IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Real-Time Geometric-Registration-Based Precision Localization for Autonomous Docking in Unstructured Factory Environment

Sebastian Fernando Chinchilla Gutierrez[1], Manaru Watanabe [1], Masahiro Ooyama[1], Takayuki Yamada[1], Tomoaki Yamada [1], Naoto Toshiki[1], Satsuki Yamane [1], Jose Victorio Salazar Luces[2], Ankit A. Ravankar[2] and Yasuhisa Hirata[2]

***Abstract*—In factory distribution processes, autonomous mobile robots must dock precisely at base stations. However, this task is challenging due to the dynamic and unstructured nature of factory environments, as well as the sparse point clouds caused by sensor occlusions and distance limitations. To address these challenges, we propose a geometric registration approach designed to handle sparse point clouds in changing, unstructured settings. Our method utilizes the Hough transform to detect lines, describes the point cloud based on the relationships between these lines, filters out lines that do not correspond to the geometric features of the target base station, and estimates the pose of both the station and the robot using global registration techniques. We evaluated our system in four typical factory scenarios across 72 trials. Results show the robot achieved docking accuracy within ±5.06 mm and ±1.11°, with a 100% success rate in docking and correctly identifying the target cart from surrounding objects. This represents a 70% reduction in errors and an 86% increase in success rate compared to existing methods.**

***Index Terms*—Autonomous Vehicle Navigation, Intelligent Transportation Systems, Intelligent and Flexible Manufacturing**

I. Introduction

AUTOMOTIVE factories rely on both humans and robots to distribute loads across various stages of their production line. A crucial aspect of this process is docking, where a robot transfers its load to a static base station, a passive cart, or a mobile vehicle. Successful docking requires the robot to accurately determine both its own pose and that of its target. It must then plan and execute a trajectory that ensures precise alignment and successful connection with the docking station.

Docking in factories presents several challenges related to accurately recognizing the pose of the base station relative to the robot. The first challenge is the absence of landmarks, which often occurs when the base is in a wide open area with few fixed objects or distinctive features to serve as navigation references. In such scenarios, robots relying on SLAM techniques can experience significant accuracy degradation and navigation errors [1]. The second challenge arises in environments with multiple landmarks that exhibit similar patterns, such as storage areas filled with identical carts or parallel-aligned walls and pallets. The similarity of these landmarks increases the risk of misidentifying the base station and docking at the wrong location when using pattern-matching algorithms like ICP or RANSAC registration [2]. The third challenge is the loss of landmarks in enclosed areas, where critical features are occluded, reducing the available data for navigation. For instance, LiDAR sensors may capture sparse point clouds in such settings, making it difficult to identify features and accurately estimate the pose of the docking station [3].

To address these challenges, we propose an autonomous docking system designed to recognize base stations in environments characterized by low-density point clouds and repetitive geometric features. Our contributions include the following: (1) we developed a geometric feature detector for low-density point clouds using the Hough transform. (2) we implemented a geometric filter and matcher to identify the detected features and align them with a predefined reference pattern. (3) we designed a localization system capable of estimating the pose of the base station based on these matched features. (4) we integrated our geometric localization system with a previously developed PID-pure pursuit controller [4], enabling precise and reliable autonomous docking.

II. Related Works

Docking systems capable of environmental recognition primarily rely on clustering, pure registration, or feature-based registration. Pure clustering methods identify the docking target by grouping points or features into clusters and modeling them to estimate pose. Popular clustering methods include DBSCAN (Density-based spatial clustering of applications with noise) [5], K-means [6], and Euclidean Clustering [7]. Cluster modeling typically involves fitting the identified clusters to a geometric shape using algorithms such as Random Sample Consensus (RANSAC) [8] or least squares. These methods have been applied in scenarios such as mobile robots docking to pallets in factories, orchard navigation [6], and marine robots docking with ships [7]. However, clustering methods perform optimally when the docking station is entirely within the sensor's field of view, the point cloud is dense, and there is adequate separation between the docking goal and nearby obstacles. Conversely, in cluttered or enclosed spaces, or when the point cloud is sparse, clusters may be misidentified (confused with surrounding objects) or fail to be detected altogether.

Manuscript received: December, 8th, 2024; Revised: March, 31st, 2025; Accepted: May, 16th, 2025.

This paper was recommended for publication by Editor Chao-Bo Yan upon evaluation of the Associate Editor and Reviewers' comments.

[1]S. Chinchilla, M. Watanabe, M. Ooyama, T. Yamada, T. Yamada, N. Toshiki, and S. Yamane are with the TOYOTA MOTOR EAST JAPAN, INC., Shizuoka 410-1198, Japan (sebastian.chinchilla, manaru.watanabe, masahiro.ooyama, takayuki.yamada, tomoaki.yamada, naoto.toshiki, satsuki.yamane@toyota-ej.co.jp).

[2]J. Salazar, A. Ravankar, and Y. Hirata are with the Department of Robotics, Tohoku University, Sendai 980-8579, Japan (j.salazar, ankit, hirata@srd.mech.tohoku.ac.jp).

Digital Object Identifier (DOI): see top of this page.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Registration methods estimate the pose of the base station by finding a transformation between the target and a reference. These methods can be categorized into point cloud-based and feature-based approaches. Point cloud-based registration matches all or part of a point cloud to an ideal reference. The Iterative Closest Point (ICP) algorithm and its variants, such as point-to-point and point-to-plane [9], are widely used. Applications include docking space robots [10], autonomous wheelchairs [10], and manufacturing robots [11]. However, ICP requires an initial rough pose estimate, a dense 3D point cloud rich in features such as planes and curvatures, and minimal offsets between actual and reference points [12]. Its accuracy diminishes in environments with sparse, two-dimensional, or feature-poor point clouds, and in cases of significant pose offsets [2].

Feature-based registration focuses on identifying and aligning specific features of the point cloud rather than raw points. The process involves four stages: feature extraction, feature matching, global registration, and refinement. In the first stage, geometric metrics such as normals [13], curvatures [14], density [12], shape [15], distances, angles, or visual characteristics like intensity differences [9] are used to describe the point cloud. Common feature descriptors include the Scale-Invariant Feature Transform (SIFT) [14] and ORB (Oriented FAST and Rotated BRIEF) [16]. Next, the extracted features are matched to the reference using algorithms such as Brute-Force Matcher (BFM) [17] or FLANN (Fast Library for Approximate Nearest Neighbors) [18]. Global registration then aligns the matched features, typically using RANSAC [8], which also removes outliers [19], improving robustness. Finally, ICP is applied during the refinement stage, using the transformation from RANSAC as the initial estimate. Feature-based registration has been applied in the autonomous docking of robots with shelves [20], pallets [21], trolleys [22], spacecraft [12], and marine vehicles [7]. Its main advantages include robustness to occlusion and clutter [12], but these benefits come at the cost of higher processing times due to feature matching and the need for dense 3D point clouds rich in distinctive features.

In summary, traditional clustering and registration methods cannot process sparse point clouds with low-density features in real time, particularly in the two-dimensional case [12]. Consequently, they do not perform reliably in environments with severe occlusions, enclosed areas, and sparse landmarks, especially when the surroundings change over time [23].

III. Docking Challenges in Unstructured Factories

Autonomous docking in factory environments presents significant challenges due to factors such as spatial restrictions, repetitive pattern features, dynamic landmarks, occlusions, varying lighting conditions, and stringent accuracy requirements.

1) Spatial Restrictions: Docking zones are narrow and enclosed areas with minimal clearance between carts and surrounding equipment. Consequently, there is very limited space for maneuvering and correcting the robot's trajectory. For example, Fig. 1(e) illustrates the dimensions of the docking areas, where the distance between carts and walls is no more than 100 mm. Additionally, the spacing between the rails of each cart is only 420 mm, providing a clearance of just 70 mm (±35 mm) for the robot, which has a body width of 350 mm (Fig. 3(a)). As a result, a docking accuracy of at least ±35 mm and ±4° is required, as shown in Fig. 3. Failure to meet these precision requirements results in unsuccessful docking.

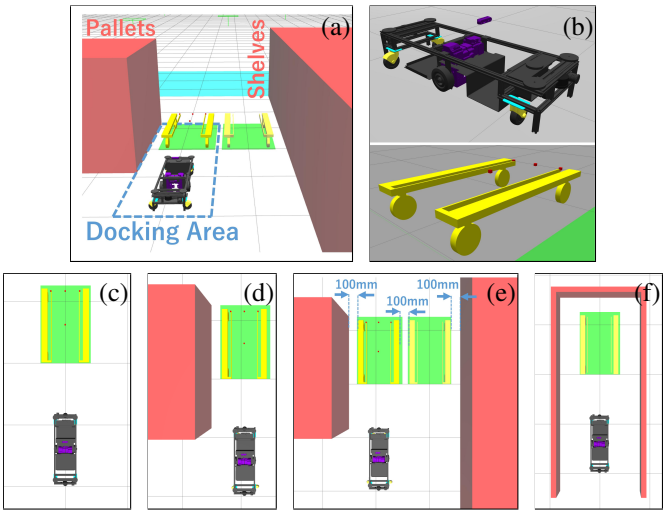

Fig. 1. Dynamic and unstructured docking area in the factory environment: (a) no walls with single cart (SC), (b) single wall with single cart (SW), (c) double wall with two carts (DC), (d) triple wall with single cart (TW).

2) Pattern Repetition: The docking area contains objects with repetitive patterns and indistinguishable features. For instance, Fig. 1(e) illustrates that within the same area, two identical carts may be positioned equidistantly from each other and from nearby pallets or shelves. Furthermore, the point cloud data detected by the LiDAR sensors mounted on the robot reveals that rails, walls, and the robot frame appear as multiple parallel straight lines and "L" patterns, with no unique distinguishing features (Fig. 2(a)). These conditions increase the risk of misidentifying the target cart and attempting to dock with a wall, an incorrect cart, or the space between carts, potentially leading to accidents.

3) Dynamic Landmarks: The factory environment is dynamic due to changing landmarks and lighting conditions. Regarding landmarks, numerous semi-static objects frequently change position or may be removed over time. For example, Fig. 1 depicts the four most common configurations of the docking area throughout the day. The red zones represent piles of pallets, totes, and racks temporarily stored around the docking area, while the green zones indicate spaces designated for carts. Depending on the factory's operational requirements, carts, pallets, and totes are frequently relocated, often without a regular schedule. This constant movement makes it challenging to establish any reliable fixed landmarks for navigation. Therefore, a system capable of tolerating changing landmarks is essential.

4) Occlusion: The visibility of the goal cart can be affected by occlusion, depending on the placement of pallets, totes, and carts, as well as the robot's pose. For example, Fig. 2 demonstrates how the point cloud of the cart rails, as captured by a LiDAR sensor, varies with the robot's pose. When the robot is properly aligned with the cart, with no positional or

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

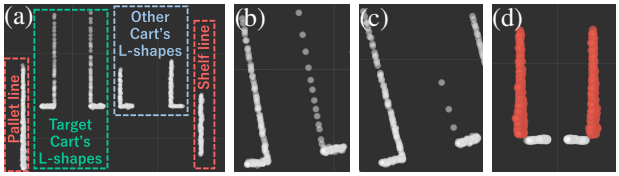

Fig. 2. Point cloud visualization of the docking area environment: (a) line patterns, (b) scarce point cloud, (c) incomplete rails, (d) occluded docking.

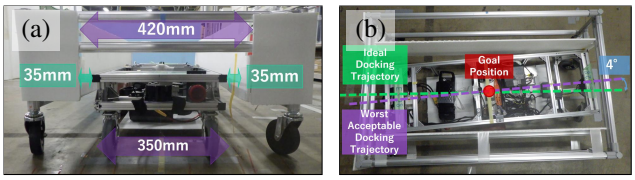

Fig. 3. Docking requirements: (a) position accuracy and (b) angle accuracy.

angular offset, a dense and uniform point cloud of the cart rails is observed. However, as the orientation offset exceeds 5°, the point cloud becomes sparse (Fig. 2(b)), and the rail is only partially visible (Fig. 2(c)). Additionally, as the robot moves beneath the cart during docking, the view of the cart rails is truncated, causing them to appear shorter than their actual dimensions (Fig. 2(d)).

5) Lighting Conditions: Due to the factory's continuous operation, the luminance in the docking area fluctuates between 7 and 90 lux [4]. Consequently, the navigation and docking system must be robust enough to operate effectively under both bright and dim lighting conditions.

IV. System Overview

A. Control Concept

The general functional blocks of our autonomous docking system are shown in Fig. 4. The system is designed to identify a passive cart with specified dimensions and autonomously dock with it, even in challenging environments characterized by dynamic changes (such as varying lighting conditions and shifting landmarks), limited space, occlusions, repetitive patterns, and sparse features.

To achieve this, we selected LiDAR as the robot's primary sensor. LiDAR operates as an active sensor, making it immune to variations in environmental illumination [24]. The data captured by the LiDAR comprises vectors that include the ranges ($\overline{range}$) and intensities ($\overline{intensity}$) of detected objects. This information is processed by the *Laser Scan to Point Cloud* block, which converts the data into a point cloud in Cartesian coordinates ($\overline{x}$ and $\overline{y}$).

The *Self-Localization* system then uses the point cloud to estimate the direct (A_{robot}^{cart}) and inverse (A_{cart}^{robot}) homogeneous

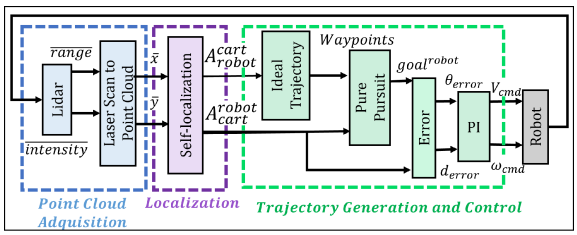

Fig. 4. General block diagram.

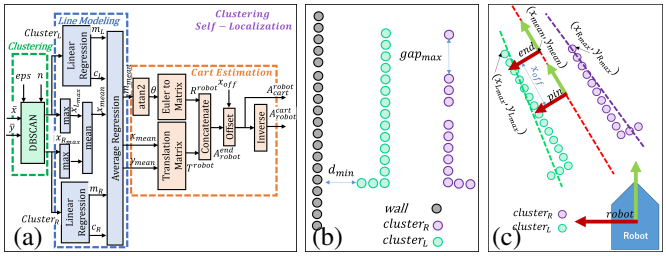

Fig. 5. CL: (a) general block diagram, (b) parameters, (c) linear regression and reference frames.

transformation matrices. These matrices define the spatial relationship between the robot's reference frame and that of the cart. This estimation is carried out using two custom algorithms developed specifically for this system: Clustering Self-Localization (CL) and Geometric Registration Self-Localization (GR). Each algorithm operates independently, and only one can be selected at a time as the self-localization system. While GR represents the novel contribution of this paper, CL is a precursor method that has been widely used in robotics [6]. Its purpose is to serve as a comparative baseline for GR.

Finally, the *Trajectory Generation and Control* block employs a PID pure-pursuit-based controller, which we previously developed for an ArUco-marker-based docking system [4]. This block takes A_{robot}^{cart} and A_{cart}^{robot} as inputs to compute the required linear (V_{cmd}) and angular (ω_{cmd}) velocity commands, ensuring the robot follows an accurate trajectory to dock successfully with the cart.

B. Clustering Self-Localization

Fig. 5(a) presents the blocks of our proposed CL system. The system's primary objective is to calculate the docking goal based on the estimated position of the cart rails. This process consists of three key steps: *Clustering*, *Line Modeling*, and *Cart Estimation*.

The first step, *Clustering* block, groups the point cloud corresponding to the cart rails into two clusters ($cluster_L$ and $cluster_R$) using the DBSCAN algorithm. It is important to mention that, depending on the angle of the robot relative to the cart, the point separation within each rail's point cloud may reach up to 30cm, as shown in Figs 5(b) and 2(c). To fully capture the rails under such conditions, the eps parameter is set to identify clusters with a maximum gap of gap_{max}. However, gap_{max} must remain smaller than the minimum distance between the rails and nearby objects (d_{min}) to avoid misclassifying surrounding objects as part of the rail.

The second step, *Line Modeling*, fits a line to each cluster using linear regression with the Least Squares method. This process calculates the slopes (m_L and m_R) and intercepts (c_L and c_R) of the lines for each cluster. These values are averaged in the *Average Regression* block to compute the mean slope (m_{mean}) and intercept (c_{mean}). As depicted in Fig. 5(c), the clusters form mirrored "L" shapes, which results in fitted lines that are not parallel. Nevertheless, since both rails are dimensionally identical, averaging aligns the resulting line with the parallel inner edges of the rails, ensuring it is centered.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

In the third step, *Cart Estimation*, the pose of the cart's endpoint is calculated relative to the robot (A_{robot}^{end}) and vice versa (A_{end}^{robot}). The cart's orientation (θ) is derived from the slope (m_{mean}) and converted into a rotation matrix (R^{robot}) using the *Euler to Matrix* block. The vertical component (x_{mean}) of the translation matrix (T^{robot}) is determined by averaging the vertical coordinates of the farthest points from each cluster relative to the robot's reference frame ($x_{R_{max}}$ and $x_{L_{max}}$). The horizontal component (y_{mean}) is calculated using the mean line equation, with x_{mean} as input.

Finally, the docking goal (*pin*) is computed by offsetting the cart's endpoint reference frame by a distance x_{off}, corresponding to the rail center. The direct and inverse homogeneous transformation matrices between the robot and the cart (A_{robot}^{cart} and A_{cart}^{robot}) are also determined, completing the localization process.

C. Geometric-Registration Self-Localization

Fig. 6(a) presents the general block diagram of our proposed GR system. The system aims to align an ideal reference pattern of the cart rails with the actual point cloud to estimate the pose of both the cart and the robot. Designed for deployment in real factory environments, the system meets key requirements, including low processing power, real-time control loops, and handling of 2D point clouds. It is also robust to sparse and occluded point clouds.

1) Point Cloud Pre-processing: The first block of the GR is the *Point Cloud Pre-Processing* (PCPP) module, which performs two primary functions. First, it generates a *Reference Pattern* that shows the expected configuration of the cart rail's point cloud relative to the robot at the goal position once docking is complete. As shown in Fig. 7(a), this pattern consists of two mirrored "L" shapes, defined by the parameters: width (w), length (l), and leg length (t). In addition, the docking goal position is set at the rail center (Fig. 7(a)) but can be adjusted by specifying non-zero values for vertical (x_{off}) and horizontal (y_{off}) offsets. Second, the PCPP module converts the 2D Cartesian point cloud ($\overline{x}$ and $\overline{y}$) into pixel coordinates, plotting them as reference image (img_{ref}) and actual image (img_{act}).

Next, *Pattern Matching* incorporates a feature descriptor and a geometric matcher. The feature descriptor, HFD (Hough-Transform-based Feature Detector), identifies key features and their relationships to construct geometric descriptors ($\overline{des}_{ref}$ and $\overline{des}_{act}$) and key points ($\overline{key}_{ref}$ and $\overline{key}_{act}$) for the reference and actual point clouds, respectively. The *Geometric Matcher* then pairs these features and descriptors, producing matched key points ($\overline{key}_{ref}$ and $\overline{key}_{act}$) for further processing.

2) HFD: Figure. 6(c) details the components of our HFD block. In summary, the HFD employs the *Hough Transform* (HT) algorithm to detect lines in the point cloud, describes the relationships between detected features using the *Descriptor Combinations* block, and filters the relevant features with the *Geometric Filter*.

- HT: To address challenges posed by occluded and sparse rail point clouds, the HT is configured to detect lines with a minimum length (h_{min}) and a maximum allowable gap between points (gap_{max}). These parameters were heuristically determined as 40% and 30% of l (the rail length), respectively, to accommodate the worst-case starting position and orientation during docking (± 50 mm, $\pm 10°$). Fig. 7(b) shows the lines identified by the HT in the point cloud for the DC scenario (Fig. 1(e)). Detected lines are categorized as red and green for actual and reference lines, respectively, while black lines are rejected by the HT as they do not meet the criteria.
- Descriptor Combinations: This block generates descriptors ($\overline{des}_0$) and key points ($\overline{key}_0$) by analyzing all possible pairs of lines. Each pair ($line[i]$, $line[j]$) is described using three parameters: the perpendicular distances from the longer to the shorter line at their starting (d_{start}) and ending (d_{end}) points, and the angular difference between their slopes (θ_{diff}). As shown in Fig. 7(c), θ_{diff} is invariant to translation and remains close to zero because the cart rails are rigid and perpendicular. The perpendicular distances (d_{start}, d_{end}) remain nearly invariant for small angular offsets, ensuring robust descriptors.
- Complete Rails: This block generates raw key points ($\overline{key}_0$) and addresses occlusions caused by visibility loss, as seen in Figs. 2(c) and 2(d). Occlusions can distort the apparent size of the rails, leading to scaling and shape errors during the affine transformation in the Cart Estimation step. To mitigate these issues, reference frames are set at the endpoints of each line, aligning their x-axes with the respective lines. The left and right lines are distinguished based on their relative positions and orientations, with the farther reference frame (e.g., *L* in Fig. 7(d)) selected as the primary frame. The secondary frame (*R*) is shifted so its x-component is zero relative to the primary frame. This ensures the primary frame better represents the end of the rail. Line lengths are then extended to match the expected rail length (l), and endpoints are updated accordingly. Finally, the start and end points of the corrected lines are selected as key points ($key[i]$, $key[j]$).
- Geometric Filter: This block filters line pairs to ensure the system operates in real-time. Line pairs with descriptors whose perpendicular distances (d_{start}, d_{end}) exceed or fall below the rail width tolerances (w_{min}, w_{max}) are discarded.

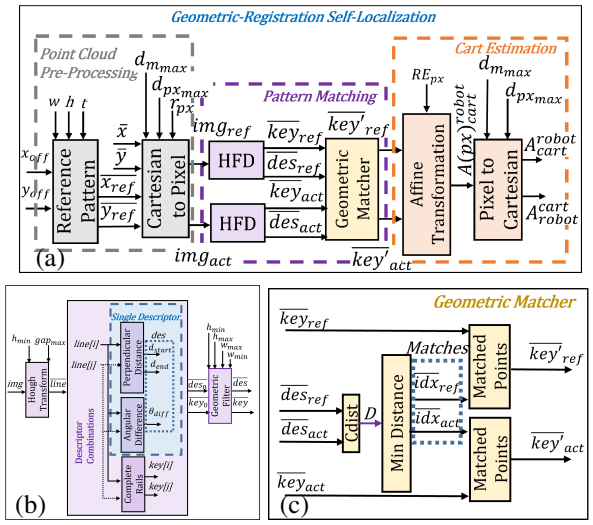

Fig. 6. GR: (a) general concept, (b) HFD, (c) Geometric Matcher.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Additionally, key points representing rail lengths outside the allowable range (h_{min}, h_{max}) are removed. The resulting filtered descriptors and key points are denoted as $\overline{des}$ and $\overline{key}$, respectively.

- Geometric Matcher: The matcher pairs the descriptors from the reference and actual point clouds to generate matched key points ($\overline{key'}_{ref}$, $\overline{key'}_{act}$). As shown in Fig. 6(c), a matrix (D) containing the Euclidean distances between all combinations of actual and reference descriptor pairs is calculated using the *cdist* block. The *Min Distance* block selects the pairs with the smallest distances, returning their indexes. These indexes are then used by the *Matched Points* block to output the corresponding matched key points. The final matched key points represent the start and end points of the filtered lines, enabling accurate alignment for docking.

3) Cart Estimation: This block (Fig. 6(a)) estimates the homogeneous transformation matrix between the robot and cart reference frames, as well as its inverse, to determine their respective poses. This is achieved through a global registration process using a RANSAC-based *Affine Transformation* applied to the matched key points. RANSAC was selected for its robustness to outliers, ensuring reliable pose estimation even with noisy data. The *Affine Transformation* was chosen for its ability to compute rigid transformations without introducing scaling or distortions, which aligns with the rigidity of the cart rails and the reference pattern. Since the *Affine Transformation* generates the transformation matrix in pixel space ($A(px)^{robot}_{cart}$), the *Pixel to Cartesian* block converts this matrix into Cartesian space, yielding the direct (A^{robot}_{cart}) and inverse (A^{cart}_{robot}) transformation matrices required for accurate localization.

D. System Integration

The robot used in our experiments is shown in Fig. 8(a). It is a replica of the robot deployed in our factories for docking with carts, such as the one depicted in Fig. 8(c). The robot features a differential drive system supported by four caster wheels for stability. Its components, illustrated in Fig. 8(b), include an Orientalmotor BLV-R-type motor driver, two RPLiDARs as the primary sensors, a NUC 11 i7 Intel Processor with integrated Intel Iris Xe Graphics as the onboard computer, a wireless router for communication, and a 24V battery with linear regulators for power. The robot operates on Ubuntu 22.04 as the operating system, with Robot Operating System 2 (ROS2) managing the system's integration and command communication. Additionally, Python 3 is used for programming the functional blocks of the system.

V. Experiments

A. Experimental Setup for the Factory Replica

To evaluate the effectiveness of our proposed localization systems (CL and GR) in realistic factory setups, specifically those of TOYOTA MOTOR EAST JAPAN, INC. factories [4], we conducted real-world tests in a laboratory replica of the factory's docking areas. These setups present challenging conditions, including dynamic landmarks, pattern repetition, and occlusion.

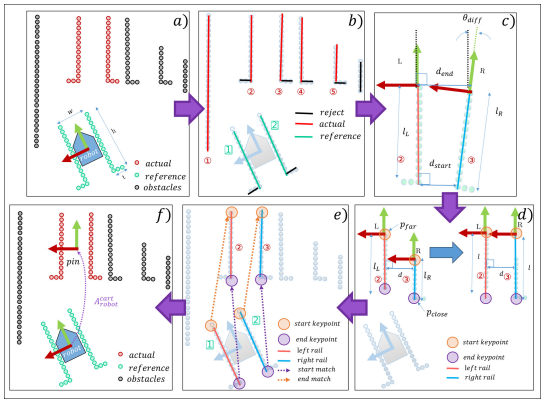

Fig. 7. Steps of the geometric-registration-based self-localization.

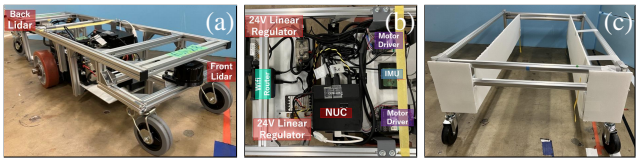

Fig. 8. Docking hardware: (a) ES28 robot, (b) drivers, and (c) cart.

Dynamic landmarks refer to semi-static material handling equipment (e.g., pallets and racks) that are constantly relocated due to layout changes. This equipment generates point clouds with repetitive L-shapes and straight lines when observed by LiDAR sensors, so we model them with styrofoam boards, which we refer to as walls.

Additionally, the docking areas contain varying amounts of obstacles. Extreme cases include wide, open areas with no landmarks and narrow, cluttered spaces where landmarks occlude the docking station. To simplify the experimental setup, we model these cases in four environmental conditions: (1) Single Cart (SC): an area with no obstacles and a single cart; (2) Wall and Cart (WC): a single wall and cart; (3) Double Cart (DC): a single wall with two carts; and (4) Three Walls and Cart (TW): three walls and a single cart. These scenarios, illustrated in Fig. 9, represent typical setups found in docking areas.

The robot's initial conditions were sampled to mimic poses typically observed when the SLAM system transitions to the docking process. Specifically, the robot started 1360 mm vertically away from the cart chassis, with three combinations of horizontal and angular offsets: 50 mm and 10°, -50 mm and -10°, and 0 mm and 0°. These values reflect the accuracy range of our SLAM system (±50 mm, ±10°). Additionally, the stop point was positioned 70 cm away from the docking goal (the "pin"), corresponding to the distance at which the cart's point clouds become occluded and the system switches to odometry mode. Each combination of initial position and angle was tested three times across all four scenarios, resulting in a total of 72 docking attempts.

The primary metrics for assessing system performance were docking success and docking accuracy. Successful docking was defined as the robot autonomously entering the cart without colliding with its body and arriving at the designated stop point. Conversely, a failed docking occurred when the robot required manual intervention or automatically stopped

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

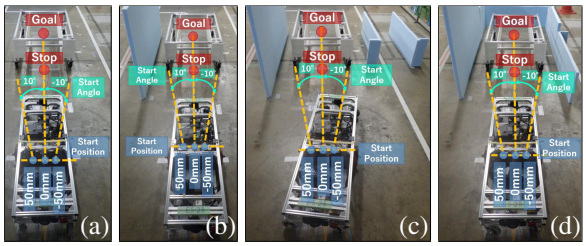

Fig. 9. Experimental scenarios: (a) SC, (b) SW,(c) DC, and (d) TW.

prematurely. Manual stops were triggered to avoid imminent collisions with the cart or surrounding objects. Automatic stops occurred due to misrecognition of the cart rails, leading the robot to incorrectly conclude that it had reached its goal. After each docking attempt, we verified whether the system met the required precision of ±35 mm and ±4° to consider the docking successful. This was done by measuring the robot's pose relative to the stop point reference frame, including x and y distances and the yaw angle, which collectively defined the docking accuracy error.

B. Results of the Docking Tests

Figures 10(a) and 10(b) illustrate the position and orientation trajectories recorded in the replica environment for a sample starting at 10° and 50 mm across the four scenarios. In all scenarios, the GR algorithm produces consistent trajectories that converge to the desired goal position and orientation. In contrast, the CL approach exhibits significant trajectory variation, with most attempts failing to reach the goal. Success rates (Fig. 10(c)) further confirm the GR algorithm's reliability, achieving a 100% success rate under all tested conditions. In comparison, the CL approach successfully docks in only 56% of trials for the SC scenario and fails entirely in the other three scenarios.

To evaluate the GR algorithm's effectiveness in reducing docking errors, we conducted a Two-way ANOVA (Analysis of Variance) [25] on horizontal (y_{err}), vertical (x_{err}), and angular (yaw_{err}) errors. The within-subject factors were the tested algorithms (GR and CL) and the experimental scenarios (NW, SC, DC, TW). Prior to analysis, we confirmed the data met the assumptions of normality (Shapiro-Wilk's test [25] , p>0.05) and homogeneity of variance (Levene's test [25] , p>0.05).

Initially, a significant two-way interaction (p<0.001) was observed between the scenario and algorithm for y_{err}, x_{err}, and yaw_{err}. Subsequently, a simple main effects analysis of the algorithm's impact revealed significant differences (p<0.01) in y_{err} for the SC, SW, and DC scenarios. For x_{err}, significant differences (p<0.0001) were found in the SW, DC, and TW scenarios, while for yaw_{err}, significant differences (p<0.001) were observed in the SW and DC scenarios. An additional simple main effects analysis of the scenarios indicated significant differences (p<0.01) between GR and CL for y_{err}, x_{err}, and yaw_{err} across scenarios.

To further explore these differences, we performed pairwise multiple comparisons using Shaffer's modified sequentially rejective Bonferroni correction [25]. This analysis revealed significant differences in y_{err}, x_{err}, and yaw_{err} both between

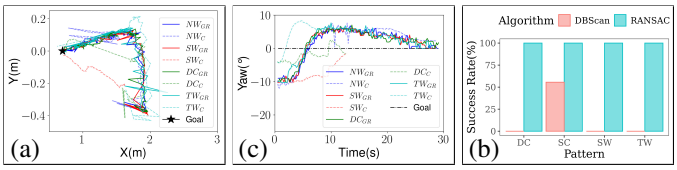

Fig. 10. Trajectory results of the docking tests for a single trial when starting at 50mm and 10° at each scenario: (a) position trajectories, (b) yaw angle trajectories. While the solid lines represent the results with the GR approach, the dashed ones report the trajectories obtained with the CL scheme. (c) represents the success rate of the docking at each scenario with both schemes for the entire 72 trials of the experiment.

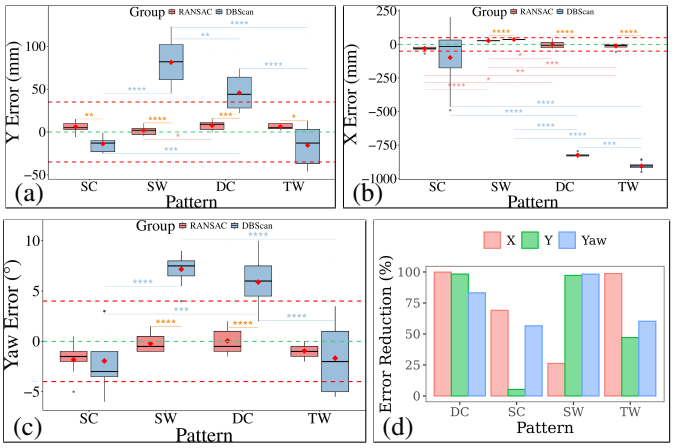

Fig. 11. Significant results from the pairwise analysis of the Two-way ANOVA: (a) Y error, (b) X error, (c) Yaw error, and (d) Error reduction. The interquartile range (IQR) is represented by pink and light blue boxes for the CL and GR algorithms in each scenario. While the red spades represent the mean error, their central horizontal lines mark the median. Gray dots denote outliers. Significant differences are indicated with stars, having the following p-values: p<0.0001 (****), p<0.001 (***), p<0.01 (**), and p<0.05 (*). Yellow stars indicate significant differences between the CL and GR at each scenario. Pink and blue stars indicate significant differences within the CL and GR algorithms, respectively. Red dashed lines show the maximum and minimum acceptable errors (target accuracy). Green dashed lines indicate the desired error.

the algorithms at various stages and between the scenarios within each algorithm.

Finally, we compared the error reduction achieved by the GR algorithm relative to the CL approach (Fig. 11(d)). The results demonstrate that the GR algorithm consistently outperformed CL across all scenarios, yielding lower errors. On average, the GR algorithm reduced y_{err}, x_{err}, and yaw_{err} by 74.6%, 73.5%, and 62.0%, respectively, for an overall error reduction of 70.0%.

In terms of algorithm efficiency, the system requires 155.0 ms (GR) and 30.0 ms (CL) to detect the docking station in the point cloud and compute a velocity command for the motor drivers. This processing time falls within our target real-time control loop frequency of 5 Hz (200 ms).

VI. Discussion

The significant interaction identified through the Two-way ANOVA analysis highlights that the docking error varies based on the combination of scenario and algorithm used. This behavior is further substantiated by the simple main effects and pairwise analyses.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

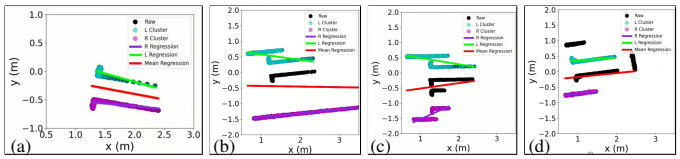

Fig. 12. Misrecognition with CL at (a) SC, (b) SW, (c) DC, and (d) TW.

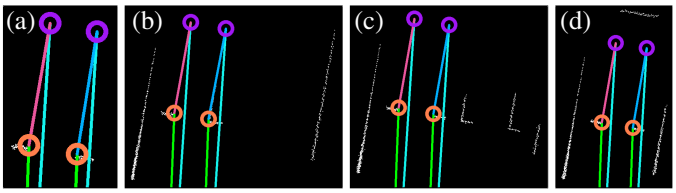

Fig. 13. Correct recognition of the cart rails with GR at each scenario: (a) SC, (b) SW, (c) DC, and (d) TW. The white points represent the points of the point cloud that were rejected as part of the cart. The pink and light blue lines designate the recognized left and right cart rails, respectively. The orange and purple circles denote the start and end points of each detected line, respectively.

A. Significant differences within the CL algorithm

The primary cause of docking failures with the CL algorithm is the misrecognition of environmental objects as the cart's rails, as depicted in Fig. 12. Specifically, the shape of the cluster often does not align with the rail's shape, leading to linear regressions with substantial positional and orientation offsets. Two types of misrecognition are observed: (1) Cluster Merging: This occurs when walls and rails are grouped into a single cluster. For instance, in the DC scenario (Fig. 12(c)), the right rail merges with the wall and part of the second cart, while in the SW scenario (Fig. 12(b)), the left rail merges with the wall. (2) Cluster Swapping: Here, the actual rail is ignored, and a surrounding object is selected instead. For example, in the SW (Fig. 12(b)) and TW (Fig. 12(d)) scenarios, the right wall is mistakenly identified as a rail, while the true rail is overlooked. These misrecognitions stem from the limitations of DBSCAN in this context.

- Cluster Merging: This issue arises from a trade-off between the distance between objects and the spacing within the rail's point cloud. To fully capture the rail, an eps distance of 30 cm is required. Smaller values result in short, slanted linear regressions with significant pose errors. However, while the rail's long section is 35 mm away from the wall, its shorter section can be as close as 10 cm, causing the wall and rail to merge into a single cluster.
- Cluster Swapping: This occurs due to the simplicity of the cluster selection process. DBSCAN outputs all detected clusters, and the one with the highest point density is selected, under the assumption that it corresponds to the rails due to their proximity to the robot. However, when the docking starts with a horizontal or angular offset, walls or adjacent carts may have denser point clouds, leading to their incorrect selection over the rail's point cloud.

  A further limitation of the CL algorithm is the modeling of the L-shaped rail as a straight line. For instance, in the SC scenario (Fig. 12), even when clusters are correctly identified, the estimated linear regression is not parallel to the rail's long section, resulting in orientation errors. Ideally, the errors in each rail's orientation should cancel each other out due to the rails' identical dimensions, producing an accurate average regression. However, when part of one rail is occluded due to the robot's pose, the perceived shape and orientation of the rails differ. This prevents error cancellation, leading to an estimation error that contributes to a 44% failure rate.

B. Significant differences within the GR algorithm

The results from the simple main and pairwise analyses, show that the GR algorithm consistently met the minimum accuracy requirement in all trials. However, significant differences in error types were observed depending on the metric evaluated. This consistency arises from the robust performance of the recognition algorithm. Specifically, the HFD accurately and independently detects both walls and cart rails, while the *Geometric Filter* eliminates lines that do not match the rail dimensions or their perpendicular distance and orientation relationships. The *Geometric Matcher* reliably identifies the rail endpoints, and RANSAC effectively estimates the homogeneous transformation matrices. Together, these processes ensure accurate pose estimations, preventing misrecognitions and maintaining stable yaw_{err} values across all scenarios.

In contrast, significant differences in y_{err} were observed across scenarios during the pairwise analysis, driven by three underlying factors:

- Noise effect: Currently, the GR algorithm uses raw pose estimates without applying noise filters. Future implementations will incorporate an extended or unscented Kalman filter [4] to reduce noise and improve accuracy.
- Point Cloud Density: The density of points in the rail point cloud varies. At the rail endpoints, the density decreases because these points are farther from the LiDAR sensor. Additionally, when the robot is positioned at an angle, occlusions further reduce point cloud density. These factors can cause the HFD to detect shorter rails. While the algorithm compensates for occlusion by extending the detected rail length, it assumes the detected endpoints are accurate. This assumption leads to fluctuations in observed y-coordinates as the robot moves, contributing to variations in y_{err}.
- Dynamic Occlusion: Despite fluctuations in point density and visibility at the endpoints, the system is robust enough to maintain target accuracy. The GR algorithm successfully compensates for occlusion, particularly as the robot transitions into the docking phase, by rectifying distortions caused by reduced visibility when entering the cart.

Overall, while y_{err} variations exist, the GR algorithm remains effective, achieving the required accuracy and docking successfully in all tested scenarios and trials. The system's resilience to noise and occlusion ensures consistent performance, and planned enhancements, such as noise filtering, are expected to further improve accuracy and robustness.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

C. Significant Differences Between the Algorithms

The significant differences identified in the simple main analysis on the effect of the algorithm indicate that the mean error varies notably between the GR and CL algorithms across most scenarios. This finding is reinforced by the pairwise analysis, which revealed significant differences in y_{err} for all scenarios, in yaw_{err} for SW and DC, and in x_{err} for SW, DC, and TW. Furthermore, when comparing the magnitude of the errors, the GR algorithm consistently reduces all three error types (y_{err}, x_{err}, and yaw_{err}) significantly, as illustrated in Fig. 11(d). These results strongly suggest that the GR algorithm achieves superior docking accuracy compared to the CL method.

The improved performance of the GR algorithm stems from its ability to accurately recognize the goal rails while selectively rejecting point clouds from walls and surrounding carts that do not match the rail geometry. Conversely, the higher errors observed with the CL algorithm are primarily due to its inability to reliably distinguish between cart rails, walls, or adjacent carts. This limitation leads to misrecognition, undermining docking accuracy and consistency. Although the enhanced recognition capabilities of the GR approach increase the system's processing time compared to the CL method, the robot docks successfully in real time within our target control loop frequency of 5 Hz.

Finally, the GR algorithm has two potential limitations: extreme occlusions (e.g., when the robot or an operator completely blocks the view of the docking station, preventing the detection of geometric features) and extreme lighting conditions (e.g., when direct, intense sunlight exposes the LiDAR sensor, reducing its accuracy [24]). However, these situations are uncommon, as docking stations are generally located indoors and subject to safety regulations.

VII. Conclusion

In this paper, we proposed two algorithms for real-time docking in unstructured and dynamic factory environments: GR and CL. First, our findings indicate that the CL method can estimate the poses of the robot and docking station effectively in environments without surrounding objects. However, when walls or additional carts are present, misrecognitions occur, significantly degrading pose estimation accuracy and leading to docking failures in most cases. In contrast, our results demonstrate that the proposed GR approach is robust to unstructured and dynamic environments. It consistently improved docking accuracy and success rates across all tested scenarios. This performance is attributed to the effectiveness of the proposed HFD, which accurately identified rails and walls, modeled them as lines, and described them using their start and end points. This precise feature detection and modeling enabled the GR algorithm to reliably dock in complex environments.

References

[1] T. Ran, L. Yuan, J. Zhang, D. Tang, and L. He, "Rs-slam: A robust semantic slam in dynamic environments based on rgb-d sensor," *IEEE Sensors Journal*, vol. 21, no. 18, pp. 20 657–20 664, 2021.

[2] H. Sun, S. Wang, J. Meng, Y. Liu, and Y. Xie, "Accurate pose tracking of mobile robot using entropy-based trimicp in dynamic environment," in *IECON 2022–48th Annual Conference of the IEEE Industrial Electronics Society*. IEEE, 2022, pp. 1–6.

[3] Y. Tang, C. Zhao, J. Wang, C. Zhang, Q. Sun, W. X. Zheng, W. Du, F. Qian, and J. Kurths, "Perception and navigation in autonomous systems in the era of learning: A survey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 12, pp. 9604–9624, 2022.

[4] S. Chinchilla, T. Saito, R. Oikawa, T. Yamada, N. Toshiki, S. Yamane, J. Salazar, A. A. Ravankar, and Y. Hirata, "Real-time marker-based monocular autonomous docking in semi-unstructured indoor environments," in *2024 IEEE/SICE International Symposium on System Integration (SII)*. IEEE, 2024, pp. 1561–1568.

[5] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *kdd*, vol. 96, no. 34, 1996, pp. 226–231.

[6] A. Jiang and T. Ahamed, "Navigation of an autonomous spraying robot for orchard operations using lidar for tree trunk detection," *Sensors*, vol. 23, no. 10, p. 4808, 2023.

[7] L. He, Y. Chen, and J. Zhao, "Automatic docking recognition and location algorithm of port oil loading arm based on 3d laser point cloud," in *2020 IEEE International Conference on Mechatronics and Automation (ICMA)*. IEEE, 2020, pp. 615–620.

[8] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[9] C. Costa, H. Sobreira, A. Sousa, and G. Veiga, "Robust 3/6 dof self-localization system with selective map update for mobile robot platforms," *Robotics and Autonomous Systems*, vol. 76, pp. 113–140, 2016.

[10] F. Aghili, "Robust 3d vision for autonomous robots," *arXiv preprint arXiv:2208.12925*, 2022.

[11] A. Yilmaz, E. Sumer, and H. Temeltas, "A precise scan matching based localization method for an autonomously guided vehicle in smart factories," *Robotics and Computer-Integrated Manufacturing*, vol. 75, p. 102302, 2022.

[12] Y. He, J. Yang, K. Xiao, C. Sun, and J. Chen, "Pose tracking of spacecraft based on point cloud dca features," *IEEE Sensors Journal*, vol. 22, no. 6, pp. 5834–5843, 2022.

[13] Y. Zhong, "Intrinsic shape signatures: A shape descriptor for 3d object recognition," in *2009 IEEE 12th international conference on computer vision workshops, ICCV Workshops*. IEEE, 2009, pp. 689–696.

[14] D. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, pp. 91–110, 2004.

[15] W. Wohlkinger and M. Vincze, "Ensemble of shape functions for 3d object classification," in *2011 IEEE international conference on robotics and biomimetics*. IEEE, 2011, pp. 2987–2992.

[16] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *2011 International conference on computer vision*. Ieee, 2011, pp. 2564–2571.

[17] F. K. Noble, "Comparison of opencv's feature detectors and feature matchers," in *2016 23rd International Conference on Mechatronics and Machine Vision in Practice (M2VIP)*. IEEE, 2016, pp. 1–6.

[18] M. Muja and D. Lowe, "Flann-fast library for approximate nearest neighbors user manual," *Computer Science Department, University of British Columbia, Vancouver, BC, Canada*, vol. 5, no. 6, 2009.

[19] P. Xie, B. Xia, A. Hu, Z. Zhao, L. Meng, Z. Sun, X. Gao, J. Wang, and M. Q.-H. Meng, "Autonomous multiple-trolley collection system with nonholonomic robots: Design, control, and implementation," *Journal of Field Robotics*, 2024.

[20] Y. Zhao, X. Li, W. Lu, and L. Hu, "A method for autonomous shelf recognition and docking of mobile robots based on 2d lidar," in *Artificial Intelligence Technologies and Applications*. IOS Press, 2024, pp. 104–112.

[21] Z. He, Y. Wang, and H. Yu, "Feature-to-feature based laser scan matching in polar coordinates with application to pallet recognition," *Procedia Engineering*, vol. 15, pp. 4800–4804, 2011.

[22] A. Xiao, H. Luan, Z. Zhao, Y. Hong, J. Zhao, W. Chen, J. Wang, and M. Q.-H. Meng, "Robotic autonomous trolley collection with progressive perception and nonlinear model predictive control," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 4480–4486.

[23] P. Li, R. Wang, Y. Wang, and W. Tao, "Evaluation of the icp algorithm in 3d point cloud registration," *IEEE access*, vol. 8, pp. 68 030–68 048, 2020.

[24] K. Yoshioka, "A tutorial and review of automobile direct tof lidar socs: Evolution of next-generation lidars," *IEICE Transactions on Electronics*, vol. 105, no. 10, pp. 534–543, 2022.

[25] M. H. Kutner, C. J. Nachtsheim, J. Neter, and W. Li, *Applied linear statistical models*. McGraw-hill, 2005.