BULLETIN OF THE POLISH ACADEMY OF SCIENCES
TECHNICAL SCIENCES, Vol. 74(6), 2026, Article number: 2514
DOI: 10.24425/bpasts.2026.2514

A terminal navigation approach for autonomous docking with transport trolleys using laser scanning

Jarosław SZREK*, Konrad SIUDZIŃSKI, and Artur MURASZKOWSKI

Wrocław University of Science and Technology, Faculty of Mechanical Engineering, Wrocław, Poland

Abstract. This study presents a low-cost approach to the localization of and docking with a transport trolley task for an autonomous mobile robot. The algorithm identifies the trolley and determines the position of its legs based on predefined detection areas and data from a single LiDAR sensor, allowing navigation and docking using a mechanism that eliminates the need for precise positioning of the robot. The system consists of a mobile robot with an ESP32-based control system, using the ROS framework for real-time processing. Experimental results demonstrated successful trolley identification and docking when the robot was positioned in front of the target. The docking mechanism was validated through multiple trials, confirming feasibility in controlled conditions. This research contributes to the development of efficient, flexible automated transport systems for warehouses and manufacturing facilities.

Keywords: mobile robotics; LiDAR; autonomous transport; robot docking; industrial automation; AGV; IoT; MQTT.

1. INTRODUCTION

Robots are employed wherever they can relieve humans of tasks that are burdensome, repetitive, or hazardous. Mobile robots have found widespread applications in factories, warehouses, logistics centers, mining, healthcare, search and rescue operations, as well as in the exploration of other planets. They are also commonly used to clean residential spaces and for the transportation of people. In many cases, mobile robots outperform humans in physical endurance, precision, repeatability, operating speed, and data processing capabilities.

The history of robotic transport systems dates back to the early 1950s, when the first automated guided vehicles (AGVs) were introduced in industrial settings in the United States [1]. Since then, transport needs have remained critical although methods of fulfilling them have evolved significantly. Today, autonomous mobile robots (AMRs) are increasingly deployed in transport applications [2]. Equipped with control systems based on PLCs or PC-class computers, AMRs utilize advanced sensors, artificial intelligence, machine learning, and complex computation to perceive their environment, plan paths, and navigate autonomously within it [3].

In contrast to AGVs, AMRs are not restricted to predefined physical paths. Thanks to spatial mapping using 2D or 3D LiDAR scanners, they can follow virtual paths and dynamically adjust them to avoid obstacles. This flexibility reduces implementation costs by decreasing the need to modify the infrastructure and also allows for a more universal application, compatible with different transport systems [3].

Mobile transport robots can be classified into three main types based on their function: unit load vehicles, towing vehicles, and fork vehicles. A crucial aspect of implementing such systems is their integration with the operating environment. This includes both mechanical adaptation, such as lifting, coupling, or docking, and synchronization of the information flow. As a result, many innovative solutions have emerged for load pickup, placement, and trailer connection.

One popular application is order fulfillment in warehouses, where robots bring entire shelving units to human workers for further processing. A well-known example is the KIVA Systems robot system [4, 5], which navigates using QR codes on the floor and requires that the shelving units be precisely positioned. Although effective, this system has limitations, such as the robot bearing the full load of the rack and the need for strict environmental constraints.

When multiple robots operate in shared spaces, effective task and path management is essential. Optimization algorithms are used to reduce travel time and avoid collisions [6]. Tasks are typically assigned wirelessly based on incoming customer orders. The assignment of tasks is usually done by warehouse management systems [7].

Alternatively, a mobile robot can be designed to drive over the trolley, and engage with it using a docking mechanism. An example of such an implementation is described in [8], where multiple sensors are used simultaneously, including three vision cameras and two laser scanners. The trolleys used in this approach must be equipped with four AR markers, enabling continuous observation and the correct identification of a selected

*e-mail: jaroslaw.szrek@pwr.edu.pl

Manuscript submitted 2025-12-02, revised 2026-06-19, initially accepted for publication 2026-07-13, published in November 2026.

© 2026 The Author(s). This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/)

cart among several visible units. However, this solution requires that all trolleys be specially adapted and that the robot be outfitted with numerous sensors, which significantly increases the cost and complexity of the system.

In some configurations, the load transfer process between the robot and the loading station is automated using a roller conveyor mounted directly on the robotic platform [9]. Other solutions, particularly those requiring automatic cargo loading and unloading, rely on custom trolley designs that incorporate dedicated docking mechanisms [10, 11].

Another approach is described in [12], where the focus is on detecting and approaching the load. In this case, a camera, a 3D scanner, and a neural network are used to identify the load in the captured image. The system is designed to recognize Euro pallets, which are relatively uniform; however, the algorithm does not differentiate between individual pallet types or configurations.

An additional example comes from the domain of autonomous surface vehicles. In [13], a docking method is presented in which the platform must connect to a base station under wave-induced disturbances. Navigation is accomplished using a vision system that detects AprilTags, combined with a simple positioning algorithm. A mechanical docking system is also employed, using a force-based grip to center the platform and enable the connection to a charging interface. A similar vision-based system implemented on a land-wheeled robot is presented in [14].

In our solution, we employ an RGB camera and a single 2D LiDAR scanner in a setup similar to [15]. In our case, LiDAR also fulfills the role of a safety sensor, which is mandatory in autonomous mobile systems. This allows for significantly lower implementation costs compared to contemporary solutions, as well as a minimal amount of changes required to modify the transport trolleys to work with the system. Regarding mechanical coupling, our proposed system ensures stable docking through four contact points between the robot and the trolley, whereas the system described in [13] utilizes only two. This difference imposes higher structural requirements on both the robot and the docking mechanism. Furthermore, our design includes a mounting system that actively aligns and centers the robot with respect to the trolley, enhancing the robustness and precision of the docking process.

This paper proposes a 2D LiDAR-based algorithm for transport trolley localization during the final docking phase, supplemented by an RGB camera for trolley identification. The article is organized as follows: Section 2 describes the transport system's architecture and operating principles. Section 3 details the localization and docking algorithms, alongside experimental validation on a physical robot. Section 4 analyzes the experimental results, and Section 5 provides concluding remarks.

2. ROBOTIC TRANSPORT WITH TROLLEY

The robot operates in an industrial setting, where tasks are assigned by a central computer and distributed using a wireless Wi-Fi network with the IoT MQTT protocol [16] (Examples of message topics: robot\trolley, robot\target). To connect the MQTT communication protocol to the robot, we use the ROS MQTT_bridge package [17]. To simulate the conditions of a real-life application, we send messages on those MQTT topics manually. This provides compatibility with different operating systems and allows the robot onboard hardware to be significantly simpler. Similar solutions are used in various applications and have been previously described in different works [18, 19].

During its operation, the robot may need to dock with a transport trolley. In industrial settings, transport trolleys are usually located in pre-designated areas marked by colored tape. This tape is used only to indicate the dimensions of a parking space for a human worker and is not required by the localization algorithm in any way. Workers are required to return the trolley to the designated area after use. In such a setting, the general placement of the transport trolley is known; however, due to the manual placement of the trolley and the parking space being larger than the trolley itself, the exact location and orientation of the trolley are unknown. An automated docking maneuver requires that the exact location and orientation of the trolley be known. This necessitates an algorithm that utilizes the available sensor data to define the exact location and orientation of the trolley. The general idea of how the docking maneuver should be performed is presented in Fig. 1.

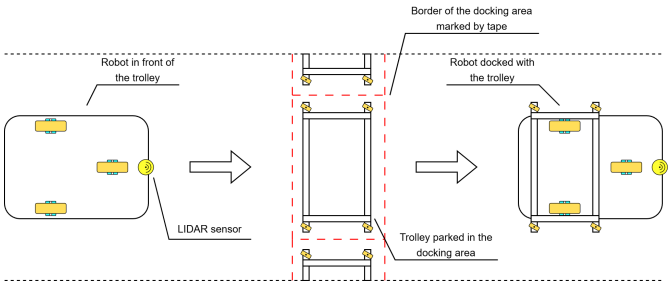


Fig. 1. The general idea of using a mobile robot with a transport trolley. The robot approaches the occupied trolley parking spot from the front, drives underneath the trolley, docks with it and hauls it away from the parking space

In such a scenario, the localization of the trolley is performed at the start, and further navigation is conducted using the localization data obtained from the algorithm. The navigation process does not need to perform obstacle avoidance maneuvers, as it takes place in an industrial setting with restricted access when the robot is in the zone, and no obstacles should be present at the time of the scenario.

2.1. Mobile robot

The robot frame was built using extruded aluminum profiles. Two wheels are attached to the chassis, creating a class (2, 0) drive system. An additional castor wheel is placed on the front of the robot to provide passive stability to the platform [20, 21].

If we assume that the robot moves on a planar surface, the description of its movement can be accomplished using only three parameters. Two of those (X_P, Y_P) are used to describe the position of the robot, and φ is used to describe its orientation.

This allows the robot to be described using a three-element vector $q = (X_P, Y_P, \varphi)^T$ based on a local coordinate system, as shown in Fig. 2 [22].

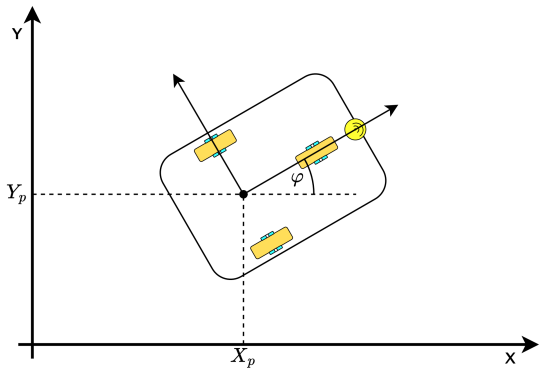


Fig. 2. Coordinates system of mobile robot

To describe the movement of the robot, it is necessary to place the coordinate system $x_P y_P$ in the middle of the axle that connects the two driving wheels. The longitudinal velocity v, along with the rotational velocity ω, creates a set of input parameters that allows for control over the robot. This allows for a concise presentation of the equations that represent the kinematics of the platform, as presented in equation (1) [23].

$$\begin{bmatrix} \dot{x}_P \\ \dot{y}_P \\ \dot{\varphi} \end{bmatrix} = \begin{bmatrix} \cos\varphi & 0 \\ \sin\varphi & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}. \tag{1}$$

The mobile platform kinematic model in the form of equation (1) has natural input parameters (v, ω). This model representation is used in motion planning algorithms. Calculations for individual wheel velocities based on the values of (v, ω) are carried out in a low-level controller module, given the knowledge of robot wheel spacing d and using equation (2) [23].

$$\begin{bmatrix} V_L \\ V_R \end{bmatrix} = \begin{bmatrix} V - \dfrac{\omega \cdot d}{2} \\ V + \dfrac{\omega \cdot d}{2} \end{bmatrix}. \tag{2}$$

Odometry is calculated and converted to SI compliant units by the low-level controller module at a frequency f, by calculating each wheel speed ($V_{L_{\text{curr}}}, V_{R_{\text{curr}}}$), as seen in equation (3). They are calculated using the current ($S_{L_{\text{curr}}}$, $S_{R_{\text{curr}}}$) and previous position data ($S_{L_{\text{prev}}}, S_{R_{\text{prev}}}$) obtained from the encoders. Afterwards, using the calculated velocities and data from the previous odometry calculation, a numerical integration is performed to obtain the current position of the robot (X_p, Y_p), as described in equation (4).

$$\begin{bmatrix} V_{L_{\text{curr}}} \\ V_{R_{\text{curr}}} \end{bmatrix} = \begin{bmatrix} f(S_{L_{\text{curr}}} - S_{L_{\text{prev}}}) \\ f(S_{R_{\text{curr}}} - S_{R_{\text{prev}}}) \end{bmatrix}, \tag{3}$$

$$\begin{bmatrix} X_{p_{\text{curr}}} \\ Y_{p_{\text{curr}}} \\ \varphi_{\text{curr}} \end{bmatrix} = \begin{bmatrix} X_{p_{\text{prev}}} + \dfrac{V_{L_{\text{curr}}} + V_{R_{\text{curr}}}}{2} \cos\left(\varphi_{\text{prev}} + \dfrac{V_{R_{\text{curr}}} - V_{L_{\text{curr}}}}{2df}\right) \dfrac{1}{f} \\ Y_{p_{\text{prev}}} + \dfrac{V_{L_{\text{curr}}} + V_{R_{\text{curr}}}}{2} \sin\left(\varphi_{\text{prev}} + \dfrac{V_{R_{\text{curr}}} - V_{L_{\text{curr}}}}{2df}\right) \dfrac{1}{f} \\ \varphi_{\text{prev}} + \dfrac{V_{R_{\text{curr}}} - V_{L_{\text{curr}}}}{d} \dfrac{1}{f} \end{bmatrix}. \tag{4}$$

2.2. Sensory and control system

The schematic of the robot's sensory and control systems is shown in Fig. 3. The main computer of the robot uses the Linux operating system with the Robot Operating System (ROS) [24] software platform installed. ROS is used to provide compatibility with a large library of developed software and to simplify communication between sensory systems, control algorithms, and command-executing software. The main computer is connected to the onboard Wi-Fi router of the robot using an Ethernet cable, allowing the operator to supervise data collection and launch various programs to realize different scenarios while remaining out of range of the stationary infrastructure. Connection to a stationary Wi-Fi network and the internet is still possible using the PCs onboard network card.

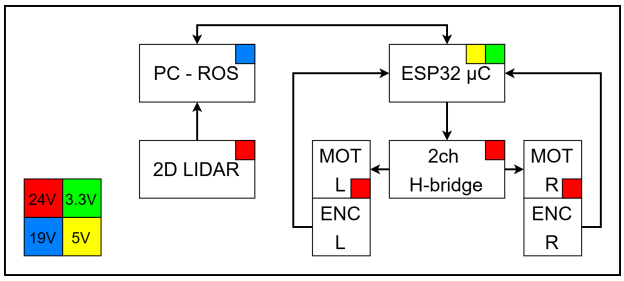


Fig. 3. Control and sensory systems of the mobile robot and their power requirements

The main way to obtain data about the robot environment is by using a 2D LiDAR sensor. LiDAR (Light Detection and Ranging) is an active ranging technology that calculates the distance to an object by measuring the round trip time of a laser pulse [25]. Due to favorable parameters and availability, the SICK S300 2D LiDAR [26] was selected for use in the robot. It is connected to the main computer using a USB-UART converter, which allows the driver to interface with the hardware and, after decoding, feed the sensor data into ROS.

The low-level hardware consists of an ESP32-WROOM-32 module [27] based on an ESP32-D0WDQ6 chip [28]. Communication between the main computer and the microcontroller is carried out via the USB port and the microcontroller onboard USB-UART converter. This allows for easy transfer of commands and odometry reporting between the main computer and low-level hardware.

The drive system is entirely controlled by the ESP32 microcontroller. Each wheel is driven by a separate H-Bridge and has its own magnetic encoder mounted directly on the motor [29]. The ESP32 reads the encoder data and calculates the current velocities for each wheel by multiplying the corresponding motor velocity by 9.78, which is the inverse of the gear ratio between

the motor and wheel speeds. Using the wheel speed, the distance traveled by each wheel is then calculated, and the wheel velocity is fed into a PI controller, which, in turn, controls the H-Bridge, forming a closed-loop control system [30]. The target speed is received from the main computer in the form of longitudinal and angular speeds, which are then recalculated to determine the target speed for each wheel. The control loop is run at a frequency of 100 Hz. Additionally, odometry calculated using data from the encoders is returned to the master computer at a frequency of 10 Hz.

2.3. Trolley docking mechanism

To provide the robot with the ability to move the trolley, a mechanical docking system had to be designed. It was determined that the system should offer tolerance for trolley positioning inaccuracies while remaining simple to control using the robot onboard microcontroller.

The mechanism was designed to consist of a minimal number of links. Four passive links were designed, each responsible for immobilizing one leg of the attached cart. The kinematic diagram presenting the arrangement of these links is shown in Fig. 4. All passive links are controlled by a single active link, which is an electric actuator. The drive system is attached on one side to the main body and on the other to a passive link. The motion of this link is transmitted to the others via tendons mounted on ball joints. The dimensions were selected to allow passive links to rotate within a 90-degree range, ensuring a secure grip on the cart. The operation of the coupling mechanism is shown in Fig. 5.

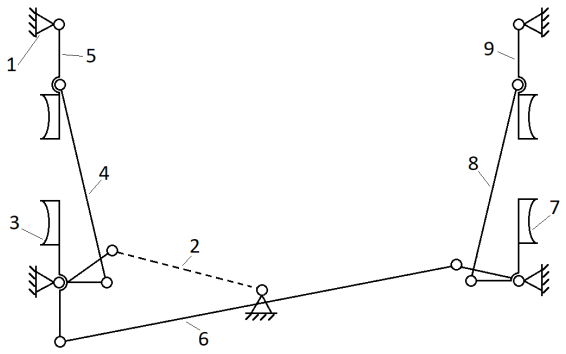
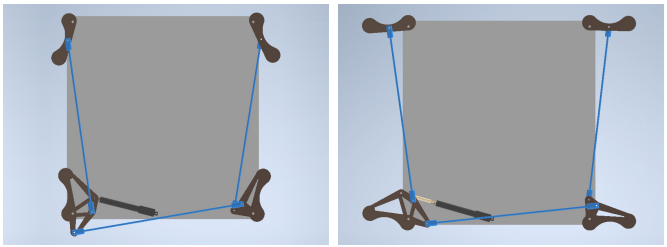


Fig. 4. Kinematic diagram of the docking mechanism with each mechanical link numbered, where 1 – base; 2 – actuator; 3, 5, 7, 9 – passive links; 4, 6, 8 – tendons (couplers)

The stiffness of the robot-cart connection can be adjusted within a certain range by indirectly regulating the force applied by the passive links to the cart structure, using the force generated by the drive system. Additionally, the selected dimensions minimize the space occupied by the mechanism, leaving a larger portion of the main body free. This allows the remaining area to be utilized as cargo space or for other equipment.

An additional advantage of the mechanism is its high scalability, achieved by modifying the length of the tendons. This enables the rapid adaptation of the robot to transport different



(a) (b)

Fig. 5. Operation of the robot docking mechanism: (a) Docking mechanism retracted; (b) Docking mechanism extended

carts or even transfer the mechanism to a different robot with dissimilar dimensions. The mechanism consists solely of rotary and ball joints, making its construction simple. Furthermore, it is mounted to the main body at only five points using screws, which simultaneously serve as the rotational axes for the passive links and the actuator. After designing, the prototype elements of the mechanism were manufactured using rapid prototyping technologies, such as 3D printing.

3. LOCALIZATION ALGORITHM AND EXPERIMENTAL VALIDATION

The initial conditions for the algorithm consist of the robot being positioned in front of the trolley parking site, with the trolley inside, similar to those shown in Fig. 6.



Fig. 6. The initial state before the localization, featuring a robot in front of the trolley

This state is a result of a previous movement task sent by a central planning computer, which placed the robot in front of the trolley parking space. This movement could have been performed by any combination of the movement planner and the executor, as this does not impede the operation of the localization algorithm in any way.

This initial state may result from any other navigation algorithm compatible with the ROS system and a previously created environmental map.

The operation of the localization algorithm using 2D LiDAR sensor data is illustrated in the diagram presented in Fig. 7.

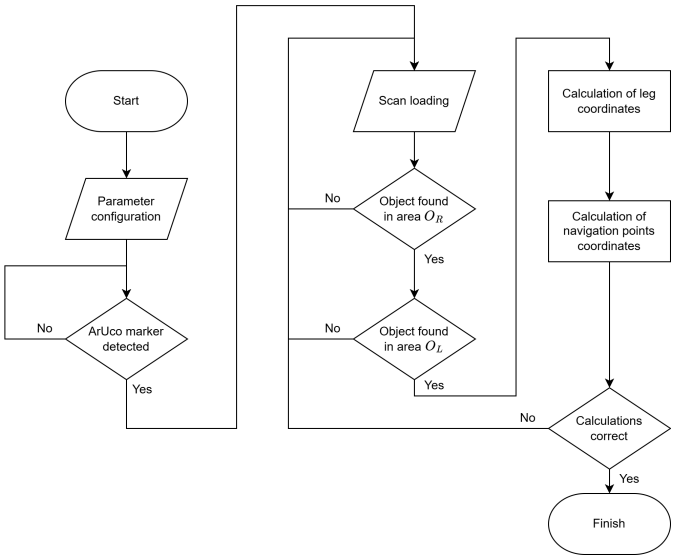


Fig. 7. Block diagram of the trolley localization algorithm

Due to the relatively low map resolution of 50 mm and the possible inaccuracy of trolley placement within the parking area, the algorithm must localize the trolley without relying on previously obtained data. For this task, an onboard 2D LiDAR scanner, already used for mapping and localization, is utilized.

Because the task received from the planning computer requires docking with a specific trolley, the algorithm verifies that an appropriate trolley is present in the parking space by reading the ArUco marker attached to the trolley. Markers are detected using an RGB camera mounted on the robot. If the trolley ID read from the ArUco marker matches the one given in the task, the robot proceeds to the detection phase.

The proposed detection is based on the localization of the trolley legs using predetermined areas of interest defined by the minimum distance $Dist_{\min}$, maximum distance $Dist_{\max}$, and minimum $\alpha_{\min}$ and maximum $\alpha_{\max}$ angles for each side of the trolley, as shown in Fig. 8. Pre-determining those parameters ensures the reliability and repeatability of the localization process, as well as eases debugging.

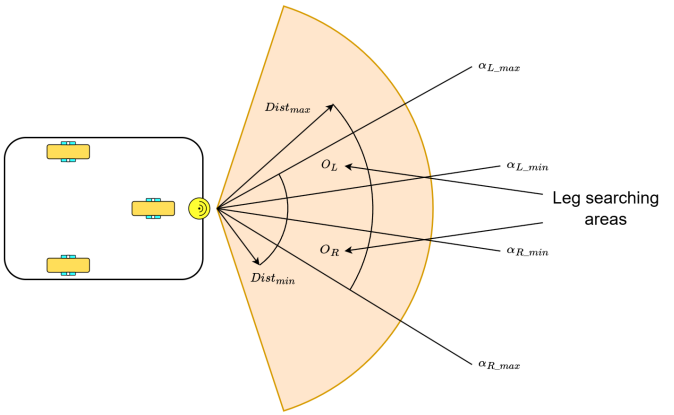


Fig. 8. Representation of where the transport trolley legs are expected to be

Figure 9 compares how the trolley is seen using an RGB camera and LiDAR. Due to the height at which the LiDAR is mounted, only the legs of the trolley are visible in the collected data. Ultimately, due to the simplicity of the data, a solely LiDAR based approach was selected.



(a) (b)

Fig. 9. Comparison of how the trolley placed near a wall is visible on camera and using LiDAR data: (a) Front view of the trolley using an RGB camera and featuring the detected and correctly recognized ArUco marker; (b) Top-down view of representation of the robots vicinity using LiDAR data, where the robots position is described using an arrow

If the trolley legs are found in the designated area, the innermost detected points are saved, and their coordinates are marked as L_X and L_Y for the left leg, and R_X and R_Y for the right leg, as visualized in Fig. 10. Based on the coordinates of the inner leg, a point T located in the middle of the trolley is determined, as seen in equation (5).

$$\begin{aligned} T_X &= \frac{L_X + R_X}{2}, \\ T_Y &= \frac{L_Y + R_Y}{2}. \end{aligned} \tag{5}$$

Based on this, equation (6) of a line L_1, located between the point of the left leg and the point of the right leg, is determined. Then, equation (7) of a line L_2 that is perpendicular to the line L_1 and passes through the midpoint T is determined.

$$\begin{aligned} L_1 &: y = L_{1_a} x + L_{1_b}, \\ L_{1_a} &= \frac{L_Y - R_Y}{L_X - R_X}, \\ L_{1_b} &= L_Y - L_{1_a} L_X, \end{aligned} \tag{6}$$

$$\begin{aligned} L_2 &: y = L_{2_a} x + L_{2_b}, \\ L_{2_a} &= -\frac{1}{L_{1_a}}, \\ L_{2_b} &= T_Y - L_{2_a} T_X. \end{aligned} \tag{7}$$

X and Y coordinates of two navigation points N_1 and N_2, located on the line L_2, at a distance $Dist_N$ from the point T, are

calculated using equation (8).

$$
\begin{aligned}
a &= (L_{2_a})^2 + 1\,,\\
b &= 2L_{2_a}L_{2_b} - 2L_{2_a}T_Y - 2T_X\,,\\
c &= (T_X)^2 + (L_{2_b})^2 - 2L_{2_b}T_Y + (T_Y)^2 - (Dist_N)^2\,,\\
\Delta &= b^2 - 4ac\,,\\
N_{1_X} &= -\frac{b+\sqrt{\Delta}}{2a}, \quad N_{1_Y} = L_{2_a}N_{1_X} + L_{2_b}\,,\\
N_{2_X} &= -\frac{b-\sqrt{\Delta}}{2a}, \quad N_{2_Y} = L_{2_a}N_{2_X} + L_{2_b}\,.
\end{aligned}
\tag{8}
$$

In rare cases where one of the values $|L_X - R_X|$, $|L_{1_a}|$ is less than 10^{-6}, the calculations are stopped, and the next measurement is taken.

After calculating the navigation points, the algorithm checks whether point N_1 is closer to the current position of the robot. If not, the N_1 and N_2 points are switched to conform to the schematic presented in Fig. 10.

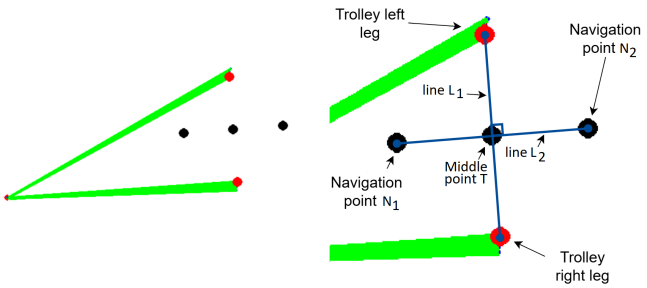


Fig. 10. Trolley leg detection: (a) General view; (b) Details

All coordinates are defined using the laser scanner as the base point. To use them for navigational purposes, it is necessary to apply a transform using the robot odometry from the time when the scan was made.

The functioning of the algorithm was tested in many different situations. The tests involved checking the success rate of the algorithm at different distances, as shown in Fig. 11, at various angles of approach, as illustrated in Fig. 11, and in the presence of other objects within the scanner range, as shown in Fig. 12. Each test was repeated at least 10 times to verify whether the algorithm functioned correctly in each scenario. Only a clean docking without hitting the trolley was counted as a success.

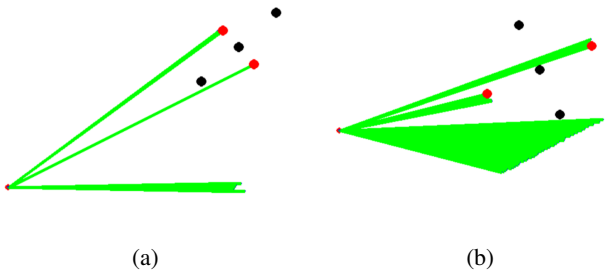


Fig. 11. Algorithm robustness test: (a) Changing the distance; (b) Changing the angle

The algorithm performed successfully within a range of up to 2 m, correctly recognizing the legs, docking with the trolley, and exiting the parking area. Above that distance, the algorithm often misidentified the front trolley legs as the rear ones and either attempted to dock while still in front of the trolley or, if only one leg was recognized correctly, tried to drive underneath the trolley at an angle, resulting in a failure of the task.

Similar results were obtained during the angle-of-approach tests. If the robot was located almost directly in front of the trolley, the algorithm worked correctly. If the angle caused the LiDAR to pick up one of the front legs as the inner one, the algorithm misidentified the position of the trolley and failed to complete the task.

The algorithm functioned correctly in the presence of additional objects, provided that the initial conditions were well defined.

Additionally, it was observed that the slight reflectivity of the trolley legs may cause minor data errors due to the LiDAR laser reflecting off the surface, similar to those seen in Fig. 12. This was easily resolved by making the surface matte; however, in cases where this is not possible, it could be addressed by filtering the data from several LiDAR sensor passes.

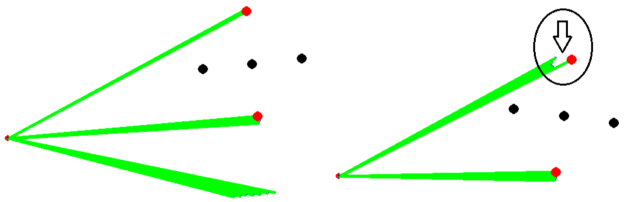


Fig. 12. An example showing uncertainty in the detection of trolley legs: (a) Another object in the surroundings; (b) Laser reflection

Algorithm verification was performed using the Python programming language [31]. To visualize the results of the simulations, the *PyGame* library was used [32]. All parameters for the algorithm were selected based on observations of the simulation results.

3.1. Algorithm implementation and experiments

The implementation of the algorithm into the ROS environment was achieved by creating a ROS node that addresses the transport trolley localization problem. The main goal of this node is to localize the trolley in front of the robot using the previously described localization algorithm and to use the obtained navigation points to drive underneath the trolley, attach it using the robot docking system, and clear the d ocking area to allow for further operations. In addition to the parameters needed

for the localization algorithm, implementation requires an additional "*trolley_width*" parameter. This parameter is used for the validation of the localization process.

Furthermore, the algorithm requires access to the "*sick_scan*" topic, where current data from the LiDAR sensor are published using the ROS "*LaserScan*" message, which can be found in the "*sensor_msgs*" package. Additionally, to correctly transform between frames of reference, the algorithm needs access to the current transform between the robot base and its environment.

The sending of movement commands is handled using an "*action_lib*" client that is compatible with the "*move_base*" interface. This allows the algorithm to be compatible with most widely used ROS movement planners.

The algorithm itself is divided into two parts, as shown in Fig. 13. The first part implements the trolley localization algorithm according to the previously discussed algorithm, as shown in Fig. 7. To validate the results of the algorithm, it is repeated multiple times. Each localization is deemed successful if the distance between the legs of the localized trolleys is less than 15 cm from the declared width of the trolley. The localization process is deemed successful if the algorithm returns 10 consecutive successes. This process has been implemented to reduce the number of false detections of stationary obstacles, such as walls. While each pass of the localization algorithm calculates navigation points based on its detected leg location, only the navigation point coordinates from the last pass are used for the path planning process.

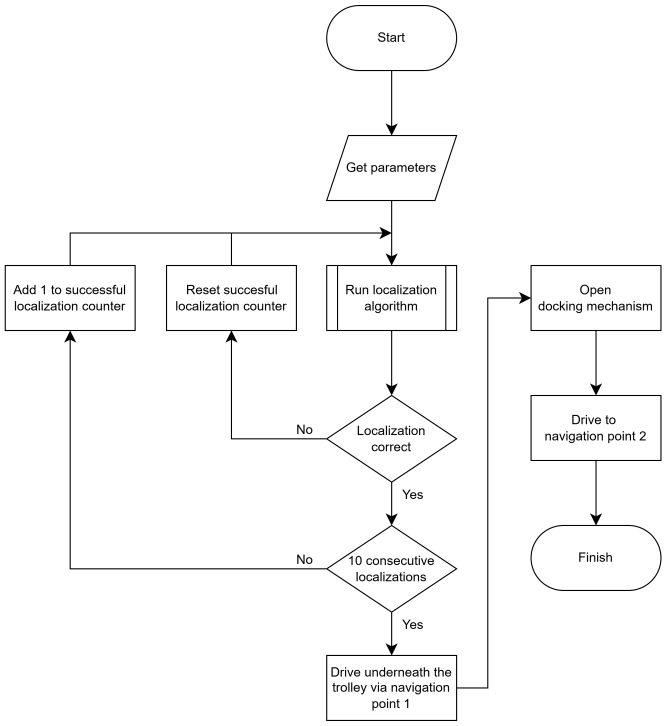


Fig. 13. Block diagram of the localization algorithm implementation during the experiment

The second part of the algorithm deals with path planning for the robot via navigation point one and the point in between the legs of the trolley. Due to the tight tolerances between the trolley and the robot, path planners that feature obstacle avoidance may cause various problems. Because of this, for experimental purposes, a simple three-phase planner was used.

The movement executor receives the current robot odometry $(x_{\text{odom}}, y_{\text{odom}})$, the coordinates of the target point $(x_{\text{target}}, y_{\text{target}})$, and the target orientation; it continuously publishes the state of the task until the task is complete.

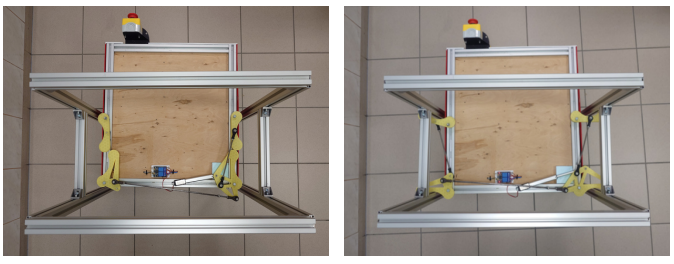
The first phase consists of aligning the bearing between the target and the robot position. To do this, the robot rotates in the appropriate direction at a constant speed until it is within 0.05 radians of the previously calculated yaw_{target}. This value is obtained by calculating a two-argument arctangent using the difference between the coordinates of the target and the starting coordinates, as shown in equation (9) [33].

$$yaw_{\text{target}} = \operatorname{atan2}(Y_{\text{target}} - Y_{\text{odom}},\ X_{\text{target}} - X_{\text{odom}}), \tag{9}$$

The second phase consists of the robot moving linearly towards the target at a constant speed of $0.1\ \frac{\text{m}}{\text{s}}$, while maintaining its heading using a simple PI regulator with experimentally selected coefficients. This phase ends when the robot is within 5 cm of the target point or if the distance from the target point increases in the last second.

The last phase consists of the robot matching the final target bearing. This is done similarly to the first phase; however, the target yaw is received at the beginning of the task.

This algorithm consists of three separate movement tasks. The first task is to drive from the starting position to the first navigation point and face the point in the middle of the trolley. The second task is to drive to the point in the middle of the trolley while maintaining heading. After this task is completed, the robot has to deploy the trolley docking mechanism (Fig. 14). When the deployment is confirmed, the robot drives with the trolley to the second navigation point, exits the trolley parking area, finishes the algorithm, and allows further operations using any available navigation method.



(a) (b)

Fig. 14. Operation of the docking mechanism:(a) Top-down view of the robot before docking with the trolley; (b) Top-down view of the robot after docking with the trolley

The experimental verification of the algorithm consisted of multiple attempts to locate and drive under the trolley. The setup constructed to simulate a real-life use case was made using a simple four-wheeled trolley made of aluminum extrusions, which

could be used for the transportation of elements in an industrial setup. The trolley was built in a rectangular shape with a length and width of 53 cm. The LiDaR sensor was placed at the front of the robot, 24 cm above the ground and 15.5 cm to the left of the longitudinal axis of the robot. The main goal of the experiment was to investigate the limitations of the algorithm under conditions that would not be encountered in normal operation, particularly to obtain guidelines for the appropriate preparation of infrastructure.

The first series of experiments tested the algorithm success rate depending on the lateral distance of the transport trolley from the longitudinal axis of the robot. The scenario was constructed as follows: The trolley was placed at a longitudinal distance of 1.0 m from the robot and a lateral distance ranging from −0.2 m to 0.2 m, where the distance to the right is counted as a positive value. Upon the start of the experiment, the trolley had to correctly recognize the positions of its legs, perform a docking maneuver, and drive to the calculated navigation point 2. The success rate is calculated as the ratio of successful docking maneuvers to the total number of attempts. Each distance was tested 10 times. The results of this series of experiments are presented in Fig. 15. The shift of 0 m indicates that the robot is in a position where the LiDAR sensor is directly in front of the trolley.

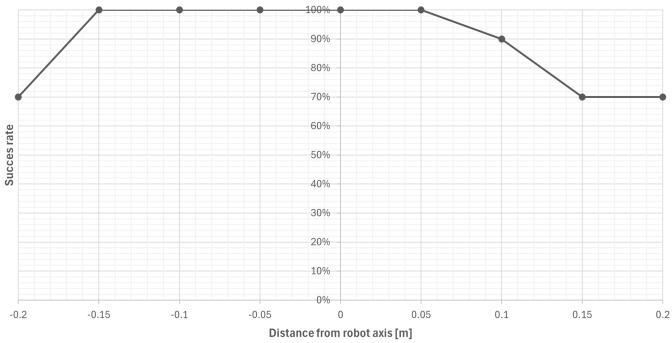


Fig. 15. Chart showing the algorithm success rate depending on the trolley's lateral distance from the longitudinal axis of the robot

The second series of experiments tested the algorithm success rate depending on the angle at which the trolley is placed in relation to the robot. The scenario was constructed as follows: The trolley was placed directly in front of the robot at a distance of 1.0 m and oriented at an angle ranging from −22.5° to 22.5°, where the positive value represents clockwise rotation. Similar to the first experiment, the robot has to correctly recognize the trolley legs, perform the docking maneuver, and drive with the trolley to the last navigation point. The success rate is calculated in the same way as in the first series of experiments. Each trolley orientation was tested 10 times. The results of this series of experiments are presented in Fig. 16.

The third series of experiments tested the algorithm success rate depending on the robot distance from the trolley. The scenario was constructed as follows: The trolley was placed directly in front of the robot at a distance of 0.5 to 3.0 m. The rest of

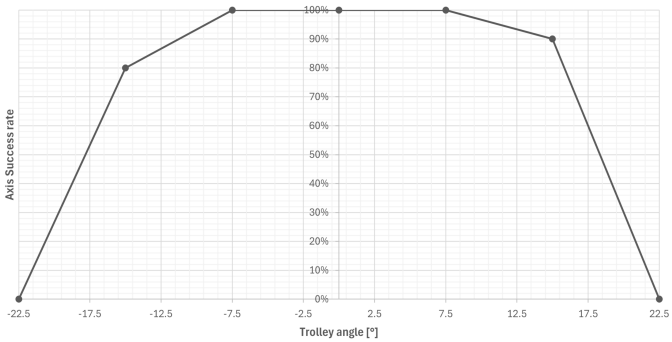


Fig. 16. Chart showing the algorithm success rate depending on the angle at which the trolley is placed at

the experiment was conducted in the same way as in the cases of experiment series 1 and 2. Each position was tested 10 times. The results of this series of experiments are presented in Fig. 17.

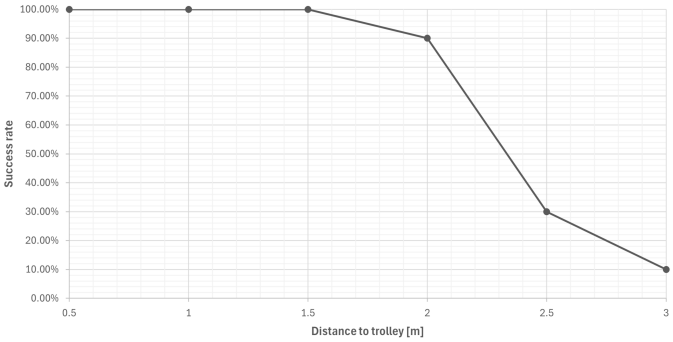


Fig. 17. Chart showing the success rate of the algorithm depending on the lateral distance to the trolley

4. DISCUSSION

The results obtained show that the algorithm works correctly if the robot is in front of or to the left of the transport trolley, within 2.0 m, at an angle ranging from 0° to 15°, and shifted sideways by up to 0.2 m. The tight fit between the trolley and the robot indicates that the proposed docking method achieves performance only slightly worse than that of more advanced approaches, such as those shown in [34]. This, in conjunction with the proposed docking mechanism, ensures the correct functioning of the whole docking system.

The decreased reliability of the algorithm when the shift distance is increased beyond 0.2 m or the trolley angle is greater than 10° is caused by reflections picked up by the LiDAR sensor and the obscuration of the rear legs of the trolleys by the front ones. The first issue could be resolved by matting or painting the surface, while addressing the second issue would require a complete redesign of the trolley to feature only two legs at the height where the LiDAR sensor is located.

The loss of reliability when the overall distance to the trolley is increased is caused by inaccuracies in the odometry system. To solve this issue, the algorithm needs to be run continuously

up to the distance within the reliability range to continuously update the navigation points.

The aforementioned issues are only relevant if the environment in which the robot is operating requires detections outside the range deemed reliable. For multiple applications, the docking algorithm would only be used for terminal guidance and the starting point would be within the range deemed reliable.

5. CONCLUSIONS

This article presents an algorithm for the identification and localization of a transport trolley using an RGB camera that detects an ArUco marker placed on the trolley, in addition to data from a 2D LiDAR scanner.

The use of a laser-based distance sensor requires proper preparation of the localized elements. This entails ensuring that there are no reflective surfaces present, which could cause incorrect readings, as well as making repeated passes of the distance reading and filtering the obtained data.

The algorithm presented in this article was tested in various scenarios that involved the hook-up of the robot to the transport trolley. The experiments carried out not only confirmed the correct identification and localization of the trolley but also aimed to determine the boundary conditions under which the algorithm operates reliably. The results help define the requirements that must be fulfilled when preparing the infrastructure at the robot deployment site.

The results obtained show that the algorithm could be viable for implementation in a potential industrial setting, provided that the trolley docking process is carried out in a structured environment where the trolley is located in the vicinity of a known area. Because such conditions are usually characteristic of an industrial environment prepared for the presence of robots, the algorithm is viable for use in such an environment. To make the process more reliable and independent of the angle of approach, it would be advisable to use a more computation-intensive algorithm that recognizes more features of the trolley and utilizes constant updates of the navigation points until the accuracy of the current algorithm is enough.

Further research on this topic may focus on developing and testing different algorithms that detect all four legs of a trolley, increasing the reliability of the current algorithm, using different sensors, or developing a capability for the robot to dock with several transport trolleys simultaneously.

REFERENCES

[1] G. Ullrich, "The history of automated guided vehicle systems," in *Automated Guided Vehicle Systems — A Primal With Practical Applications*. Springer, 2014, pp. 1–14.

[2] G. Fragapane, R. de Koster, F. Sgarbossa, and J.O. Strandhagen, "Planning and control of autonomous mobile robots for intralogistics: Literature review and research agenda," *Eur. J. Oper. Res.*, vol. 294, no. 2, pp. 405–426, 2021, doi: 10.1016/j.ejor.2021.01.019.

[3] P.S. Pratama *et al.*, "Positioning and obstacle avoidance of automatic guided vehicle in partially known environment," *Int. J. Control Autom. Syst.*, vol. 14, no. 6, p. 1572–1581, 2016, doi: 10.1007/s12555-014-0553-y.

[4] Amazon.com, Inc., "Meet the Robots at Amazon - promotion movie," Amazon.com, Inc. [Online]. Available: https://www.youtube.com/watch?v=HSA5Bq-1fU4 (Accessed 2022-12-26).

[5] R.D. Meller, D. Nazzal, and L.M. Thomas, "Collaborative bots in distribution centers," in *15th IMHRC Proceedings,Savannah, Georgia. USA*, Savannah, USA, 2018.

[6] E. Roszkowska, P. Makowski-Czerski, and L. Janiec, "Multilevel control for mobile robot systems," *IFAC-PapersOnLine (16th IFAC Workshop on Discrete Event Systems WODES 2022)*, vol. 55, no. 28, pp. 87–92, 2022, doi: 10.1016/j.ifacol.2022.10.328.

[7] C. Kirsch and C. Röhrig, "Global localization and position tracking of an automated guided vehicle," *IFAC Proc. Vol. (18th IFAC World Congress)*, vol. 44, no. 1, pp. 14 036–14 041, 2011, doi: 10.3182/20110828-6-IT-1002.01245.

[8] G. Yang *et al.*, "Autonomous cart docking for transportation robots in complex and dynamic construction environments," *Autom. Constr.*, vol. 171, p. 105968, 2025, doi: 10.1016/j.autcon.2025.105968.

[9] Mobile Industrial Robots (MiR), "Mobile Industrial Robots company webpage," Mobile Industrial Robots (MiR). [Online]. Available: https://www.mobile-industrial-robots.com/case-studies/mir100-honeywell-safety-productivity-solutions-united-kingdom/ (Accessed 2022-12-26).

[10] Mobile Industrial Robots (MiR), "Mir250 mobile robot with top shelf," Mobile Industrial Robots (MiR). [Online]. Available: https://www.mobile-industrial-robots.com/mir-go-applications/framesshelves/blueamr-cart250/ (Accessed 2023-02-20).

[11] Versabox, "Offer of versabox," Versabox. [Online]. Available: https://versabox.eu (Accessed 2023-02-20).

[12] E. Tsiogas *et al.*, "Pallet detection and docking strategy for autonomous pallet truck agv operation," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 3444–3451, doi: 10.1109/IROS51168.2021.9636270.

[13] K. Xue *et al.*, "A two-stage automatic latching system for the usvs charging in disturbed berth," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 1748–1754, doi: 10.1109/IROS45743.2020.9341091.

[14] A.M. Romanov and A.A. Tararin, "An automatic docking system for wheeled mobile robots," in *2021 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus)*, 2021, pp. 1040–1045, doi: 10.1109/ElConRus51938.2021.9396509.

[15] F. Jia, M. Afaq, B. Ripka, Q. Huda, and R. Ahmad, "Vision- and lidar-based autonomous docking and recharging of a mobile robot for machine tending in autonomous manufacturing environments," *Appl. Sci.*, vol. 13, no. 19, p. 10675, 2023, doi: 10.3390/app131910675.

[16] OASIS, "MQTT: The standard for iot messaging," OASIS. [Online]. Available: https://mqtt.org/ (Accessed 2025-05-27).

[17] Open Robotics, "mqtt_bridge entry on the ros wiki," Open Robotics. [Online]. Available: http://wiki.ros.org/mqtt_bridge (Accessed 2025-05-27).

[18] J. Szrek, J. Jakubiak, and R. Zimroz, "A mobile robot-based system for automatic inspection of belt conveyors in mining industry," *Energies*, vol. 15, no. 1, p. 327, 2022, doi: 10.3390/en15010327.

[19] J. Szrek, J. Wodecki, R. Błażej, and R. Zimroz, "An inspection robot for belt conveyor maintenance in underground mine—infrared thermography for overheated idlers detection," *Appl. Sci.*, vol. 10, no. 14, p. 4984, 2020, doi: 10.3390/app10144984.

[20] R. Siegwart, I.R. Nourbakhsh, and D. Scaramuzza, *Introduction to Autonomous Mobile Robots*. The MIT Press, 2011, ch. 2. Locomotion, pp. 33–38.

[21] B. Siciliano and O. Khatib, *Handbook of robotics*. Springer, 2016, ch. 24. Wheeled Robots, pp. 583–584.

[22] B. Siciliano and O. Khatib, *Handbook of robotics*. Springer, 2016, ch. 24. Wheeled Robots, pp. 578–579.

[23] R. Siegwart, I.R. Nourbakhsh, and D. Scaramuzza, *Introduction to Autonomous Mobile Robots*. The MIT Press, 2011, ch. 3. Kinematics, pp. 51–52.

[24] Open Robotics, "Ros project webpage," Open Robotics. [Online]. Available: https://www.ros.org/ (Accessed 2023-01-15).

[25] E. Marti, M.A. de Miguel, F. Garcia, and J. Perez, "A review of sensor technologies for perception in automated driving," *IEEE Intell. Transp. Syst. Mag.*, vol. 11, no. 4, pp. 94–108, 2019, doi: 10.1109/MITS.2019.2907630.

[26] *S300 Safety laser scanners*, SICK AG, 2024. [Online]. Available: https://cdn.sick.com/media/docs/3/13/613/operating_instructions_s300_safety_laser_scanner_en_im0017613.pdf

[27] *ESP32 WROOM 32 Datasheet*, Espressif Systems, 2023. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf

[28] *ESP32 Series Datasheet Version 4.8*, Espressif Systems, 2025. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

[29] T. Bräunl, *Embedded Robotics – Mobile Robot Design and Applications with Embedded Systems*. Springer, 2006, ch. 3 Actuators, pp. 44–46.

[30] K.J. Astrom and T. Hagglund, *PID Controllers: Theory, Design, and Tuning*. Instrument Society of America, 1995, ch. 3 PID control, pp. 64–69.

[31] Python Software Foundation, "Python home page." [Online]. Available: https://www.python.org/ (Accessed 2023-02-15).

[32] pygame community, "Pygame library." [Online]. Available: https://www.pygame.org (Accessed 2023-01-15).

[33] T. Yoshikawa, *Appendix 1 Function atan2*. The MIT Press, 1990, pp. 259–262.

[34] S. Hong, H. Kwon, G. Sim, K. Choi, and D. Park, "Ml-based fast and precise target docking of autonomous mobile robots for intelligent transportation systems using 2-d lidar," *IEEE Trans. Intell. Transp. Syst.*, vol. 26, no. 10, pp. 16 361–16 376, 2025, doi: 10.1109/TITS.2025.3585295.