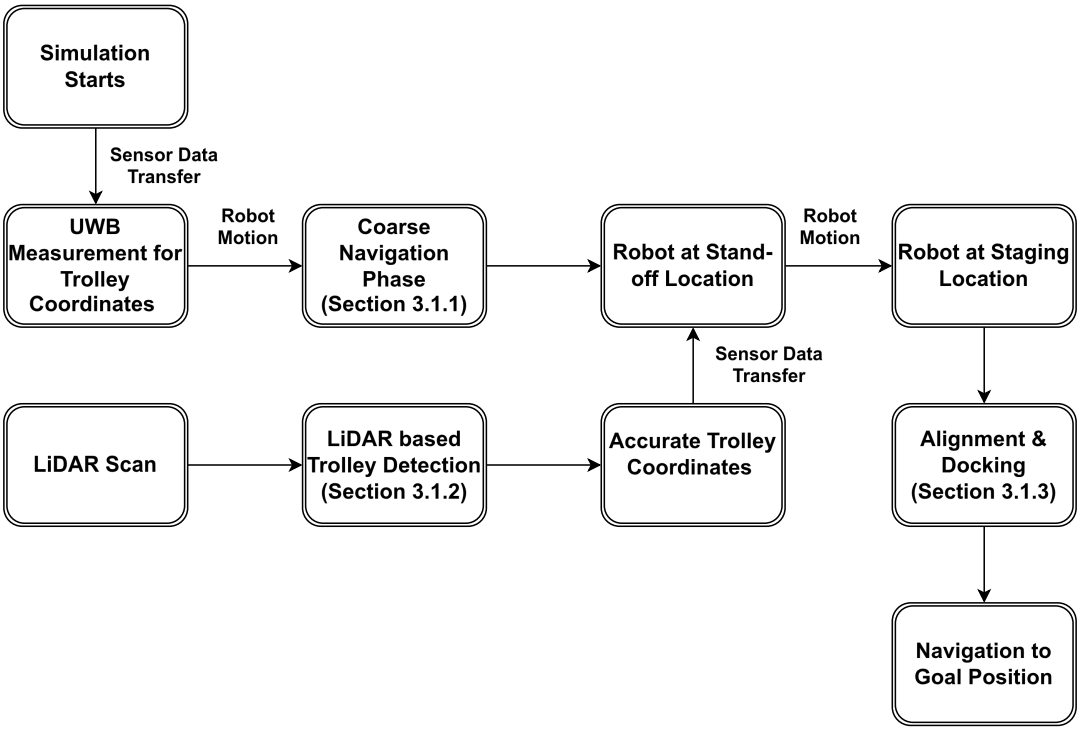


Figure 1: Flowchart representing the autonomous navigation framework presented in this work.

3.1.1 Coarse Navigation Phase

The robot navigation environment chosen for this problem is equipped with UWB anchors and a UWB tag is placed on the target object which is a trolley in this case. UWB based localization is useful in congested indoor environments because it can provide a global estimate of the position of the object of interest even when direct perception using a camera or LiDAR is not possible. However, UWB measurements usually contain significant noise and are therefore not suitable for accurate position detection by themselves. In our case, the UWB estimate is used only as a coarse estimate of the position to guide the robot near the trolley, while the final trolley pose required for docking is obtained using LiDAR based geometric detection.

The coarse trolley position is used to generate a stand off pose in front of the estimated trolley location. A fixed distance is maintained from the trolley estimate so that the robot stops before reaching the trolley. This prevents the robot from moving too close based only on the noisy UWB measurement. After the stand off pose is generated, it is sent to the Nav2 navigation stack as a navigation goal and robot travels to that location. Once the robot reaches the stand off region, or becomes sufficiently close to the trolley estimate, the coarse navigation goal is canceled and the system switches to the LiDAR based detection phase.

The main purpose of this phase is to reduce the search area for the accurate detector. The UWB estimate provides approximate global guidance, while the LiDAR detector later provides the precise trolley geometry required for docking. The overview of the coarse trolley navigation is shown in the Figure 2

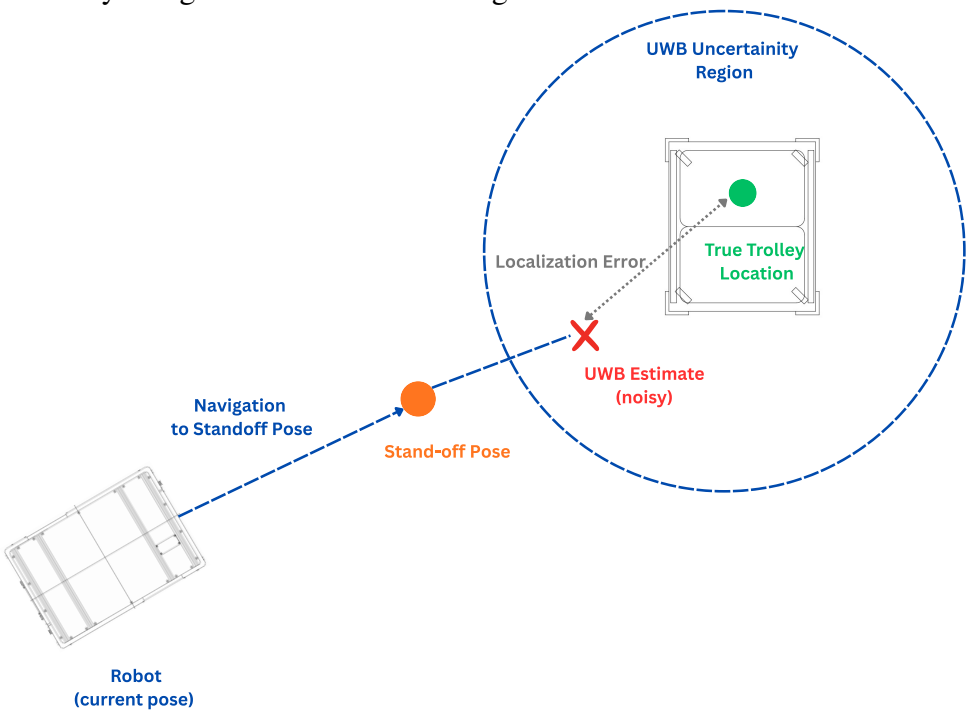


Figure 2: Coarse navigation phase of the proposed framework. The noisy UWB measurement (red) is used to generate the stand-off pose (orange), while the actual trolley location (green) is shown for comparison.

3.1.2 LiDAR Based Trolley Detection

After the robot reaches the coarse stand off goal, the system switches from coarse navigation to accurate trolley pose detection using the 2D LiDAR scan. A number of tests are then performed on this scan data to extract the exact location of the trolley in the environment. The trolley detection using LiDAR is presented in the Fig 3.

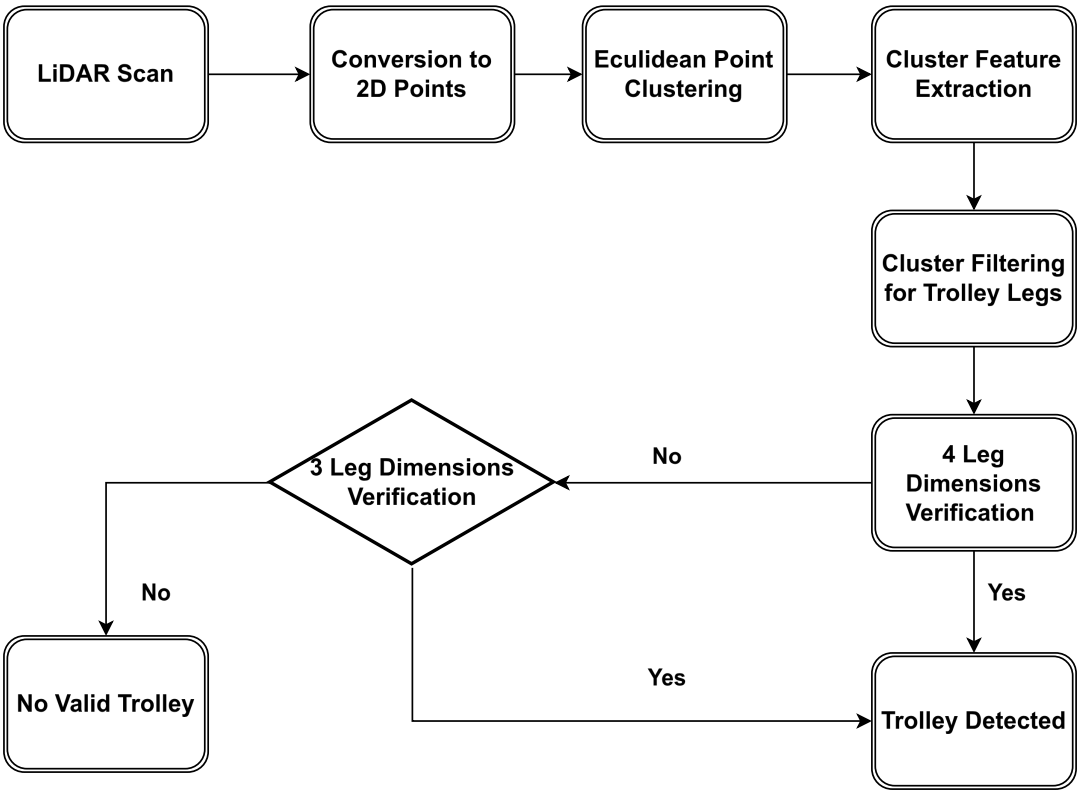


Figure 3: Flowchart representing the steps for detection of trolley from LiDAR scan

A 2d lidar gives range values r_i at different angles θ_i, these values are converted into cartesian coordinates as follows:

$$x_i = r_i cos(\theta_i) \tag{1}$$

and,

$$y_i = r_i sin(\theta_i) \tag{2}$$

So each lidar measurement becomes a 2d point.

$$p_i = (x_i, y_i) \tag{3}$$

For two consecutive points p_i and p_{i-1} , the Euclidean distance between these points is defined as,

$$d_i = ||p_i - p_{i-1}|| \tag{4}$$

For two points to be in the same cluster, the Euclidean distance between these points must be less than the Euclidean distance threshold for the cluster,

$$d_i \leq d_{cluster} \tag{5}$$

If the Euclidean distance between consecutive scan points exceeds the clustering threshold, a new cluster is created. In this work, the clustering threshold is set to 0.06 m.

After clustering, a number of tests are performed on clusters to filter the trolley legs from the LiDAR scan. The detail of all the tests that are performed on the clusters is shown in Figure 4. First of all, each cluster is evaluated based on the number of points it contains. Clusters that do not satisfy the defined minimum and maximum point limits are rejected, since they are unlikely to represent a trolley leg. Therefore, a cluster is considered a possible trolley-leg candidate only if it satisfies the following condition:

$$N_{min} \leq N \leq N_{max} \tag{6}$$

where $N_{\min}$ represents the minimum number of points required for an acceptable cluster, and $N_{\max}$ represents the maximum number of points allowed for an acceptable trolley-leg cluster. A cluster containing less than the minimum number of points may correspond to noise, while a cluster exceeding the maximum point limit is unlikely to represent a trolley leg. In this work, $N_{\min}$ and $N_{\max}$ are set to 2 and 20, respectively. Therefore, clusters consisting of only a single point are rejected as likely noise, while clusters containing more than 20 points are rejected because they are too large to correspond to a trolley leg. The remaining clusters are then further analyzed by checking their major and minor extent, but for calculation of major and minor extent a number of parameters need to be calculated, the centroid for each cluster is calculated as follows:

$$c_x = 1/N \sum_{i=1}^{N} x_i \tag{7}$$

and,

$$c_y = 1/N \sum_{i=1}^{N} y_i \tag{8}$$

So, the centroid for each cluster becomes,

$$c = (c_x, c_y) \tag{9}$$

Centroid provides a compact location of each cluster. After finding the centroid, each point is shifted relative to centroid.

$$\bar{x_i} = x_i - c_i \tag{10}$$

$$\bar{y_i} = y_i - c_i \tag{11}$$

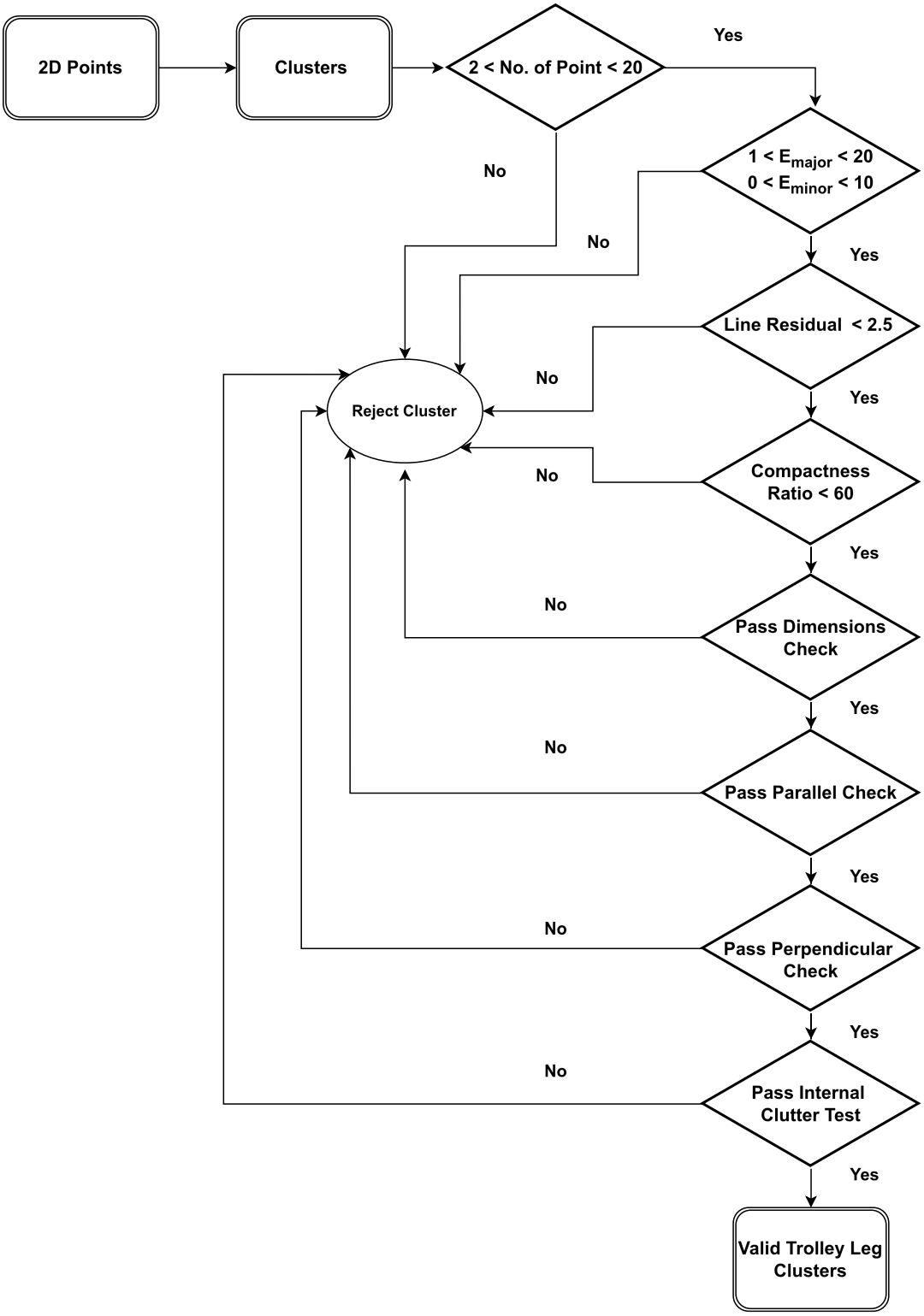


Figure 4: Flowchart of the cluster filtering process used to identify trolley leg clusters from the detected point clusters.

This gives the centered points of the cluster.

$$\bar{p}_i = (\bar{x}_i, \bar{y}_i) \tag{12}$$

The centered points represent the points of the cluster relative to centroid so it helps in estimating the shape of the cluster. The centered points are used to calculate the spread of the cluster in all direction. The spread is calculated as

$$s_{xx} = 1/N \sum_{i=1}^{N} (\bar{x_i})^2 \tag{13}$$

$$s_{yy} = 1/N \sum_{i=1}^{N} (\bar{y_i})^2 \tag{14}$$

$$s_{xy} = 1/N \sum_{i=1}^{N} \bar{x}_i \bar{y}_i \tag{15}$$

The spread value S_{xx} describes the variation of the cluster points in the x-direction. A larger value of S_{xx} indicates that the points are distributed farther from the centroid along the x-axis. Similarly, S_{yy} describes the variation of the cluster points in the y-direction, where a larger value indicates a greater spread along the y-axis.

The term S_{xy} represents the relationship between the variations in the x- and y-directions. A positive value of S_{xy} indicates that x and y tend to increase together, meaning that the cluster has an upward slope. In contrast, a negative value indicates that one coordinate tends to increase while the other decreases, corresponding to a downward slope. When S_{xy} is close to zero, the cluster does not show a strong diagonal trend and may be mostly horizontal, mostly vertical, or approximately symmetric around the centroid. These spreads are then used to estimate the main direction of spread, for main direction trace is calculated which is the sum of spread in both direction.

$$tr = s_{xx} + s_{yy} \tag{16}$$

then the algorithm calculates the determinant,

$$det = s_{xx} s_{yy} - (s_{xy})^2 \tag{17}$$

The determinant tells how the cluster is spread in 2D, if the determinant is large the cluster is spread in both direction and if the determinant is small the cluster is spread in one direction only. Then the trace and determinant are used to calculate the eigenvalues which represent the spread across the two principal directions of cluster.

$$\lambda_1 = tr/2 + \sqrt{(tr/2)^2 - det} \tag{18}$$

and

$$\lambda_2 = tr/2 - \sqrt{(tr/2)^2 - det} \tag{19}$$

λ_1 represents the spread of the cluster in the main direction while λ_2 represents the spread of cluster along perpendicular direction. The trace, determinant and lambda are intermediate values which are used to calculate the main direction of the cluster and their actual value is of no significance. These are used to calculate the corresponding direction using.

$$Sv = \lambda v \tag{20}$$

where S is the spread matrix, λ is an eigenvalue, and v is the corresponding eigenvector. For the larger eigenvalue λ_1, the corresponding eigenvector v_1 is calculated from:

$$(S - \lambda_1 I)v_1 = 0 \tag{21}$$

This gives us the main direction v_1 of the cluster. In our case, this direction is calculated using:

$$v_1 = \begin{bmatrix} \lambda_1 - S_{yy} \\ S_{xy} \end{bmatrix} \tag{22}$$

which is then normalized to form a unit direction vector.

$$\hat{v}_1 = \frac{v_1}{\|v_1\|} \tag{23}$$

The second direction v_2 is chosen perpendicular to v_1 as:

$$v_2 = \begin{bmatrix} -v_{1y} \\ v_{1x} \end{bmatrix} \tag{24}$$

After estimating the main and perpendicular direction each centered point is projected on these direction as:

$$a_i = \bar{p}_i \hat{v}_1 \tag{25}$$

and

$$b_i = \bar{p}_i \hat{v}_2 \tag{26}$$

The dot operation in the above equations represents the dot product of these two vectors. These projected points are used to estimate the amount of spread in each direction.

$$E_1 = max(a_i) - min(a_i) \tag{27}$$

$$E_2 = max(b_i) - min(b_i) \tag{28}$$

The larger spread is called major extent and the smaller extent is called minor extent.

$$E_{major} = max(E_1, E_2) \tag{29}$$

$$E_{minor} = min(E_1, E_2) \tag{30}$$

The major extent simply represents the larger side of the cluster and the minor extent represents the smaller side of the cluster. In this case, for possible trolley leg candidate

the major extend should be between 1 to 20 cm and the minor extent should be at most 10 cm, the clusters not falling in these ranges are filtered out.

Then for each cluster line residual is calculated which is given as:

$$r_{line} = \sqrt{1/N. \sum_{i=1}^{N} b_i^2} \tag{31}$$

b_i is the projection of the centered point in the perpendicular direction. Line residual helps us in understanding how far the spread in a cluster is from main direction. This allows us to filter out the noisy or irregular clusters. In this case, the clusters with line residual of more than 2.5 cm are filtered out.

The compactness ratio for each cluster is also measured which is given by:

$$\rho = E_{major}/E_{minor} \tag{32}$$

It helps us in understanding how scattered the cluster is if major extent is much larger than the minor extent the cluster is very elongated. In this case, the threshold for compactness ratio is 60.0, which is a fairly moderate value preventing the extremely stretched clusters to be accepted as trolley leg candidates.

So, each cluster is basically checked for the possible trolley leg candidate based on four parameters, major extent, minor extent, line residual and compactness ratio. The candidate that pass these tests become the candidate for possible trolley legs. These clusters are then grouped based on their distance from each other, if the distance is less than 1.60 m they are grouped together. This distance is the threshold since any clusters which are separated by more than this distance cannot be the legs of the trolley since this is greater than the maximum length of the trolley. Each group consist of 4 clusters and one cluster can be part of multiple groups and then they are checked if they form a rectangle. The expected trolley length and width in this case are:

$$L_{exp} = 1.13m \tag{33}$$

$$W_{exp} = 0.90m \tag{34}$$

These values correspond to the distance between the trolley legs center in longitudinal and lateral direction. The expected diagonal length is

$$D_{exp} = \sqrt{L_{exp}^2 + W_{exp}^2} = 1.44m \tag{35}$$

so the detected width, length and the diagonal length from these candidates must correspond to the expected dimensions with a little tolerance.

$$L_{det} - L_{exp} \leq \tau_L \tag{36}$$

$$W_{det} - W_{exp} \leq \tau_W \tag{37}$$

$$D_{det} - D_{exp} \leq \tau_D \tag{38}$$

where τ_L , τ_W , τ_D represent the tolerances in the length, width and the diagonal length and their values were 0.18m, 0.20m and 0.22m in this case. The accepted four cluster p_0,p_1,p_2,p_3 that passes the diagonal check are tested for parallelism. For four accepted clusters, edge vectors are calculated as follow:

$$e_0 = p_1 - p_0 \tag{39}$$

$$e_1 = p_2 - p_1 \tag{40}$$

$$e_2 = p_3 - p_2 \tag{41}$$

$$e_3 = p_0 - p_3 \tag{42}$$

In this case e_0,e_1 and e_2,e_3 are adjacent edges and e_0,e_2 and e_1,e_3 are opposite edges. These vectors in the detected trolley are represented in the figure 5.

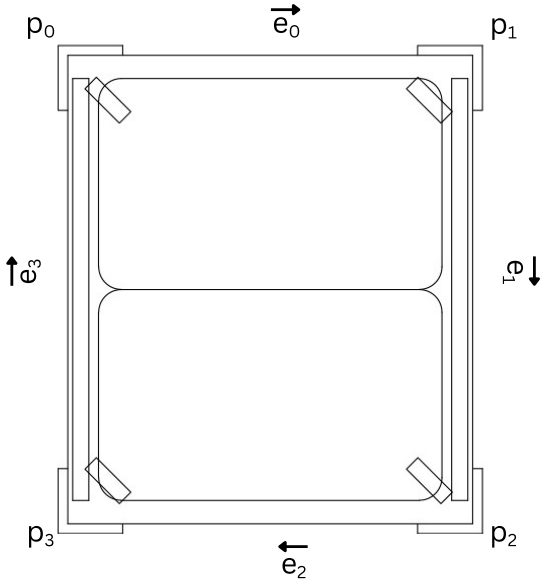


Figure 5: Estimated trolley direction vector obtained from LiDAR-based detection of trolley leg clusters.

The parallel check is performed for the width candidates e_0 and e_2 and the length candidates e_1 and e_3. First these vectors are normalized using

$$\hat{e}_0 = \frac{e_0}{\|e_0\|}, \quad \hat{e}_2 = \frac{e_2}{\|e_2\|} \tag{43}$$

and there dot product is checked for the width side of the trolley.

$$w_p = |\hat{e}_0.\hat{e}_2| \tag{44}$$

The dot operator represents the dot product between two vectors. Ideally, for a parallel case w_p must be equal to one in this case to account for the noise and other disturbances the clusters are considered parallel, if

$$w_p \geq 0.94 \tag{45}$$

Similarly, for length candidates the parallel check is performed as:

$$L_p = |\hat{e}_1.\hat{e}_3| \geq 0.94 \tag{46}$$

The dot operator represents the dot product between two vectors. Then the clusters that pass the parallel test are checked for the orthogonal check. Referring to Figure 5, for the detected clusters to form a trolley the adjacent vectors should be perpendicular and their dot product should be zero. so the following conditions are checked.

$$|\hat{e}_0.\hat{e}_1| = 0 \tag{47}$$

$$|\hat{e}_1.\hat{e}_2| = 0 \tag{48}$$

$$|\hat{e}_2.\hat{e}_3| = 0 \tag{49}$$

$$|\hat{e}_3.\hat{e}_0| = 0 \tag{50}$$

To cater for noise and disturbances, the acceptable limit is 0.06 so if the dot product is less than 0.06 it is considered orthogonal.

The passed candidates are checked for internal clutter, since there can not be any thing under the trolley in the environment so ideally no other cluster should appear inside area formed by these four clusters. So, the remaining clusters are checked if any of them lie in between these four detected clusters. For that the distance is calculated from the detected trolley center to the centroid of the cluster.

$$d = x - C_t \tag{51}$$

where x is the centroid of the cluster being checked and the C_t is the center of the trolley formed by the four detected clusters. Then this distance is projected along length and width direction of the trolley.

$$z_w = d.\hat{w} \tag{52}$$

$$z_l = d.\hat{l} \tag{53}$$

This converts this cluster being tested into trolley's own coordinates. Then this cluster would be checked if its inside the trolley as follows:

$$|z_w| > W/2 \tag{54}$$

$$|z_l| > L/2 \tag{55}$$

where W and L are width and length of the detected trolley. If the distance is less than the half of length or width it means that cluster is located inside the four clusters forming the trolley indicating a false positive for the trolley and that cluster is rejected.

The clusters that pass all these test are finally detected as the legs for the trolley. In case, no four clustered pass all the tests and no valid trolley is detected, then three leg search for the trolley begins, checking the distance between the clusters against the dimensions of the trolley, but in this case no parallel test is performed and directly orthogonal test is performed and then the fourth leg is predicted based on the remaining

three and then the detected trolley is checked for internal clutter similarly to four cluster case.

Once the trolley is confirmed, the docking reference is frozen in the map frame. The detected trolley legs, trolley centre, and selected opening point are transformed from the LiDAR scan frame to the map frame. A local docking frame is then defined with its origin at the trolley opening. The positive x-axis points inward from the opening toward the trolley centre, while the y-axis represents the lateral direction used for alignment.

Based on this docking frame, a staging pose is generated outside the trolley opening. In the current implementation, the staging distance is defined 1.50 m outside the trolley opening while facing the docking direction. This staging pose provides a safe approach location before the robot begins alignment and insertion under trolley.

3.1.3 Alignment and Docking

After reaching the staging location, the robot must be accurately aligned with the trolley before docking. This alignment is performed in the docking frame, where the longitudinal direction is aligned with the trolley opening and the lateral direction represents the side to side offset from the trolley centerline. In this frame, three main errors are evaluated: the longitudinal error e_x, the lateral error e_y, and the yaw error e_θ. The longitudinal error describes whether the robot is too far from or too close to the trolley opening, the lateral error describes the offset from the center of the trolley opening, and the yaw error describes the angular difference between the robot heading and the desired docking direction.

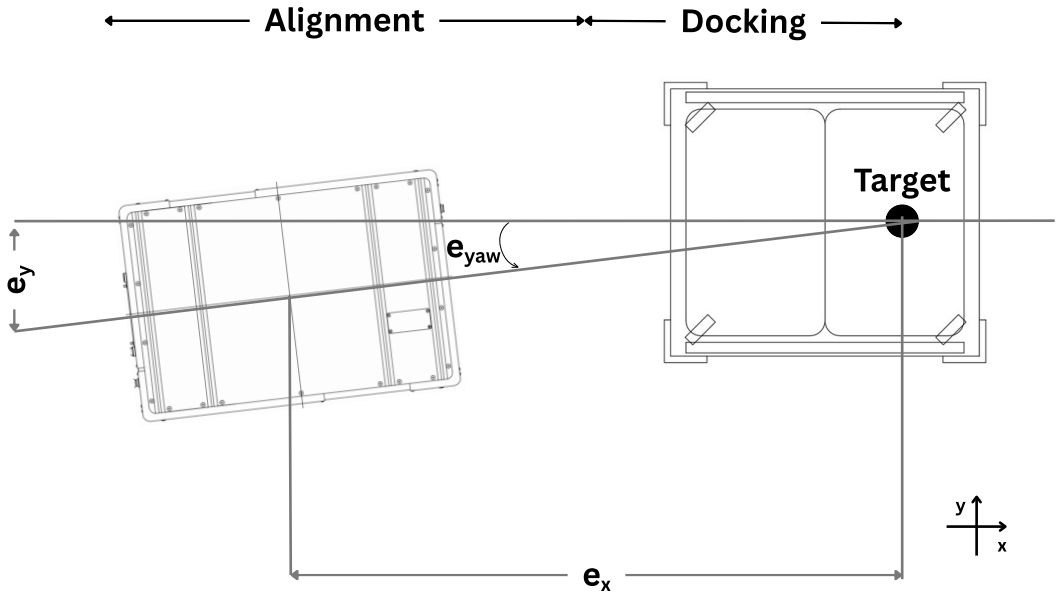


Figure 6: Robot–trolley configuration during the alignment and docking phases.

Before docking, the robot is also checked to ensure that it remains within the predefined alignment corridor. This corridor is located outside the trolley opening and provides a safe region where the robot can correct its position and orientation before

entering the trolley. In addition, the robot footprint is checked against the detected trolley opening to ensure that the robot can physically fit between the trolley legs. Robot is only allowed to go under the trolley once it is sufficiently aligned in position and orientation and when the footprint check is satisfied.

For a differential drive robot, the robot's kinematics are defined in [40] as:

$$\dot{x} = v \cos\theta \tag{56}$$

$$\dot{y} = v \sin\theta \tag{57}$$

$$\dot{\theta} = \omega \tag{58}$$

where x and y represent the robot position in the global coordinate frame, while θ represents the robot heading angle with respect to the global x-axis. The terms $\dot{x}$ and $\dot{y}$ represent the velocity components in the global x and y directions, respectively. The term $\dot{\theta}$ represents the rate of change of the robot heading angle. The variables v and ω represent the linear and angular velocities of the robot, respectively.

From these equations, it can be stated that for a differential drive robot, the robot heading changes directly with angular velocity ω, which means that error in heading angle directly changes with angular velocity. The changes in lateral movement of the robot occurs according to Equation (57). If we write it in terms of errors:

$$\dot{e_y} = v \sin e_\theta \tag{59}$$

Since during alignment, the change in θ is very small and so is the change in the yaw error so we can assume that for this case.

$$\sin e_\theta \approx e_\theta \tag{60}$$

So, the equation (60) becomes:

$$\dot{e_y} = v e_\theta \tag{61}$$

This means that the lateral error is coupled with the yaw error, while the yaw error is influenced by the angular velocity. Consequently, changes in angular velocity affect the yaw error, which in turn influences the lateral alignment of the robot. Therefore, a discrete-time LQR is used to simultaneously correct the lateral and yaw errors. Since a differential-drive robot cannot move sideways directly, the lateral error must be corrected by steering the robot while it moves forward. The angular velocity ω is selected as the control input because it directly affects the robot yaw angle, and the yaw angle subsequently influences the lateral motion of the robot. The discrete time state space error model is written as

$$x_{k+1} = Ax_k + Bu_k \tag{62}$$

where x_k is the error state at time step k, u_k is the control input, and A and B describe the discrete-time error dynamics. In this controller, the LQR state is defined as

$$x_k = \begin{bmatrix} e_y \\ e_\theta \end{bmatrix} \tag{63}$$

where e_y is the lateral error and e_θ is the yaw error. The control input is the angular velocity,

$$u_k = \omega \tag{64}$$

The discrete model used for the LQR controller is

$$A = \begin{bmatrix} 1 & v\Delta t \\ 0 & 1 \end{bmatrix}, \tag{65}$$

and,

$$B = \begin{bmatrix} 0 \\ \Delta t \end{bmatrix} \tag{66}$$

where v is the nominal linear velocity and Δt is the controller sampling period. This model is obtained from the local motion of the robot near the docking direction. When the robot has a yaw error and moves with velocity v, its lateral error changes approximately according to

$$e_y(k+1) = e_y(k) + \Delta t \dot{e_y} \tag{67}$$

Since rate of change in the lateral error $\dot{e_y}$ is given by equation (61), so

$$e_y(k+1) = e_y(k) + v\Delta t e_\theta(k) \tag{68}$$

The yaw error is directly affected by the angular velocity command,

$$e_\theta(k+1) = e_\theta(k) + \Delta t \omega(k) \tag{69}$$

Therefore, the complete discrete-time model becomes

$$\begin{bmatrix} e_y(k+1) \\ e_\theta(k+1) \end{bmatrix} = \begin{bmatrix} 1 & v\Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} e_y(k) \\ e_\theta(k) \end{bmatrix} + \begin{bmatrix} 0 \\ \Delta t \end{bmatrix} \omega(k) \tag{70}$$

The LQR controller determines the angular velocity by minimizing a quadratic cost function. The cost function is given by

$$J = \sum_{k=0}^{\infty} \left(x_k^T Q x_k + u_k^T R u_k \right) \tag{71}$$

where Q is the state weighting matrix and R is the control weighting matrix. The matrix Q determines how strongly the controller penalizes lateral and yaw errors, while R determines how strongly it penalizes angular velocity effort. For this controller, the cost function can be written as

$$J = \sum_{k=0}^{\infty} \left(q_y e_y(k)^2 + q_\theta e_\theta(k)^2 + r_\omega \omega(k)^2 \right) \tag{72}$$

Here, q_y is the penalty on lateral error, q_θ is the penalty on yaw error, and r_ω is the penalty on angular velocity. A larger value of q_y makes the controller more aggressive in reducing lateral offset, while a larger value of q_θ makes it more aggressive in correcting yaw error. A larger value of r_ω discourages large angular velocity commands, resulting in smoother and less aggressive steering.

The values of Q and R are selected separately for the outside alignment and docking phases. During docking, the lateral and yaw error penalties are increased because small misalignment are more critical when the robot is entering the trolley. Therefore, the controller is made stricter during docking than during the outside alignment phase.

To compute the optimal feedback gain, the discrete algebraic Riccati equation given in [41] is solved:

$$P = Q + A^T PA - A^T PB \left(R + B^T PB\right)^{-1} B^T PA \tag{73}$$

The matrix P represents the long term cost associated with the current error state. It is not used directly as a velocity command. Instead, it is used to compute the LQR feedback gain K, which is given by

$$K = \left(R + B^T PB\right)^{-1} B^T PA \tag{74}$$

Finally, the angular velocity command is calculated using the LQR feedback law:

$$\omega = -Kx \tag{75}$$

Since K contains the feedback gains for lateral and yaw error, the controller generates an angular velocity that simultaneously reduces the lateral offset and yaw misalignment. The resulting angular velocity is then limited by predefined saturation bounds to ensure safe and smooth robot motion.

The longitudinal motion of the robot is not controlled by the LQR controller. Instead, it is regulated using a proportional controller based on the longitudinal error in the docking frame. The longitudinal error e_x represents the difference between the current robot position and the desired target position along the docking direction. During the outside alignment phase, this target corresponds to a safe alignment position outside the trolley opening. During the docking phase, the target corresponds to the final insertion depth inside the trolley.

The longitudinal error is defined as

$$e_x = x - x_f \tag{76}$$

where x is the current robot position in the docking frame and x_f is the desired longitudinal position. The linear velocity command is then calculated using proportional control:

$$v = -k_x e_x \tag{77}$$

where k_x is the proportional gain for the longitudinal controller. The negative sign ensures that the robot moves in the direction that reduces the longitudinal error. For

example, if the robot is too far behind the target position, it moves forward. If it is too far ahead during the alignment phase, it can move backward to return to the desired alignment location.

The longitudinal velocity is controlled separately from the LQR because the robot can directly actuate motion in the longitudinal direction using its linear velocity command. Therefore, the x direction error can be corrected using a simple proportional controller. In contrast, the lateral error cannot be corrected directly because the robot cannot move sideways. The lateral error must be corrected indirectly by changing the robot heading while moving. For this reason, the LQR controller is used for the coupled lateral and yaw dynamics, while the longitudinal motion is handled separately.

To improve safety, the longitudinal velocity is also modified using a speed gate. The speed gate reduces the forward velocity when the lateral error or yaw error is large. It is defined as

$$g = \frac{1}{1 + k_{g,y}|e_y| + k_{g,\theta}|e_\theta|} \tag{78}$$

where g is the speed gate, e_y is the lateral error, e_θ is the yaw error, $k_{g,y}$ is the gain associated with lateral error, and $k_{g,\theta}$ is the gain associated with yaw error. The final longitudinal velocity is then calculated as

$$v_{\text{cmd}} = gv \tag{79}$$

This means that when the robot is well aligned, e_y and e_θ are small, so g is close to one and the robot can move normally. When the robot has a large lateral or yaw error, g becomes smaller and the robot slows down. This gives the controller more time to correct the alignment before the robot moves further toward or into the trolley. The final velocity command is also limited by predefined minimum and maximum velocity bounds to ensure smooth and safe motion.

After alignment and insertion, the robot reaches the correct coupling position relative to the trolley. Since the system is implemented in simulation rather than on the real robot, the trolley is connected using an attachment plugin instead of a physical coupling actuator. Once the attachment plugin connects the trolley to the robot, the robot footprint is updated to include the trolley dimensions so that Nav2 can safely plan for the combined robot-trolley system. Nav2 then moves the trolley to the target position, after which the attachment plugin is used to detach the trolley, the footprint is restored to its original size, and the task is completed.

3.2 Simulation Setup

3.2.1 Simulation Environments

To test the algorithm in simulation, ignition gazebo is used as a simulator, the algorithm is tested in three different environments with details below

The first simulated environment represents a physical laboratory space located at Aalto University. The laboratory has an approximately rectangular layout with dimensions of about 5.7 m × 8.5 m. Most of the static obstacles are positioned near

the boundaries of the environment, while the central region provides relatively open workspace for robot motion and task execution.

The environment contains several objects commonly found in an industrial laboratory setting, including storage racks, pallets, workbenches, and other laboratory equipment. These objects create partially constrained and cluttered regions within the workspace, making the environment suitable for evaluating the robustness of the proposed navigation algorithm. In particular, the layout allows the algorithm to be tested under realistic conditions involving limited clearance, static obstacles, and constrained maneuvering space. The environment is shown in the Figure 7

Figure 7: Cage Lab used as environment 1 in the simulation.

The second test environment represents an factory warehouse scenario with approximate dimensions of 24 m × 40 m. This environment was selected because it closely reflects the intended application domain of the proposed trolley tracking and autonomous navigation algorithm. The warehouse layout consists of multiple storage racks and storage units arranged in close proximity, forming narrow aisles and constrained maneuvering regions.

Compared with the laboratory environment, the warehouse scenario provides a more challenging test case. The narrow aisles between the storage racks impose strict spatial constraints on the robot, requiring accurate path following, reliable trolley tracking, and precise alignment during approach and docking.These constraints are particularly relevant for evaluating the algorithm's ability to operate in realistic intralogistics environments where available space is limited.

In addition, the warehouse contains several objects with geometric features similar to the target trolley. This increases the complexity of trolley detection and tracking, as the perception system must distinguish the correct trolley from geometrically similar

surrounding objects. Therefore, this environment provides a challenging evaluation scenario for assessing both the robustness of the trolley identification process and the reliability of the robot's navigation behavior in a cluttered factory setting. The environment is shown in Figure 8

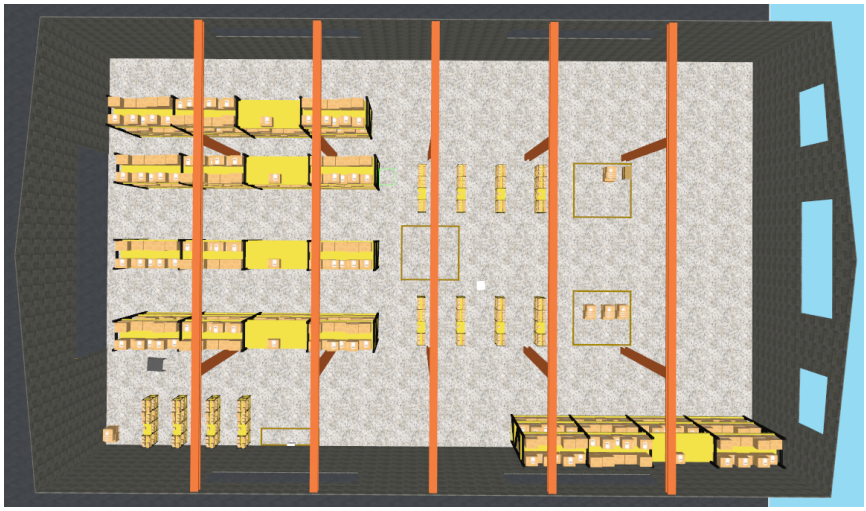


Figure 8: Factory Warehouse used as environment 2 in the simulation.

The third test environment represents a factory like setting with approximate dimensions of 42 m × 60 m. In contrast to the other two environments, this scenario provides a more open workspace with fewer spatial constraints. The environment includes several industrial objects and equipment placed within specific regions, creating a layout representative of a factory floor.

This environment was selected to evaluate the proposed trolley tracking and navigation algorithm in a larger and less congested factory scenario. The open layout allows the robot to move over longer distances while still requiring reliable tracking, obstacle avoidance, and alignment with the target trolley. Therefore, this environment provides a complementary test case for assessing the algorithm's performance beyond highly constrained or cluttered spaces. The environment is shown in Figure 9

3.2.2 Robot and Trolley Model for Simulation

The robot platform used for this project is Neobotix Rox Diff, which is a mobile robot with differential drive configuration. The robot uses a two wheel differential drive system for motion, making it suitable for navigation tasks in structured indoor environments. The platform has overall dimensions of 811 mm × 680 mm and is equipped with two LiDAR sensors and a camera, which provide the perception capabilities required for environment sensing, localization, and navigation. This platform is primarily intended for indoor operation and is well suited for industrial and factory-like environments. For this reason, it was selected for this project, as its design

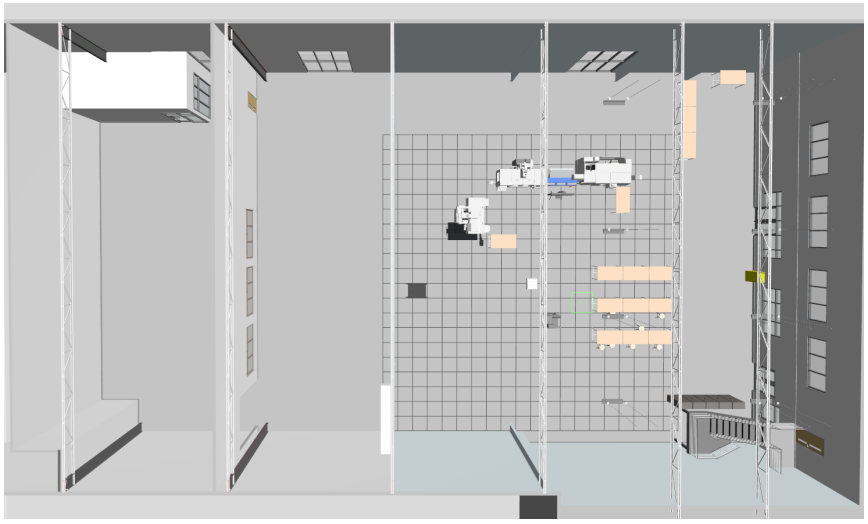


Figure 9: Factory Floor used as environment 3 in the simulation.

and intended application are consistent with the operating conditions considered in this project. The mobile platform is shown in the Figure 10.

Figure 10: Neobotix ROX-Diff mobile robot platform used as the robot model in the simulation.[42]

For the simulation of the robot in Gazebo, an open-source model is provided by Neobotix. The model closely represents the physical robot platform and includes the main mechanical and sensing components required for this work. The simulated robot model consists of two drive wheels for differential motion and four castor wheels for structural support and stability.

The simulated platform also includes the sensors required for navigation and perception, namely an IMU, two 2D LiDAR sensors, and an RGB-D camera. The LiDAR sensors were configured to publish laser scan data at 10 Hz, while the IMU operated at 50 Hz. The camera provided RGB-D data at 30 Hz with a resolution of 640 × 480 pixels. Sensor noise is also included in the simulation to achieve more realistic sensor behavior. This sensor configuration made the robot model suitable for testing localization, perception, and autonomous navigation in the simulated environment.

The trolley model used in this project has dimensions of 1200 mm × 800 mm and

is supported by four wheels. Its structure is supported only at the four corners, leaving the central area underneath unobstructed. This prevents the trolley from blocking the robot's LiDAR field of view during operation. The trolley dimensions were selected to provide sufficient clearance for the robot to fit underneath during alignment and docking. The trolley model used in this simulation is shown in Figure 11

Figure 11: Trolley model used in the simulation for the navigation task.

3.2.3 Testing Setup

To validate the algorithm developed for the navigation of the trolley, tests were conducted in three different environments as described earlier under varying operating conditions. These conditions were varied by changing the accuracy of the UWB output, the level of noise in the LiDAR data, orientation of the trolley and also position of both trolley and the robot.

In simulation, the UWB is simulated by adding gaussian noise to the accurate trolley coordinates , this was done to make the simulation conditions more realistic as UWB measurements contain inherent noise in actual operation. The trolley position obtained form trolley transform is given by:

$$p_t = \begin{bmatrix} x_t \\ y_t \end{bmatrix} \tag{80}$$

where x_t and y_t represent the x and y coordinates of the trolley. After receiving the coordinates of the trolley the gaussian noise is added to it. For gaussian noise, the range is defined between 0.3 to 1 m. This is the typical error in the measurement of a UWB measurement. Every time before publishing the trolley location a random noise value is selected from this range and a noise matrix is generated as:

$$n = \begin{bmatrix} n_x \\ n_y \end{bmatrix} \tag{81}$$

This noise is added to the trolley pose to generate the simulated UWB estimate of the trolley location.

$$p_{uwb} = p_t + n = \begin{bmatrix} x_t \\ y_t \end{bmatrix} + \begin{bmatrix} n_x \\ n_y \end{bmatrix} = \begin{bmatrix} x_{uwb} \\ y_{uwb} \end{bmatrix} \tag{82}$$

This noisy trolley position is published as a coarse pose estimate. When the docking node receives this pose, it first checks whether the estimate is recent enough to be used. The received pose is then converted to the map frame.

$$p_{uwb}^{map} = R(\theta_{tf})p_{uwb} + t_{tf} \tag{83}$$

where $R(\theta_{tf})$ is the rotation matrix and t_{tf} is the translation vector.

Similarly, noise was added to the LiDAR measurements because real LiDAR sensors are also affected by measurement errors. The LiDAR included a Gaussian noise of 0.01 m, meaning that each LiDAR reading contained a random error with a standard deviation of 1 cm. Without these noise, the simulation would represent an ideal environment rather than conditions that are closer to real world sensor behavior.

The test conditions were further modified by changing the initial position of the robot, as well as the position and orientation of the trolley. This made it possible to evaluate the algorithm across multiple scenarios and assess its reliability and robustness.

Three initial trolley orientations were considered with respect to the robot's approach direction:

- The trolley is positioned in the same direction as the robot's approach direction, resulting in an approximate relative orientation of 0°.
- The trolley is positioned perpendicular to the robot's approach direction, resulting in an approximate relative orientation of 90°.
- The trolley is positioned at an intermediate angle with respect to the robot's approach direction, resulting in a non-zero relative orientation other than 90°.

The orientation of the trolley was varied to evaluate the reliability and robustness of the controller used for robot and trolley alignment. Testing the trolley at different angles makes it possible to assess how well the controller can achieve the desired final docking pose while maintaining the required tolerances in the longitudinal, lateral, and yaw directions.

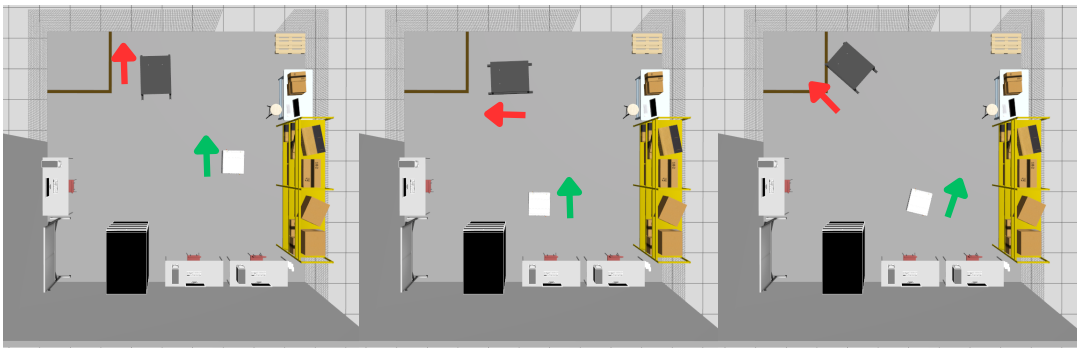


Figure 12: Different trolley orientations used for testing in the simulation.

In addition to changing the orientation of the trolley, the physical locations of both the robot and the trolley were also varied within the environment. Changing the locations is important to test the algorithm under different obstacle layouts. The positions of the trolley and the robot used in the testing are shown in Figures 13, 14, 15

Figure 13: Different initial testing positions of the trolley and the robot in simulation environment 1, where × represents the trolley location and • represents the robot location.

Figure 14: Different initial testing positions of the trolley and the robot in simulation environment 2, where × represents the trolley location and • represents the robot location.

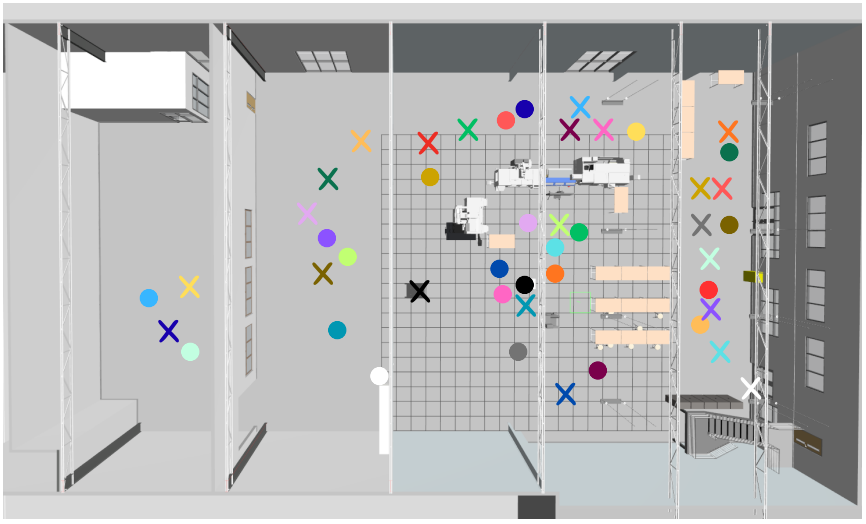


Figure 15: Different initial testing positions of the trolley and the robot in simulation environment 3, where × represents the trolley location and • represents the robot location.

The positions marked in the these figures approximately represent the locations of the trolley and the robot during testing, some of these location might be overlapping in simulation but are shown a bit far apart in the figure for clarity. Also, note that for each of these trolley location, the trolley was tested in all the three orientation as defined earlier.

4 Results

The evaluation was primarily based on the success of the docking process, although the overall success of the complete mission was also considered. The first performance parameter analyzed was the time required for the robot to complete docking. This includes both the alignment of the robot with the trolley and its subsequent docking beneath the trolley at the desired position.

For successful docking and transportation of the trolley, the robot must dock accurately at the intended location underneath the trolley. Therefore, positional misalignment was considered as an important evaluation parameter. This includes lateral and longitudinal position errors, as well as orientation error between the robot and the trolley.

In addition to docking performance, the overall behavior of the robot during the complete mission was also evaluated. For this purpose, the overall mission speed was considered, which was calculated as the ratio of the total mission length to the total mission time.

4.1 Docking Time

Docking time was measured for each test case across three different simulation environments. In each environment, the tests were carried out by varying both the position and orientation of the trolley, as well as the initial starting position of the robot model. This allowed the docking algorithm to be evaluated under different approach conditions and helped assess its consistency across a range of scenarios. In total, results were obtained from 70 simulation trials in each of the three environments.

The average docking time in the first simulation environment was approximately 25.13 s, with a standard deviation of 5.13 s. In the second environment, the average docking time was almost similar to first one, at approximately 25.15 s, with a standard deviation of 4.6 s. The third simulation environment resulted in a lower average docking time of approximately 17.66 s. However, it also showed a higher standard deviation of 6.88 s, indicating greater variability in the docking performance within that environment.

The third environment has more open and less congested areas, which results in generally lower docking times because the robot can align with the trolley quickly regardless of its placement or orientation. However, a few congested regions still require additional maneuvering when the trolley is placed there, increasing the docking time in those cases. As a result, the mean docking time remained low, while the variation caused by these few difficult locations resulted in relatively higher standard deviation.

Across the individual tests, the trolley orientation had a clear effect on docking time. When the trolley was placed at an angle to the robot's approach direction, the algorithm generally required more time to align the robot with the trolley, resulting in the highest docking times. Perpendicular trolley orientations required relatively less time, while the lowest docking times were observed when the trolley was aligned with the robot's approach direction.

The level of noise in the UWB measurements had no direct impact on the docking time, since the UWB measurements were not used during the docking phase. The UWB noise only affected the estimation of the stand-off location, from which the robot navigated toward the staging location. Docking started only after the robot had reached the staging location; therefore, variations in UWB noise did not directly influence the docking time.

The docking time obtained across the three environments shown in the Figure 16 and the distribution of these results are shown in Figure 17

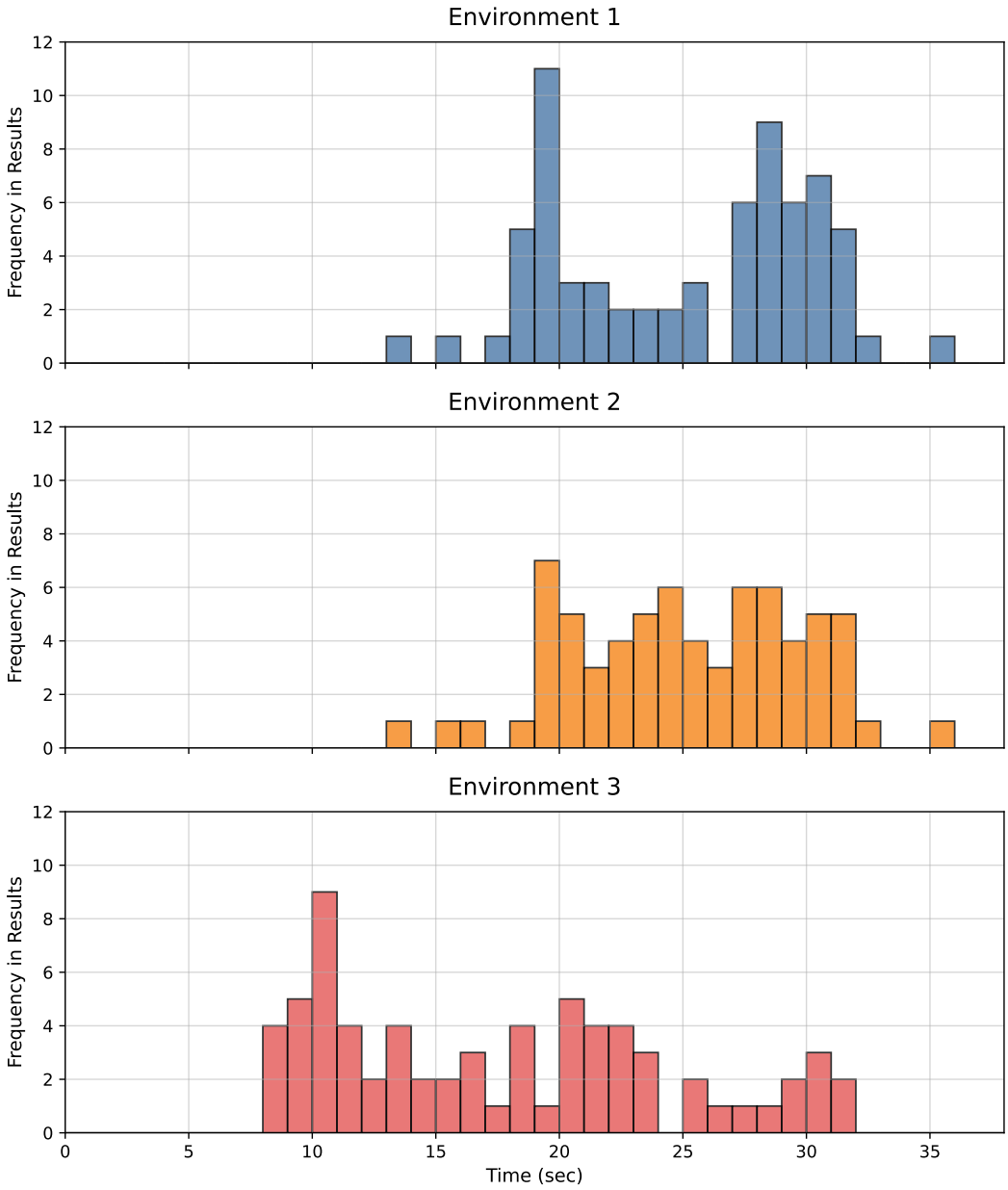


Figure 16: Histograms of the docking time observed across three environments.

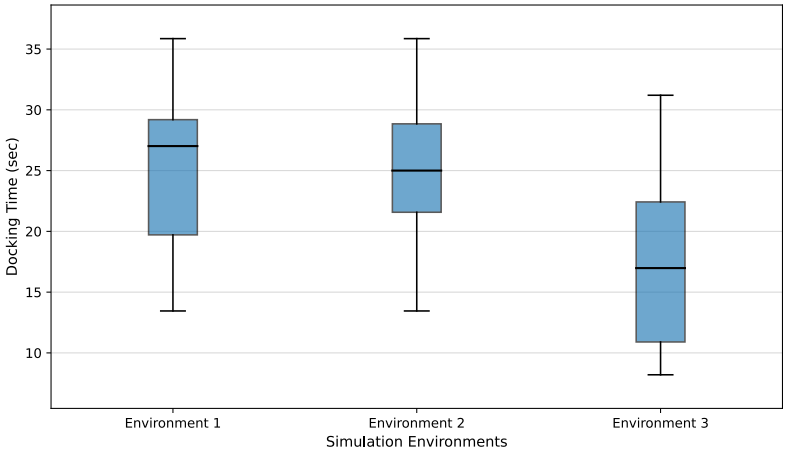


Figure 17: Distribution of the docking time observed in each environment

The spread in the results is mainly attributed to the orientation of the trolley and the docking location of the trolley, the higher docking time in these figures represent the conditions when the trolley was placed at an angled orientation with respect to trolley. The maximum docking time was observed in the cases when the trolley was approximately at 45 deg from the approach angle of the robot and the minimum time was observed when the angle between the robot and the trolley was close to zero. Figure 17 represents the distribution of these results in each environment, the blue area represents the main spread in the data while whiskers represent the upper and lower limits observed in the data. The highest values of the docking time represent some of the cases where the robot had to attempt the docking more than once to align itself with the robot because of the large misalignment between the robot and the trolley.

4.2 Position Misalignment

To evaluate how accurately the trolley position was detected and how precisely the algorithm reached the desired final docking location, the position error was measured. The position error represents the final misalignment of the robot with respect to the trolley in both the lateral and longitudinal directions. This is an important performance parameter because accurate positioning is required for successful docking and coupling.

Lateral alignment is particularly important to ensure that the robot can move underneath the trolley without colliding with the trolley side walls. If the lateral error is too large, the robot may not be centered correctly with respect to the trolley, which can prevent smooth docking and increase the risk of contact with the trolley structure. Similarly, longitudinal alignment is critical because the robot must reach the correct position underneath the trolley for the coupling mechanism to operate properly. Any significant longitudinal offset may result in incomplete or failed coupling.

The position error was evaluated for all test cases in the three simulation environments using the same testing procedure as for docking time. For the longitudinal error, the mean values were -0.49 cm, -0.68 cm, and 0.70 cm in the first, second, and third environments, respectively, with corresponding standard deviations of 0.65 cm, 0.70

cm, and 0.50 cm. For the lateral error, the mean values were 0.18 cm, 0.15 cm, and 0.45 cm, with standard deviations of 0.55 cm, 0.45 cm, and 0.28 cm, respectively.

Overall, the longitudinal errors remained within a similar range across all three environments. The lateral errors in the first two environments were also comparable, while the third environment showed a slightly higher mean lateral error. The difference in the average values shows a little variation in the performance of the controller but the magnitude of the error remained still very small in all cases. The longitudinal error across the three environment is shown in Figure 18 and the distribution of longitudinal error results is shown in Figure 20, the lateral error across three environments is shown in Figure 19 and the distribution is shown in Figure 21

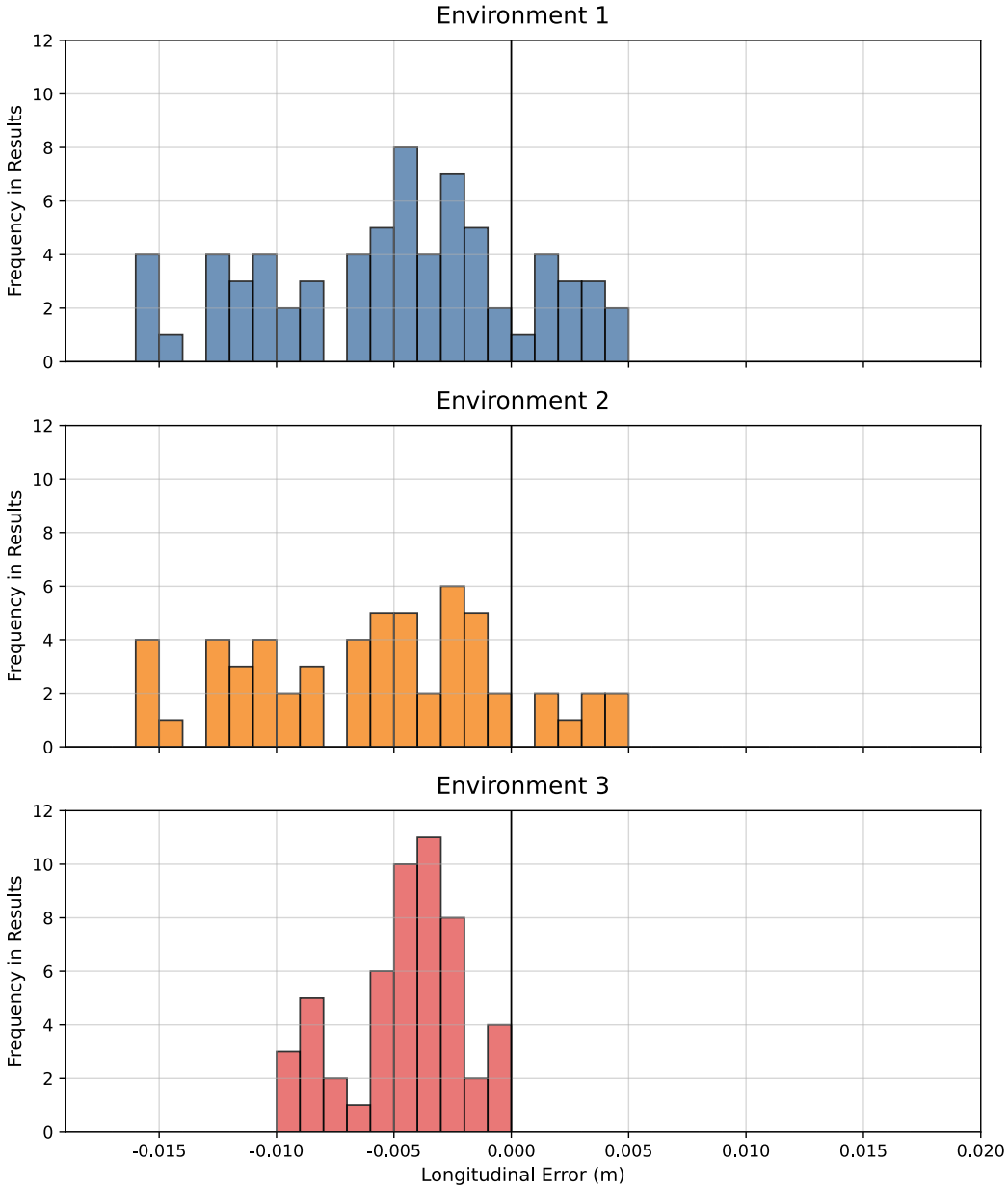


Figure 18: Histograms of longitudinal error observed across three environments.

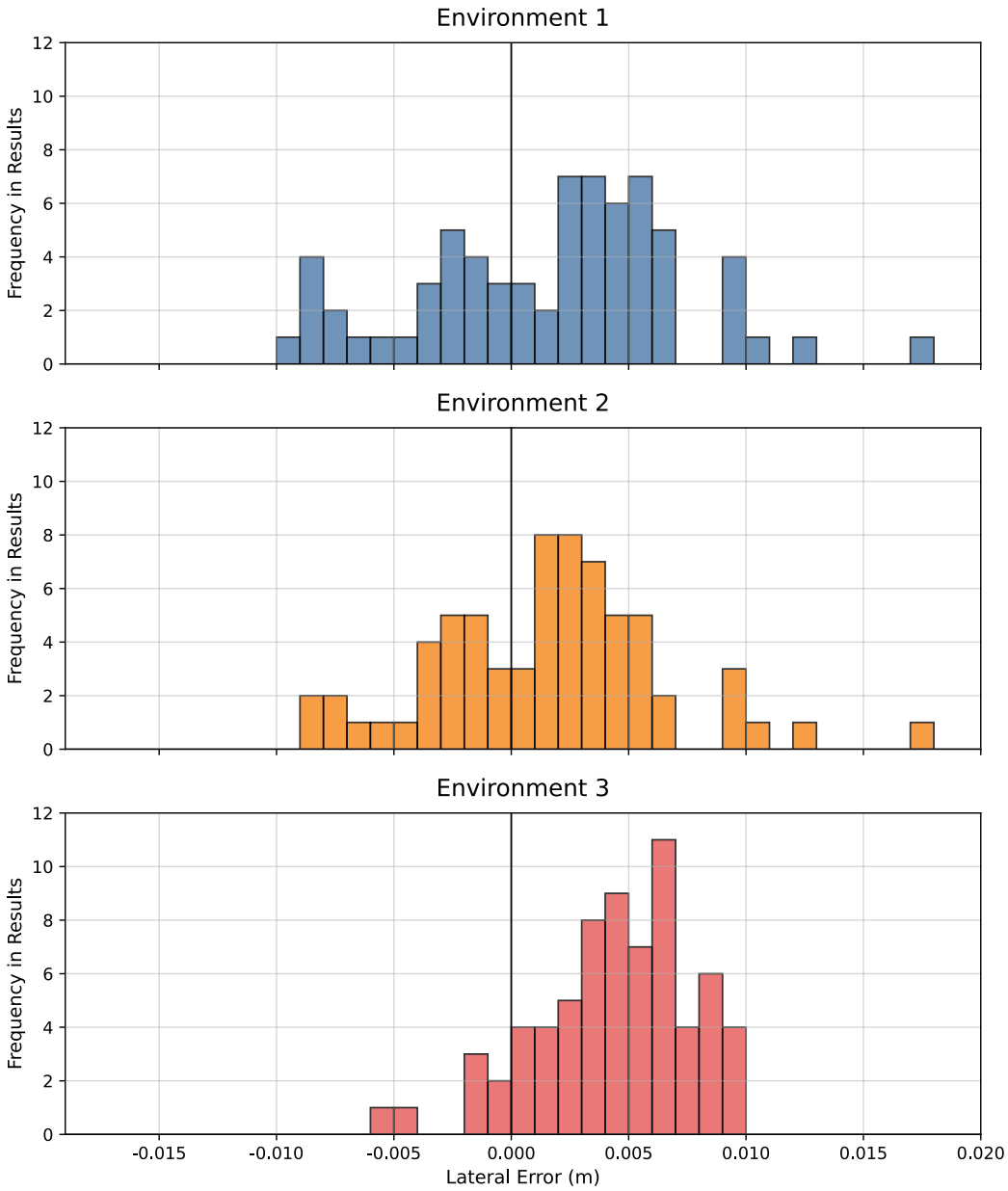


Figure 19: Histograms of lateral error observed across three environments.

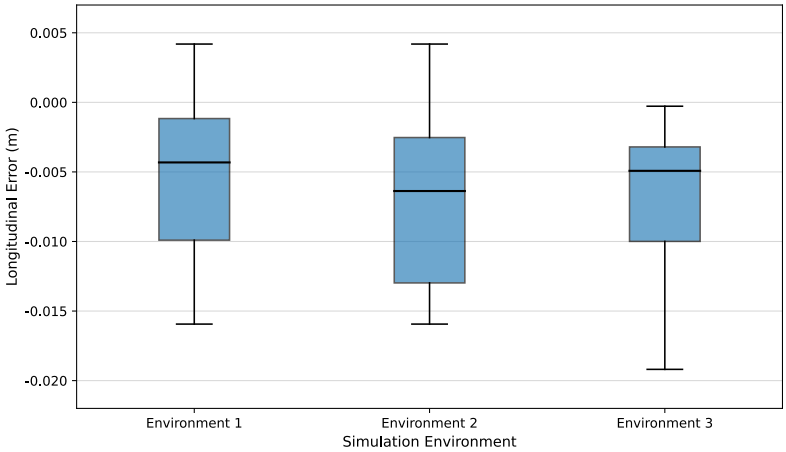


Figure 20: Distribution of longitudinal error observed across each environments

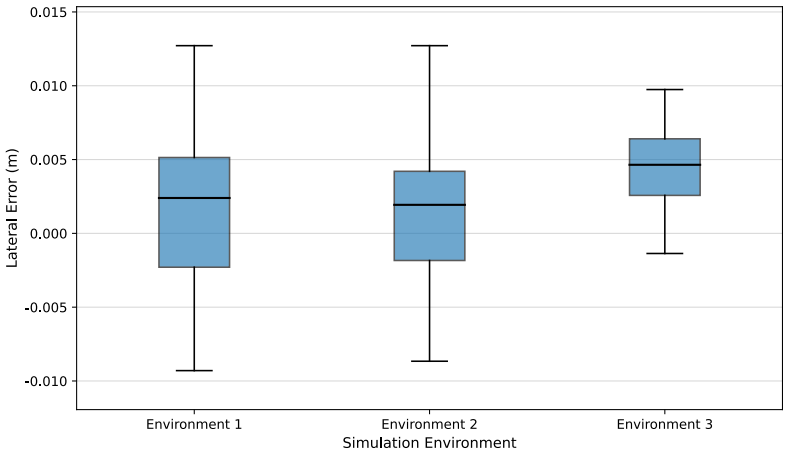


Figure 21: Distribution of lateral error observed across each environments

UWB noise had no impact on the final position error of the robot, similar to docking time as discussed earlier. The variation observed in the longitudinal and lateral errors is mainly due to the predefined tolerance limits used for docking. All recorded error values remained within these specified limits. The results also show no clear relationship between the longitudinal or lateral error and the trolley orientation or trolley location during docking.

4.3 Orientation Misalignment

Orientation misalignment refers to the yaw error between the robot and the trolley at the final docking position. It indicates how well the robot is angularly aligned with the trolley during docking. This parameter is important because successful docking not only requires accurate lateral and longitudinal positioning but also proper angular alignment. A large yaw error can cause the robot to collide with trolley. Therefore, orientation alignment is important to accurately dock the robot under trolley.

The orientation error was evaluated across the three simulation environments using the same test scenarios as the position error and docking time analyses. The mean

orientation errors were −0.001 rad, −0.002 rad, and −0.006 rad in the first, second, and third environments, respectively, with corresponding standard deviations of 0.007, 0.006, and 0.004 rad.

Overall, the orientation errors remained small and comparable across all environments. The first two environments showed very similar results, while the third environment showed a slightly larger mean error. This behavior is consistent with the lateral error results, since the same controller influences both lateral alignment and yaw correction. However, the overall orientation error remained very small in all cases, indicating stable alignment performance. The orientation error obtained in the three environments is shown in the Figure 22 and the distribution is shown in 23

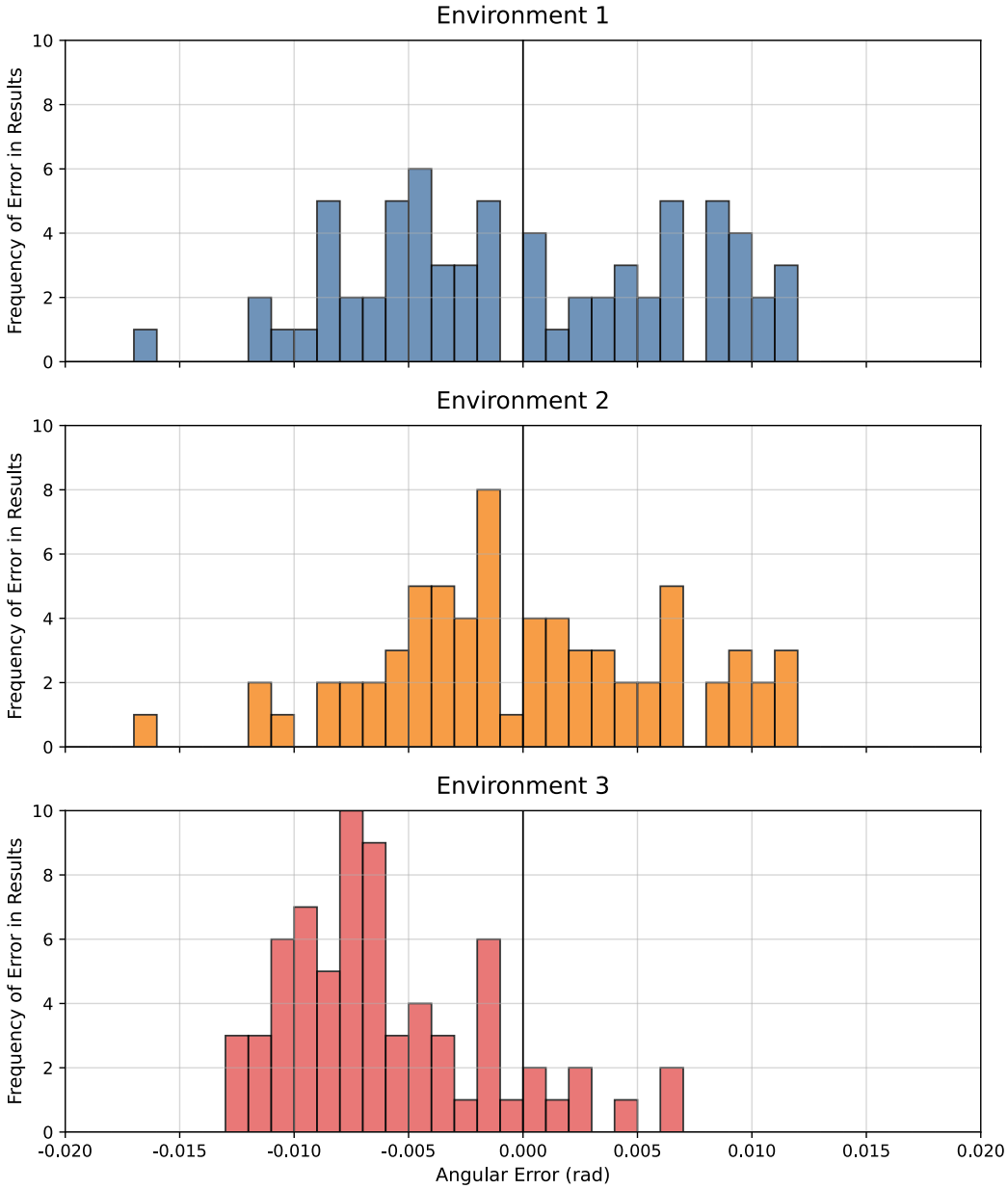


Figure 22: Histograms of orientation error observed across three environments.

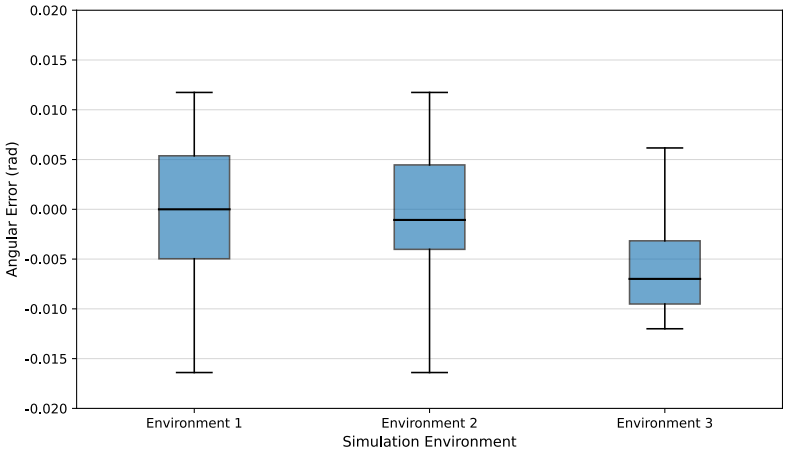


Figure 23: Distribution of orientation error observed across each environments

Similar to the position misalignment results, the orientation misalignment results also remained within the predefined limits. The misalignment in the error was not effected by the trolley alignment or the trolley position in the environment. The error values were rather random in between these limits. Similar to the previous case, the UWB noise had no impact on the orientation misalignment at the docking stage.

4.4 Average Mission Speed

The total mission speed represents the average speed of the robot over the complete docking mission. While the previous performance parameters focused mainly on the docking phase itself, the total mission speed was used to evaluate the overall motion performance of the robot throughout the entire task. This parameter provides an indication of the overall efficiency of the navigation and docking algorithm, as it considers the complete mission duration rather than only the final docking maneuver.

The total mission speed was calculated as the ratio between the total path length traveled by the robot and the total time required to complete the mission. Therefore, a higher mission speed indicates that the robot was able to complete the task more efficiently, while a lower mission speed may indicate longer travel distances, additional alignment maneuvers, obstacle avoidance behavior, or slower motion during the approach to the trolley.

The average mission speeds in the first, second, and third simulation environments were 0.240 m/s, 0.237 m/s, and 0.162 m/s, with standard deviations of 0.065 m/s, 0.054 m/s, and 0.066 m/s, respectively.

The average mission speeds in the first two environments were comparable, indicating similar overall motion behavior despite differences in obstacle layout. The first environment was more spatially constrained, while the second environment introduced challenges due to narrow aisles, multiple obstacles, and objects similar to the target trolley. In the third environment, the lower average speed can be attributed to the mixed layout, where open regions are combined with a few congested areas. When the trolley was located in constrained regions, the robot required more adjustment during tracking and alignment, increasing the variability in speed. This is reflected in

the relatively higher standard deviation observed in the third environment.

The total mission speed observed across the three simulation environment is shown below:

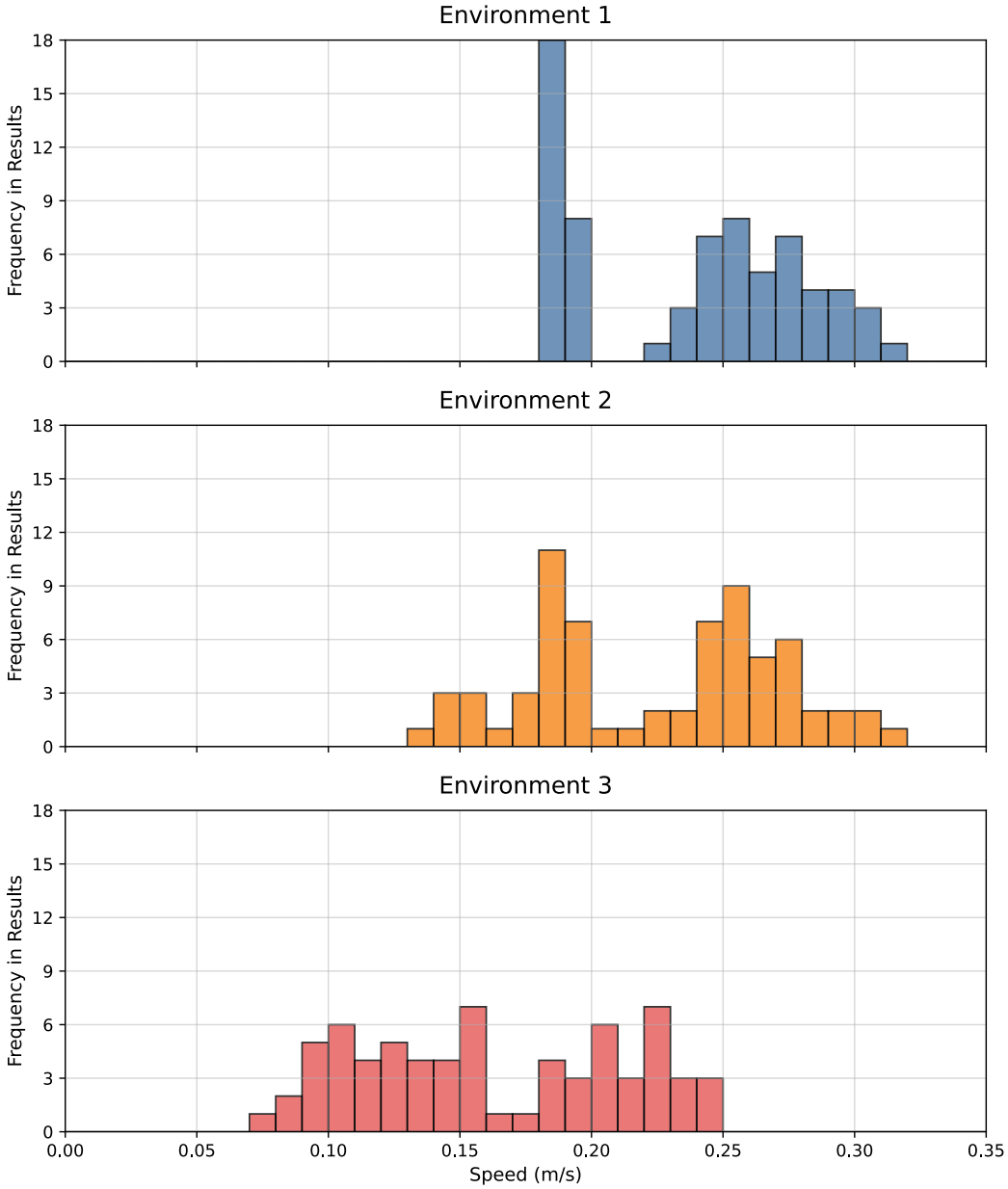


Figure 24: Histograms of average mission speed observed across three environments.

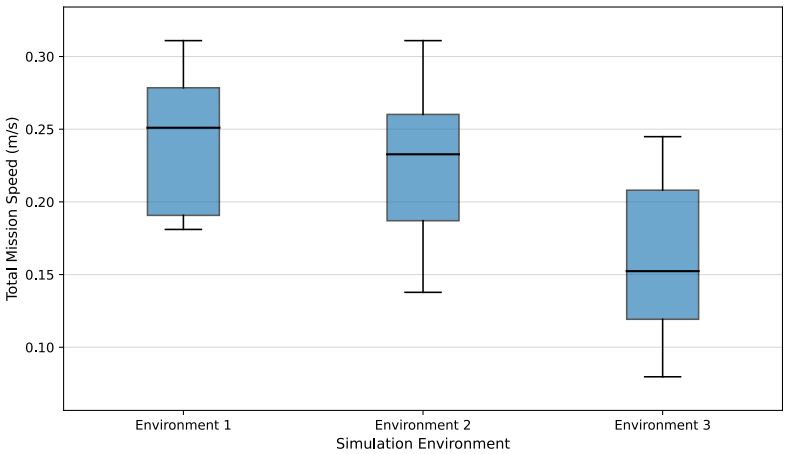


Figure 25: Distribution of average mission speed observed across each environments

The average mission speed was affected by changes in the environmental conditions. When the robot or trolley was positioned such that the path contained several obstacles, the robot required more navigation adjustments, which reduced the average mission speed. Similarly, the trolley orientation influenced the docking time, cases with larger orientation differences generally required longer docking times, which resulted in a lower average mission speed compared with cases where the robot and trolley were more closely aligned. UWB noise also had some effect on the average mission speed. A noisier UWB measurement could generate a stand-off pose farther from the trolley, increasing the time required to reach the staging location. However, this effect was limited because the additional distance was usually small, typically within about 12 m. Therefore, although UWB noise could slightly affect the overall mission speed, its impact was not significant compared with the effects of obstacle layout, trolley position, and trolley orientation. Overall, the spread observed in the histograms for the three environments can be attributed to these factors, with obstacles in the navigation path being the main contributing factor.

4.5 Failure Classification

During testing in different simulation environments, the algorithm did not complete the mission successfully in all cases. The observed failures occurred at different stages of the overall navigation and docking process and were caused by several factors.

One of the common failure cases was related to trolley detection. In some scenarios, the robot was unable to detect the trolley reliably, especially when it approached the trolley from an unfavorable direction or when the surrounding environment was highly cluttered. In such cases, the algorithm was not always able to identify the trolley features clearly enough to estimate its position and orientation accurately.

Another failure mode was observed during the docking phase, where the robot occasionally collided with the trolley. This was mainly caused by noisy lidar measurements, which affected the accuracy of trolley leg detection. Since the algorithm relies on identifying the four trolley legs to estimate the trolley pose, inaccurate or incomplete detection of these features resulted in errors in the estimated

docking position. As a result, the robot sometimes approached the trolley with incorrect alignment, leading to contact with the trolley structure.

Failures were also observed during the navigation phase after docking, when the robot was transporting the trolley toward the final goal position. In some cases, the robot got stuck while navigating with the trolley. This may have been caused by limitations in path planning, local obstacle avoidance, or the increased footprint and motion constraints of the robot-trolley system after docking.

Overall, the failure cases show that although the algorithm was able to complete the mission successfully in most scenarios, its performance was affected by environmental clutter, sensor noise, and navigation constraints during trolley transport. These observations highlight important areas for further improvement, particularly in robust trolley detection, LiDAR data filtering, collision avoidance during docking, and navigation planning after coupling with the trolley.

The failure cases discussed in this section were analyzed separately from the successful trials presented earlier. These failures represent the additional unsuccessful simulation attempts encountered while obtaining a total of 70 successful mission completions. Figure 26 illustrates the distribution of failures across the three simulation environments and classifies them into three main categories: failure in trolley detection, robot collision with the trolley, and failure during navigation after docking.

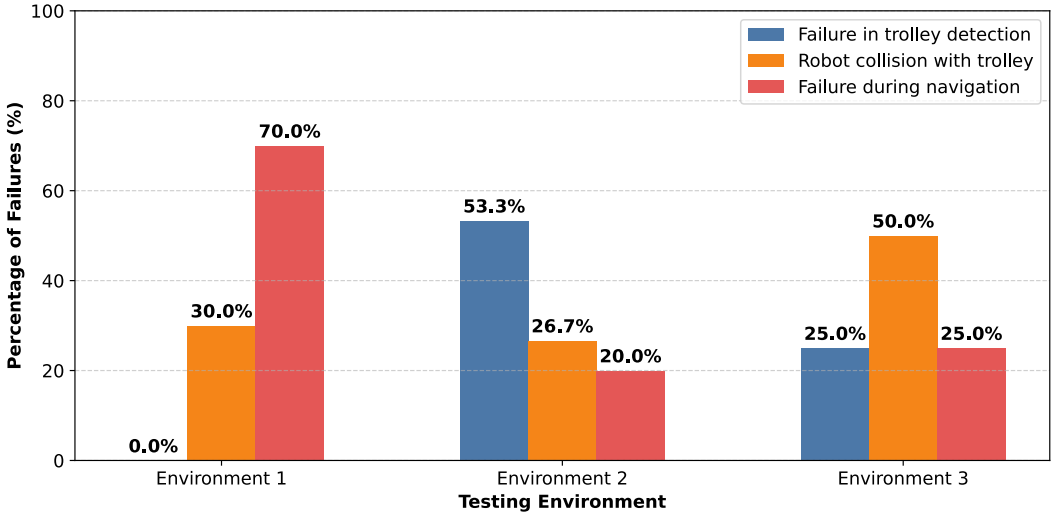


Figure 26: Comparison of failure modes across the three simulation environments.

It can be observed from the figure that the dominant failure mode varied depending on the characteristics of the environment. In Environment 1, the algorithm failed 10 times and the highest number of failures was associated with navigation after docking accounting for nearly 70 %, while 30 % cases were related to collision with the trolley, and no failures were recorded for trolley detection. This trend can be attributed mainly to the congested nature of the first environment. Although the robot was generally able to detect and dock with the trolley, navigating while transporting the trolley was more difficult because the restricted space increased the likelihood of the robot getting stuck or failing to complete the path to the final goal.

In Environment 2, the algorithm failed 15 times and the most significant failure

mode was failure in trolley detection, which accounted for nearly 53.3 % of the total failures. In comparison, 26.7 % failures were caused by collision with the trolley, and 20 % of failures occurred during navigation. This environment represented a warehouse like setting, where the trolley was sometimes positioned in narrow aisles between storage racks. Under these conditions, the robot often approached the trolley from a highly unfavorable angle, which made trolley detection more difficult. In addition, the legs of the trolley could be confused with clustered lidar returns from the rack legs, making it harder for the algorithm to distinguish the trolley structure from the surrounding environment. The collision related failures in this environment were also likely influenced by noisy lidar measurements, which reduced the accuracy of estimating the trolley leg positions and, consequently, the docking pose.

In Environment 3, the failures were distributed more evenly across the three categories and in total algorithm failed 8 times. The figure shows 25 % failures occurred in trolley detection, 50 % failures occurred due to collision with the trolley, and 25 % failures during navigation. Compared with the other environments, this environment was more open and less cluttered, which reduced the likelihood of failures being concentrated in a single category. As a result, the observed failures were more balanced and appeared to arise from general limitations of sensing and docking accuracy rather than from severe environmental constraints.

This representation of the failures across different environments highlights how changes in operating conditions affect the behavior and reliability of the overall navigation process.

5 Discussion

The results showed that the layout and configuration of the testing environment had some effect on the achieved performance. For docking time, the average value across the three environments was approximately 22.64 s. However, when the individual environments were compared, it was observed that the docking time for the first two environments was comparable and the peak values were observed at the similar locations and the docking was a bit faster in the third environment. This improvement was not caused by a major change in the algorithm's performance, but rather by the layout of the last environment. The third environment contained some congested areas with multiple obstacles, but also had a large proportion of open space. Since the trolley positions were selected using the same method as in the other environments, many docking tests occurred in wider areas where navigation was easier and less obstacle avoidance was required. As a result, the overall mean docking time in this environment was reduced. Also, for a particular environment it was observed that when the trolley was at an angled orientation, it took longer time for the robot to dock as compared to when the trolley was parallel or perpendicular, the reason is since the robot is a differential drive robot and cannot correct the lateral offset directly and thus takes longer time in case of an angled orientation.

The average docking time of 22.64 s observed in this work is considerably higher than the best value of 11.21 s reported by [17]. In the current implementation, an LQR controller is used together with a proportional controller. The forward velocity of the robot is limited by the lateral and longitudinal offset errors corrected by the LQR controller, which may contribute to the longer docking time. Therefore, it would be valuable to investigate how other control strategies perform under similar conditions. During the initial development, a simple proportional controller was implemented to correct all three pose errors of the robot. Although this controller achieved faster docking, its performance was worse in terms of correcting the lateral offset of the robot. This indicates a trade off between docking speed and final alignment accuracy. As a result, future work could explore alternative control approaches, such as feedback linearization, to evaluate whether they can provide a better balance between docking time and alignment accuracy under the same operating conditions.

It is important to consider the differences in working and operating conditions when comparing the two results. Zhao et al. [17] defined a 4.5 m approach phase, during which the robot travels at a fixed speed of 0.4 m/s while continuously correcting its position and orientation errors. Their method formulates an error equation and iteratively solves it to determine the required correction for alignment. In the final phase, visual feedback from a QR code is also used to compensate for noise in the LiDAR measurements. In our case, the alignment phase is shorter, with a length of 1.5 m, and the robot speed is not fixed. Instead, the speed is adjusted based on controller feedback, where larger misalignment results in a lower speed and smaller misalignment allows a higher speed, with the maximum speed limited to 0.4 m/s. Therefore, the shorter alignment region in our case leads to a more conservative docking behavior, resulting in a slower docking speed.

For the position error, the results obtained from all three environments were

comparable and remained within the predefined controller tolerances. In the case of longitudinal error, the error values were more evenly distributed in the first two environments, whereas in the third environment they were more concentrated within a smaller range. This reflects the behavior of the controller during the final docking phase. Since the longitudinal error represents how close the robot is to the desired docking position, the alignment phase is terminated once the robot reaches the defined tolerance limits, and the corresponding error at that instant is recorded. Similarly, the lateral error values were also comparable across all three environments and remained within the specified tolerance range. These results indicate that the controller was able to achieve consistent lateral and longitudinal positioning performance under different environmental conditions.

For the orientation error, a similar trend was observed. The yaw error values were distributed within a comparable range across the three environments and remained within the predefined tolerance limits. Since the lateral and orientation errors are coupled during the alignment process, the algorithm terminates the alignment phase once both errors satisfy their respective tolerance criteria. Therefore, the lateral and yaw errors recorded at the end of alignment represent the final values at the moment when the controller considered the robot sufficiently aligned for docking. As these errors remained within the allowed limits in all three environments, the results demonstrate the consistent performance of the controller in maintaining the required docking alignment.

The average longitudinal and lateral misalignment values obtained in this work were approximately 0.62 cm and 0.26 cm, respectively. In comparison, [17] reported a best position misalignment of approximately 1 cm in both directions. Similarly, the average orientation misalignment achieved in this work was approximately 0.003 rad, compared with approximately 0.014 rad reported by [17]. These results show that the controller achieved very accurate alignment in simulation. However, it is important to note that the experiments in this work were conducted in a simulation environment, whereas the results used for comparison in the literature were obtained using an actual robot.Real world testing with a physical LiDAR introduces additional challenges depending on the hardware used, the operating range, and the surrounding environment. Typical noise in a 2D LiDAR can range from 1 cm to 3 cm. In this work, a noise level of 1 cm was used, which is reasonable because the LiDAR was primarily used for short range and close distance detection of the trolley. Nevertheless, the actual effect of LiDAR noise and other hardware related uncertainties can only be fully evaluated after implementing and testing the system on a real robot.

For the total mission speed, the average speed in the third environment was slightly lower than in the other two environments. However, when only the docking speed is compared across the three environments, the mean docking speed was actually higher in the third environment. Therefore, the lower total mission speed cannot be directly attributed to the docking phase. This behavior was somewhat unexpected because the third environment was relatively more open than the other environments, and the robot would normally be expected to move faster due to simpler navigation. One possible explanation is that the open layout resulted in shorter paths to reach the trolley, so a significant proportion of the total mission time was spent during the alignment phase

rather than during navigation. Since the total mission speed depends on the overall distance traveled and the total mission time, a shorter traveled distance combined with the time required for alignment can result in a lower average speed.

During the evaluation, three main types of failures were observed: failure in trolley detection, collision between the robot and trolley, and failure during navigation. In the first environment, which was smaller and less cluttered as discussed earlier, no failures were observed in trolley detection. The trolley was successfully detected in all test cases. However, the limited size of the environment affected the navigation performance, especially when the robot was transporting along with the trolley. In some cases, the combined footprint of the robot and trolley was too large to complete turns in narrow areas, resulting in navigation failure. A few failures were also caused by collisions between the robot and the trolley, which can be attributed to noise in the trolley position estimated from the LiDAR data.Failures caused by robot trolley collision were observed in all three environments with comparable frequency. This suggests that the effect of LiDAR noise was not specific to a particular environment, but influenced the docking process in a similar way across all test scenarios.

In the second environment, the most significant cause of failure was unsuccessful trolley detection. This was mainly due to the warehouse like structure of the environment, where several objects had geometric features similar to the trolley legs. As a result, the detection algorithm sometimes failed to correctly identify the trolley. In addition, the narrow aisles in this environment caused the robot to approach the trolley from angles where not all trolley legs were visible to the LiDAR, which further contributed to detection failures.

In the third environment, most failures were caused by collisions between the robot and the trolley due to LiDAR noise. Although this failure mode had the highest percentage contribution in this environment, the actual number of collision cases was comparable to the other environments. This again indicates that LiDAR noise affected all environments in a similar manner. The contribution of the other failure types was lower in the third environment because it was relatively more open, reducing the chances of navigation failure and trolley detection failure caused by surrounding obstacles.

One limitation of the current implementation is that the trolley pose is locked after the trolley is detected from the LiDAR point cloud. The robot then performs the final alignment using this fixed trolley position. Continuously updating the trolley pose during alignment could improve robustness, since multiple LiDAR scans could be used to reduce the effect of noise in the detected trolley position. However, this would also introduce additional challenges. For example, the trolley legs may become temporarily occluded as the robot changes its orientation, causing the LiDAR to lose reliable detection. Therefore, continuous LiDAR-based updating would require additional logic to handle temporary loss of detection and occlusion.

A related limitation was observed in few simulation conditions, where the algorithm failed to detect the trolley after the robot reached the coarse navigation location estimated using UWB. In these cases, the robot arrived close to the trolley but with an orientation in which the trolley legs were not visible to the LiDAR. As a result, the detection algorithm failed even though the trolley was nearby. To address this issue,

the algorithm could be modified so that if no valid trolley is detected after reaching the coarse navigation location, the robot performs an in-place rotation to search for the trolley. This would reduce false detection failures caused solely by an unfavorable initial orientation of the robot.

Another limitation of the current implementation is that the algorithm was tested only in environments where only static obstacles were considered. In real factory environments, obstacles are often dynamic, and the robot may encounter moving objects, workers, or other equipment. Dynamic obstacles are less likely to affect the initial movement of the robot toward the coarse trolley location, since this phase is handled using Nav2, which is capable of dealing with dynamic obstacles. However, the main concern arises during the final alignment phase, when the robot is positioning itself with respect to the trolley. If an obstacle appears between the robot and the trolley during this phase, the current implementation may not be able to react appropriately because the robot is not continuously relying on LiDAR data for obstacle avoidance during alignment. Therefore, an additional obstacle avoidance mechanism should be incorporated into the alignment phase to improve the safety and reliability of the system in real-world environments.

One way to improve the testing is to define realistic testing scenario based on the actual use case within a factory environment. This would involve clearly specifying the docking location, the expected position and orientation of the trolley, the required final pose of the robot, and the overall navigation task to be performed. In the current experiments, the test conditions were relatively random, since the position and orientation of both the trolley and the robot were varied without being constrained by a specific factory layout or operational requirement. For example, there were no predefined conditions specifying that the trolley could only be located within a certain area or at a specific orientation. Establishing well defined testing or implementation conditions would make it possible to identify the specific problems and errors that occur in a realistic deployment scenario. Based on these findings, appropriate mitigation strategies or modifications to the algorithm could then be developed to improve the robustness of the system for that particular implementation.

One major limitation of this work is its strong dependence on the geometry of the trolley. The system is designed for a trolley with specific dimensions, which reduces its flexibility when applied to other trolley types. To use the same approach with a different trolley, the parameters related to the trolley dimensions would need to be modified. This also limits its applicability in real factory environments where multiple types of trolleys may be present. For the algorithm to work reliably, all trolleys would need to have the same dimensions otherwise, the system may fail to identify the correct trolley location. In addition, the algorithm has not been tested with trolleys of different sizes. Although it is expected that the method could be adapted to other trolley geometries by updating the relevant parameters, this still needs to be validated experimentally.

As discussed earlier, the algorithm requires a minimum clearance of approximately 1.5 m in front of the trolley for the robot to perform the alignment phase and complete the docking . This requirement can limit the applicability of the method in real world factory environments, where sufficient free space in front of the trolley may not always

be available. In cluttered or congested areas, the trolley may be surrounded by other objects, equipment, or storage units, leaving limited space for the robot to align itself properly before docking. As a result, the robot may not be able to reach the required alignment position, or it may need to perform additional navigation adjustments before docking can begin. Therefore, the required alignment distance represents an important limitation of the current approach, especially for applications in dense industrial environments where the available space is limited.

Overall, the implementation was successful in achieving accurate final alignment in simulation, which was one of the main objectives of the docking controller. The results imply that the proposed framework can improve the flexibility of autonomous intralogistics systems by reducing the dependence on predefined target locations. Instead of requiring the trolley to be placed at a fixed position, the robot is able to estimate the target location during runtime and use LiDAR-based detection to refine the trolley pose before docking. This makes the system more adaptable to factory environments where trolleys may be moved by humans or placed with small positional variations. The accurate final alignment also reduces the risk of collision between the robot, trolley, and surrounding objects during the docking process. In addition, the small final position and orientation errors indicate that the proposed method can provide reliable docking performance under the tested conditions. Therefore, the work contributes toward more flexible and robust autonomous navigation solution for factory intralogistics, especially in applications where accurate target localization and safe docking are required. The main weaknesses are related to absence of continuous obstacle avoidance during final alignment, and the lack of real world validation. Future work should focus on real robot testing, continuous trolley pose updating, search behavior when the trolley is not detected, dynamic obstacle handling during final alignment, and evaluation under factory specific operating conditions.

6 Conclusion

This thesis presented a simulation based implementation of an autonomous trolley tracking and docking system for intralogistics applications. The system was developed to address the problem of autonomous mobile robot operation in factory environments, where a trolley may initially be outside the visible sensing range of the robot. The main objective was to enable the robot to estimate the trolley location, navigation, alignment and docking with the trolley and final transport of robot along with trolley.

The proposed system used a hybrid localization approach for the trolley. When the trolley was outside the direct sensing range of the robot, UWB-based position information was used to estimate the approximate trolley location. The robot then used the Nav2 navigation stack to move from its initial position to the coarse trolley location. Once the robot reached the nearby region, LiDAR point cloud data was used to detect the trolley more accurately and estimate its relative pose. The final alignment and docking phase was performed using a combination of proportional control and an LQR controller, which corrected the longitudinal, lateral, and yaw errors between the robot and the trolley. After successful docking, Nav2 was used again to perform the final navigation task while the robot was positioned under the trolley.

The system was evaluated in three simulation environments with different layouts, dimensions, obstacle configurations, and levels of congestion. These environments were selected to represent different intralogistics operating conditions, including a laboratory environment, a warehouse environment, and a larger factory environment. The results showed that the robot was able to complete the docking task in the tested scenarios and achieved accurate final alignment with the trolley. The average longitudinal and lateral position errors across the three environments were approximately 0.62 cm and 0.26 cm, respectively, while the average orientation error was approximately 0.003 rad. These results indicate that the implemented controller was able to achieve precise final docking in simulation.

The results also showed that the environment layout and trolley orientation had a clear influence on mission performance. While the final docking accuracy remained consistently high, the docking time and average mission speed varied between environments. Congested regions and angled trolley placements required additional maneuvering during the alignment phase, which increased docking time. In contrast, open regions allowed the robot to align with the trolley more quickly. This indicates that the proposed method is effective for accurate docking, but the overall mission efficiency depends on the available maneuvering space, obstacle distribution, and initial trolley pose.

Although the simulation results were promising, the work has several limitations. The system was evaluated only in simulated environments, where the obstacles were static and the sensor behavior was simplified compared with real hardware. In a real factory environment, the robot may experience more LiDAR noise, localization errors, wheel slip, actuation delays, occlusion of trolley legs, and dynamic obstacles such as workers or other mobile equipment. In addition, the trolley pose was fixed after LiDAR based detection in the current implementation. This means that the robot did not continuously update the trolley position during final alignment. Continuous

LiDAR based pose updating could improve robustness, but it would require additional handling for temporary occlusions or cases where the trolley is lost from the LiDAR field of view.

Future work should focus on validating the system on a physical robot in a factory environment. The trolley detection algorithm should be improved to handle cases where the trolley is not immediately visible after the robot reaches the UWB based coarse location. A search behavior, such as in-place rotation, could be added to reduce detection failures caused by unfavorable robot orientation. In addition, obstacle avoidance should be incorporated into the final docking phase to improve safety when dynamic obstacles appear between the robot and the trolley. Further work should also investigate continuous trolley pose estimation during alignment and alternative control strategies to improve the balance between docking accuracy and docking time.

Overall, the thesis demonstrates that a combined UWB, LiDAR, Nav2, and LQR based controller can be used for autonomous trolley tracking and docking in simulated intralogistics environments. The implementation successfully achieved accurate final alignment under different environmental conditions, showing the feasibility of the proposed method. With further development and real world validation, the system could contribute to autonomous material handling in actual factory environments.

References

[1] Ifekanandu Chukwudi Christian. Automated material handling and operational efficiency of industrial goods manufacturing companies in south-west, nigeria. *Advanced Journal of Science, Technology and Engineering*, 6(1):12–27, 2026.

[2] International Federation of Robotics. World Robotics 2025 Report: Service Robots See Global Growth Boom. https://ifr.org/ifr-press-releases/news/service-robots-see-global-growth-boom, 2025. Accessed: 5 July 2026.

[3] Grand View Research. Logistics robot market size, share & trends analysis report, 2030. https://www.grandviewresearch.com/industry-analysis/logistics-robot-market-report, 2024.

[4] M. De Ryck, M. Versteyhe, and F. Debrouwere. Automated guided vehicle systems, state-of-the-art control algorithms and techniques. *Journal of Manufacturing Systems*, 54:152–173, 2020.

[5] Sezen Gokmen. Bridging the gap in warehouse automation: A comparative analysis of autonomous case-handling mobile robots (acmrs), amrs, and agvs. 2025.

[6] Ireneus Wior, Stefan Jerenz, and Alexander Fay. Automated transportation systems subject to interruptions in production and intralogistics-a survey and evaluation. *International Journal of Logistics Systems and Management*, 30(4):421–457, 2018.

[7] Surajkumar Goraksha Kumbhar, Rachana B. Thombare, and Amitkumar B. Salunkhe. Automated guided vehicles for small manufacturing enterprises: A review. *SAE International Journal of Materials and Manufacturing*, 11(3):253–258, 2018.

[8] Bruno Siciliano, Oussama Khatib, and Torsten Kröger. *Springer handbook of robotics*, volume 200. Springer, 2008.

[9] Jakub Pizoń, Łukasz Wójcik, Arkadiusz Gola, Łukasz Kański, and Izabela Ewa Nielsen. Autonomous mobile robots in automotive remanufacturing: A case study for intra-logistics support. *Advances in Science and Technology. Research Journal*, 18(1), 2024.

[10] 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.

[11] 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.

[12] Radim Hercik, Radek Byrtus, Rene Jaros, and Jiri Koziorek. Implementation of autonomous mobile robot in smartfactory. *Applied Sciences*, 12(17):8912, 2022.

[13] Jan Richter, David Bohlig, Andreas Nüchter, and Klaus Schilling. Advanced edge detection of apriltags for precise docking maneuvers of mobile robots. In *IFAC-PapersOnLine*, volume 55, pages 117–123, 2022.

[14] Syed Muhammad Abbas, Salman Aslam, Karsten Berns, and Abubakr Muhammad. Analysis and improvements in apriltag based state estimation. *Sensors*, 19(24):5480, 2019.

[15] Øystein Volden, Annette Stahl, and Thor I. Fossen. Vision-based positioning system for auto-docking of unmanned surface vehicles (usvs). *International Journal of Intelligent Robotics and Applications*, 6:86–103, 2022.

[16] Ruotao He, Juan Rojas, and Yisheng Guan. A 3d object detection and pose estimation pipeline using rgb-d images. In *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1527–1532, 2017.

[17] Yifan Zhao, Xiang Li, Wei Lu, and Lizhi Hu. A method for autonomous shelf recognition and docking of mobile robots based on 2d lidar. *Artificial Intelligence Technologies and Applications*, pages 104–112, 2024.

[18] Leonardo A. Jr. Fagundes, Alexandre G. Caldeira, Matheus B. Quemelli, Felipe N. Martins, and Alexandre S. Brandão. Analytical formalism for data representation and object detection with 2d lidar: Application in mobile robotics. *Sensors*, 24(7):2284, 2024.

[19] Ning Xu, Mingyang Guan, and Changyun Wen. A survey on ultra wide band based localization for mobile autonomous machines. *Journal of Automation and Intelligence*, 4(2):82–97, 2025.

[20] Zhiqiang Cao, Ran Liu, Chau Yuen, Achala Athukorala, Benny Kai Kiat Ng, Muraleetharan Mathanraj, and U-Xuan Tan. Relative localization of mobile robots with multiple ultra-wideband ranging measurements. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5857–5863, 2021.

[21] Paola Torrico Morón, Sahar Salimpour, Lei Fu, Xianjia Yu, Jorge Peña Queralta, and Tomi Westerlund. Benchmarking uwb-based infrastructure-free positioning and multi-robot relative localization: Dataset and characterization. In *2023 IEEE Sensors Applications Symposium (SAS)*, pages 1–6, 2023.

[22] Muhammad Shalihan, Zhiqiang Cao, Khattiya Pongsirijinda, Lin Guo, Billy Pik Lik Lau, Ran Liu, Chau Yuen, and U-Xuan Tan. Moving object localization

based on the fusion of ultra-wideband and lidar with a mobile robot. In *2023 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2023.

[23] Yang Song, Mingyang Guan, Wee Peng Tay, Choi Look Law, and Changyun Wen. Uwb/lidar fusion for cooperative range-only slam. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6568–6574, 2019.

[24] Yanyan Dai and Kidong Lee. Multi-sensor fusion for autonomous mobile robot docking: Integrating lidar, yolo-based apriltag detection, and depth-aided localization. *Electronics*, 14(14):2769, 2025.

[25] Yuanyuan Li, Yuan Zou, Xudong Zhang, Xiaoran Lu, Jie Fan, Zheng Zang, and Dongfang Zhang. Luot: Lidar-uwb object tracking with zero id switches and centimeter-level precision. *IEEE Internet of Things Journal*, 12(22):48945–48961, 2025.

[26] Jun Seok Oh and Min Young Kim. Regression-based docking system for autonomous mobile robots using a monocular camera and aruco markers. *Sensors*, 25(12):3742, 2025.

[27] Yuanzhe Wang, Mao Shan, Yufeng Yue, and Danwei Wang. Autonomous target docking of nonholonomic mobile robots using relative pose measurements. *IEEE Transactions on Industrial Electronics*, 68(8):7233–7243, 2021.

[28] Luis A Mateos, Wei Wang, Banti Gheneti, Fabio Duarte, Carlo Ratti, and Daniela Rus. Autonomous latching system for robotic boats. In *2019 International conference on robotics and automation (ICRA)*, pages 7933–7939. IEEE, 2019.

[29] David Fernández-Gutiérrez, Niklas Hagemann, Wei Wang, Rens Doornbusch, Joshua Jordan, Jonathan Schiphorst, Pietro Leoni, Fabio Duarte, Carlo Ratti, and Daniela Rus. Design of an autonomous latching system for surface vessels. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 1099–1105. IEEE, 2022.

[30] Shumin Feng, Yujiong Liu, Isaac Pressgrove, and Pinhas Ben-Tzvi. Autonomous alignment and docking control for a self-reconfigurable modular mobile robotic system. *Robotics*, 13(5):81, 2024.

[31] Anxing Xiao, Hao Luan, Ziqi Zhao, Yue Hong, Jieting Zhao, Weinan Chen, Jiankun Wang, and Max Q.-H. Meng. Robotic autonomous trolley collection with progressive perception and nonlinear model predictive control. In *2022 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4480–4486, 2022.

[32] Peijia Xie, Bingyi Xia, Anjun Hu, Ziqi Zhao, Lingxiao Meng, Zhirui Sun, Xuheng Gao, Jiankun Wang, and Max Q.-H. Meng. Autonomous multiple-trolley collection system with nonholonomic robots: Design, control, and implementation. *Journal of Field Robotics*, 42(1):20–36, 2025.

[33] Yuhan Pang, Bingyi Xia, Zhe Zhang, Zhirui Sun, Peijia Xie, Bike Zhu, Wenjun Xu, and Jiankun Wang. Robust docking maneuvers for autonomous trolley collection: An optimization-based visual servoing scheme, 2025.

[34] Yang Chen, Diego F. Paez-Granados, Bruno Leme, and Kenji Suzuki. Virtual landmark-based control of docking support for assistive mobility devices. *IEEE/ASME Transactions on Mechatronics*, 26(4):2007–2015, 2021.

[35] Hao-Jie Zhang, Jian-Wei Gong, Yan Jiang, Guang-Ming Xiong, and Hui-Yan Chen. An iterative linear quadratic regulator based trajectory tracking controller for wheeled mobile robot. *Journal of Zhejiang University: Science C*, 13(8):593–600, 2012.

[36] Jair Cornejo, Jose Magallanes, Eddy Denegri, and Ruth Canahuire. Trajectory tracking control of a differential wheeled mobile robot: A polar coordinates control and lqr comparison. In *2018 IEEE XXV International Conference on Electronics, Electrical Engineering and Computing (INTERCON)*, 2018.

[37] Payam Nourizadeh, Aghil Yousefi-Koma, and Moosa Ayati. Optimal control of non-holonomic wheeled mobile robot using linear quadratic gaussian controller. *Modares Mechanical Engineering*, 16(9):421–428, 2016. In Persian.

[38] Medha Chatterjee, Omar Hanif, Nihar G. Deshpande, and Alexandru Stancu. Trajectory tracking of a nonholonomic mobile robot using optimal cascade sliding mode controller. In *2020 3rd International Conference on Intelligent Robotics and Control Engineering (IRCE)*, 2020.

[39] Jinwhan Kim. *Dual Control Approach for Automatic Rover Docking*. PhD thesis, Stanford University, 2007.

[40] Yutaka Kanayama, Yoshihiko Kimura, Fumio Miyazaki, and Tetsuo Noguchi. A stable tracking control method for an autonomous mobile robot. In *Proceedings., IEEE International Conference on Robotics and Automation*, pages 384–389. IEEE, 1990.

[41] Russ Tedrake. *Underactuated Robotics*. 2025. Chapter 8: Linear Quadratic Regulators. Accessed: 2026-07-05.

[42] Neobotix GmbH. Autonomous mobile robots, 2026. Accessed: 9 June 2026.