Comparison of Hector SLAM and Gmapping for a Self-driving Mobile Robot on Slippery Surface

M. S. I. Safizan[1], N. M. Thamrin[2,*], K. A. Juhari[3]

[1]School of Electrical Engineering, College of Engineering, Universiti Teknologi MARA, Shah Alam, Selangor, Malaysia.

[2]Microwave Research Institute, Universiti Teknologi MARA, Shah Alam, Selangor, Malaysia

[3]Bizbot Technology, UKM-MTDC Technology Centre, Universiti Kebangsaan Malaysia, Bangi, Selangor,

ABSTRACT: This work compared the performance of two Simultaneous Localisation and Mapping (SLAM) algorithms, Hector SLAM and Gmapping, for self-navigation of a mobile robot in a small, slippery surface and controlled environment. The experiment utilised the Bveeta Mini mobile robot within a tiled corridor area. The primary objective was to evaluate and compare the accuracy of these algorithms in self-navigating the robot using acquired robot positions in 2D coordinates. The experiment involved manual mapping of the environment using both Gmapping and Hector SLAM, followed by autonomous navigation tasks with each algorithm. Performance was assessed by comparing the absolute error, absolute relative error, and percentage error between the robot's position obtained from the manual map and its position during autonomous navigation provided by the SLAM algorithms. It was found that the Hector SLAM achieved higher accuracy in all navigation paths than Gmapping. Gmapping suffered from significant errors, particularly in the robot's initial position, likely due to its reliance on odometry data, which was highly susceptible to errors from the slippery surface in the experimental area. In conclusion, both algorithms can be integrated with other advanced SLAM techniques to improve the accuracy of the generated map and robot position.

KEYWORDS: Mobile robot, Robot position, Hector SLAM, Gmapping, Autonomous

[Received Sep. 1, 2024; Revised Nov 27, 2024; Accepted Dec. 2, 2024] Print ISSN: 0189-9546 | Online ISSN: 2437-2110

I. INTRODUCTION

In many industries, the cost of manual labour is rising, making robots a more cost-effective alternative for repetitive (Chahal et al., 2023), dangerous (Javaid et al., 2022), or tedious tasks (Mehrotra & Shetty, 2023). Due to the advancement of robot computational in manufacturing, the increased competition in this sector has driven demand for automation and efficiency, which mobile robots can provide (Javaid et al., 2022). Furthermore, the growth of e-commerce has fuelled demand for automated warehouse and logistics solutions, where mobile robots play a crucial role (Ma et al., 2023; Zennaro et al., 2022). In the societal aspect, mobile robots are popular in assisting the aging population, creating a need for assistive robots in healthcare (Dilip et al., 2022), and elder care (Ahuja et al., 2022; Asgharian et al., 2022). Some of the advantages of mobile robots are that they can be deployed in hazardous environments or for dangerous tasks, reducing risks to human workers (Berx et al., 2022). As demand for convenience and efficiency by the consumer, personal service robots are gaining popularity for tasks like cleaning (Anl & Doğan, 2022; Bisht et al., 2022), delivery (Gehrke et al., 2023; Hakli & others, 2023), and entertainment (Ayad et al., 2023; Tran et al., 2023).

In addition, advancements in sensors used in mobile robots such as LiDAR, cameras, and other sensors have enhanced robots' perception of their surroundings, enabling more accurate navigation and interaction (Ferreira et al., 2023; Sharma et al., 2023; Yaqoob & Imran Sarwar Bajwa, 2023). The increased computational power and miniaturised processors in mobile robots allow them to perform complex calculations and decision-making in real-time (Rani et al., 2024). Moreover, improved battery capacity and efficiency enable mobile robots to operate longer without recharging, expanding their operational range in the environment (Farooq et al., 2023; Mikołajczyk et al., 2023). In advanced mobile robots, integrating artificial intelligence and machine learning algorithms empowers them to learn, adapt, and make intelligent decisions, leading to greater autonomy and flexibility (Maheswari et al., 2023). The convergence of technological advancements, economic pressures, and societal needs has created a good opportunity for expanding mobile robots. They are becoming increasingly capable, affordable, and adaptable, making them valuable tools across various industries and applications.

Converting a mobile robot into a self-navigating machine requires a comprehensive approach. In addition to meticulous selection of the hardware platform and assistive sensors, software development must be considered. Therefore, this work focuses on a Simultaneous Localisation and Mapping (SLAM) based technique for self-navigation mobile robots. This technique systematically generates an intricate environment map while the robot navigates and concurrently performs a path-planning algorithm to strategically identify the

*Corresponding author: dr.norashikinmt@uitm.edu.my

doi: http://dx.doi.org/10.4314/njtd.v18i4.2871

most efficient route to the designated locations (Tourani et al., 2022). Among various SLAM approaches, Gmapping and Hector SLAM stand out as popular choices within the Robot Operating System (ROS) framework (Mallma, 2024) for 2-dimensional Light Detection and Ranging (LiDAR) maps. GMapping, developed in 2007, based on the Rao-Blackwellized Particle Filter, remains one of the most widely used systems for robot applications. The system creates the grid-based map using a particle filter. It also needs laser data from the LiDAR and odometry data from the wheel's encoder (Thale et al., 2020). Moreover, as this approach considers the mobile robot's movement and the most recent sensor observation with odometry information, it decreases the uncertainty for the robot pose in the particle filter's prediction step. Unfortunately, the localisation accuracy using Gmapping is reduced when the robot's velocity increases (Chow et al., 2019).

On the other hand, Hector SLAM generates a map solely based on the data received from the LiDAR. It does not require any odometry data and relies only on the information from the laser scan matching approaches, which contrasts with the Gmapping (Skarga-Bandurova et al., 2019). While these algorithms have been extensively studied and applied in diverse scenarios, limited research exists comparing their performance on slippery surfaces. Such conditions, common in industrial and outdoor settings, can significantly affect odometry-dependent algorithms like Gmapping, where wheel slip may distort positional estimates. Conversely, Hector SLAM's independence from odometry makes it a promising alternative for such conditions. However, existing literature has not thoroughly explored a direct performance comparison under these specific circumstances. Addressing this gap is critical for optimising the deployment of SLAM algorithms in real-world applications involving challenging terrains. The choice of a slippery surface for this study is rooted in its relevance to real-world scenarios where low-traction conditions pose significant challenges to autonomous navigation. Surfaces such as wet tiles, icy roads, and polished industrial floors are common in environments where robots are deployed, including warehouses, healthcare facilities, and outdoor applications. These conditions can exacerbate the limitations of odometry-dependent SLAM algorithms like Gmapping, making it critical to evaluate their performance under such constraints.

This research aims to evaluate and compare the performance of Hector SLAM and Gmapping for autonomous navigation of a mobile robot on a slippery tiled surface. The study assesses these algorithms' accuracy using metrics such as absolute, relative, and percentage errors to identify their strengths and limitations. By focusing on the challenges posed by slippery surfaces, this work provides insights into the suitability of these algorithms for adverse environments, thereby bridging the identified research gap. Ultimately, the findings contribute to selecting SLAM algorithms tailored to specific operational conditions, enhancing the reliability and precision of autonomous navigation systems.

II. THEORETICAL ANALYSIS

This work uses a mobile robot named Bveeta Mini® R007, a two-wheeled robot with one dummy wheel, as shown in Figure 1. The mobile robot is equipped with two DC motors, encoders (front wheels), and one dummy wheel (rear wheel) to maintain the stability of its platform. Table 1 shows the parameters related to this robot. While on the other hand, the kinematic model of this mobile robot is shown in Figure 2. The kinematic model of a two-wheeled mobile robot is fundamental for understanding how the robot navigates its environment. This model provides insights into the robot's motion by describing the relationship between the wheel velocities and the robot's overall movement.

Table 1. Bveeta Mini® Type R-007 parameters

Parameter	Size (mm)
Robot's width, W	170
Wheel radius, R	33.5

According to the Kinematic Model in Figure 2, the mobile robot used in this work has linear velocities (robot movement in the X and Y axes) and angular velocities (rotation of the front wheels). Eqn. 1 describes the linear velocities of a two-wheeled mobile robot. Eqn. 1 and 2 describe the linear velocities (vr, vl) of the left and right wheels, respectively.

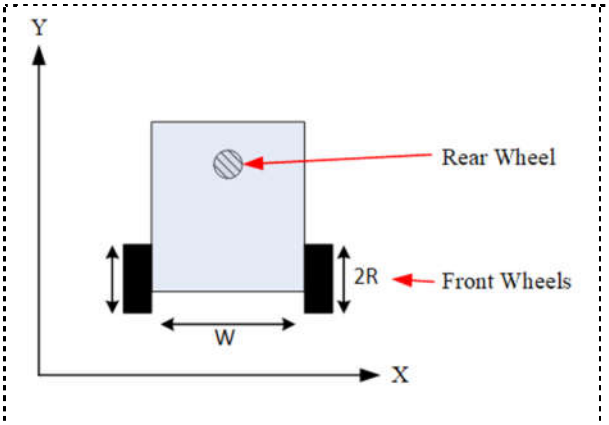

Figure 1. Bveeta Mini® Type R-007 robot schematic

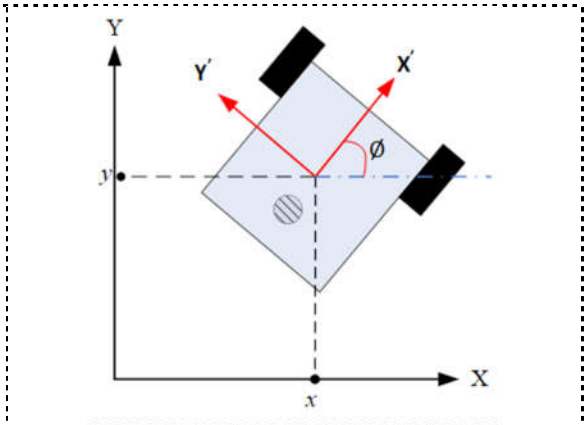

Figure 2. Kinematic model for Bveeta Mini® Type R-007 mobile robot

$$v_l = R \cdot \omega_l \quad (1)$$
$$v_r = R \cdot \omega_r$$

where,
v_l = linear velocity of the left wheel
v_r= linear velocity of the right wheel

The average velocity of the mobile robot (v_{robot}) can be calculated as in Eqn. 2, which is the simple mean of velocities in Eqn. 1. This average velocity helps determine the robot's forward or backward motion.

$$v_{robot} = \frac{v_l + v_r}{2} \quad (2)$$

If the linear velocities are projected along the x and y axes, Eqn. 3 is obtained. This projection is crucial because it allows us to understand how the robot moves in a 2D plane. Using trigonometric functions (cosine and sine) to project the velocities ensures we accurately account for the robot's orientation ($\emptyset$) at any given time.

$$X' = v_{robot} \cdot cos\,(\emptyset) = \frac{R}{2}[\omega_l \cdot cos(\emptyset) + \omega_r \cdot cos(\emptyset)\,] \quad (3)$$
$$Y' = v_{robot} \cdot sin\,(\emptyset) = \frac{R}{2}[\omega_l \cdot sin(\emptyset) + \omega_r \cdot sin(\emptyset)\,]$$

where,
X' = projected linear velocity in the *x*-axis
Y' = projected linear velocity in the *y*-axis

The angular velocity of the mobile robot is given by the difference of the wheels' linear velocities shown in Eqn. 4.

$$W \cdot \Delta_\emptyset = (r \cdot \omega_l) - (r \cdot \omega_r) \quad (4)$$

where,
$\Delta_\emptyset$ = displacement of $\emptyset$ in *z*-axis
r = the radius of the robot's wheel

Therefore, the displacement of the robot orientation can be calculated as in Eqn. 5. The angular velocity is derived from the difference in the linear velocities of the wheels, indicating how quickly the robot turns. The displacement of orientation is vital for SLAM, as it affects the robot's ability to map and navigate its surroundings correctly.

$$\Delta_\emptyset = \frac{(r \cdot \omega_l) - (r \cdot \omega_r)}{W} \quad (5)$$

where,
$\Delta_\emptyset$ = displacement of $\emptyset$ in *z*-axis

The kinematic model of the Bveeta Mini® Type R-007 two-wheeled mobile robot offers significant advantages for SLAM applications such as Gmapping and Hector mapping. One of the primary benefits is the precision in movement that the model provides. This precision is essential for SLAM algorithms, which rely on accurate data to construct reliable maps. By calculating exact velocities and orientations, the robot can navigate tight spaces and complex environments more effectively, ensuring high control over its movements. Real-time localisation is another crucial advantage offered by the kinematic model. SLAM algorithms require the robot's position and orientation continuously updated with sensor data. The kinematic model provides a robust framework for these real-time predictions, improving the robot's ability to adapt to dynamic environments and maintain an accurate map. This real-time adaptability is essential for effective SLAM performance.

The Bveeta Mini® Type R-007 robot also benefits from increased stability and manoeuvrability, including a dummy wheel for balance and two driven wheels. This configuration ensures the robot maintains stability and can manoeuvre smoothly, even on uneven surfaces or when encountering obstacles. This stability is particularly beneficial in SLAM experiments, where quick adjustments in direction are often necessary. Compatibility with SLAM algorithms is another significant advantage. Both Gmapping and Hector mapping benefit from the precise kinematic model. Gmapping, which uses a particle filter, relies on accurate motion models to predict the robot's position based on this kinematic model. The detailed kinematic equations improve the accuracy of these predictions. Hector mapping, which employs scan matching, relies on correct orientation and velocity data to align scans precisely. This compatibility enhances the overall performance of SLAM systems.

Moreover, the kinematic model contributes to improved mapping accuracy. The robot ensures that the generated map is as accurate as possible by maintaining a consistent path and avoiding drift, a common issue in SLAM. Precisely calculating linear and angular velocities reduces the need for extensive post-processing and correction, leading to more reliable maps.

However, several critical points need consideration for optimal performance. Sensor integration is essential, as the kinematic model's effectiveness is significantly enhanced. when integrated with high-quality sensors like LIDAR and encoders to ensure that it's predictions align with the robot's movements. Additionally, real-world environments introduce complexities such as wheel slips or obstacles. Incorporating adaptive algorithms that can account for these variations can further enhance performance. The choice between Gmapping and Hector mapping depends on the specific application. Gmapping might be preferred for environments where the advantages of a particle filter are pronounced, while Hector mapping excels in high-frequency LiDAR scan environments. Table 2 tabulates the specification of the sensors used in this work.

Table 2. Sensors specification

Sensor	Specification
LiDAR	Range Frequency: 5000Hz Range Distance: 0.12-10m Angle Resolution: 0.43-0.86 Degree Scan Frequency: 6-12Hz Scan Angle: 360 Degree
Encoder	Resolution: 2000 ppr

III. MATERIALS AND METHODS

Building on the robust ROS framework, this paper prioritises utilising existing ROS packages effectively for real-time simulation purposes.

A. Bveeta Mini Mobile Robot

Bveeta Mini® R007, as shown in Figure 3, is an open-source ROS-based mobile robot that can be programmable for use in teaching, research, and rapid product development. It is a two-wheeled mobile robot that can be controlled using a Linux-based laptop with ROS Melodic. This work focuses on studying the existing mapping techniques, which further improved in the particular code related to robot pose data during trajectory for analysis.

Figure 3. Bveeta Mini Type R-007 mobile robot with ROS framework.

B. Experimental Work

Figure 4 depicts the procedural steps involved in the manual mapping operation conducted by the Bveeta Mini ® mobile robot. Manual mapping is crucial as algorithms such as Gmapping and Hector SLAM necessitate a preliminary comprehension of the surroundings to initiate map construction. During the initial phase, the robot is physically manoeuvred in the region to gather sensor data necessary for the algorithm to begin mapping. The experiment is initially conducted to chart the environment manually. Upon activating the mobile robot, the system remains idle for a brief period to allow for the completion of hardware initialisation. After the indicator confirms the readiness of the LiDAR and USB motors, the roscore is run. Then, the teleop_keyboard.py, a Python code, is executed to start the manual mapping operation. Subsequently, the launch file is equipped with either Gmapping or Hector SLAM nodes activated. A Rosbag function is used to record the pose of the mobile robot in the Cartesian plane as it is being manually manoeuvred around. The map generated can be visualised using ROS Visualisation (Rviz). Once all areas have been thoroughly mapped and deemed satisfactory, the map is stored on a map server for future reference during autonomous navigation.

After the map is successfully generated, the next procedure is to obtain several location points for the autonomous navigation operation later. The mobile robot was manually guided to the specified location on the map. Once the robot reaches that point, a Publish Point menu in the Rviz is used and clicked on the robot's location. To get the coordinates of this point, rostopic echo/clicked_point is executed and saved in Sequential_Goal.py for later autonomous navigation operations. The process is repeated a few times to get different point locations for comprehensive experiments. The whole process of getting the point locations is shown in Figure 5.

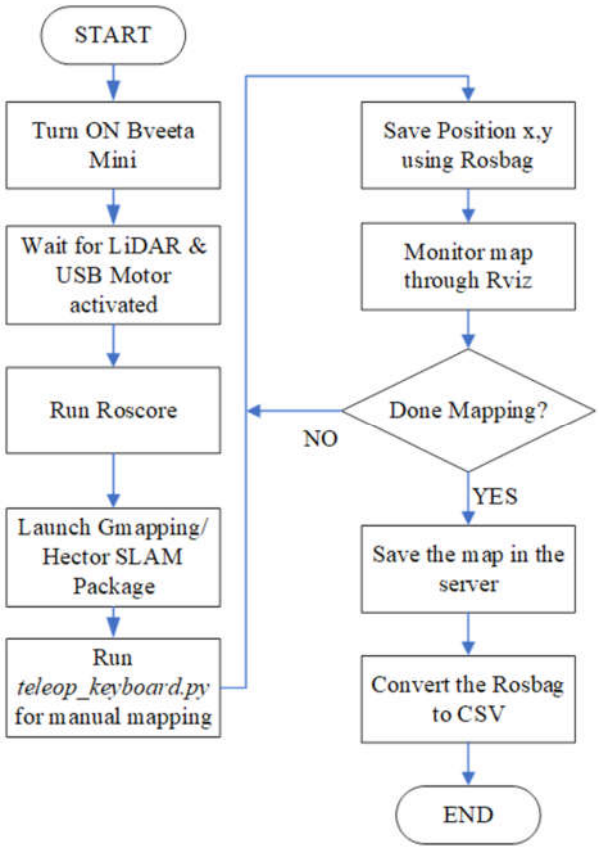

Figure 4. Experimental procedure to manually map the environment

The Dynamic Window Approach (DWA) local planner is used to test the autonomous navigation. It takes the mobile robot's position and maps information from SLAM as input. In this work, the inflation radius in the DWA local planner is adjusted to be 0.05, which creates a small buffer zone around the obstacles or walls so that the robot can navigate closer to them as the experimental area is small. In addition, a smaller inflation radius value allows the robot to achieve shorter paths and potentially faster navigation. However, this increases the risk of collisions if sensor data is not accurate or unexpected obstacles appear. Figure 6 shows the experimental procedures to autonomously navigate the mobile robot in the map generated by the *Gmapping* and *Hector SLAM*. The position of the mobile robot during the autonomous navigation is recorded

and converted to .csv format for easier data analysis. Figure 7 shows the experimental area chosen for this work. The experiment was done in a small tiled corridor area with an area size of 1.6m (W) x 0.7m(H).

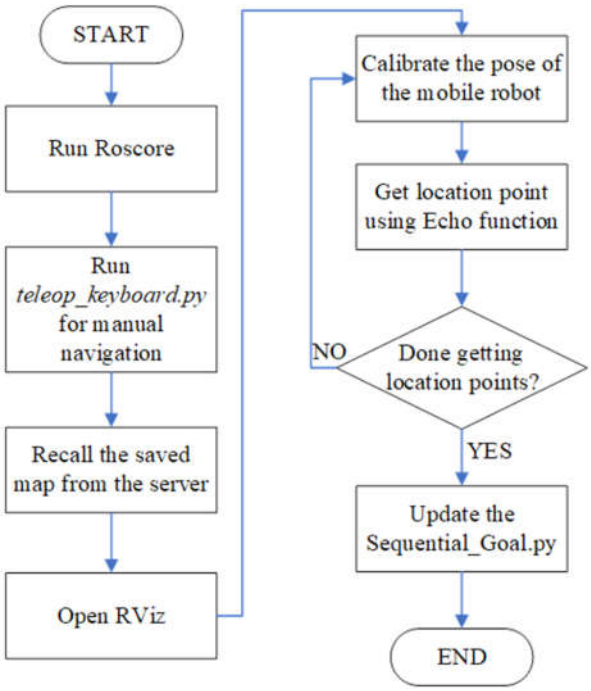

Figure 5. Experimental procedure to obtain location points in a Cartesian plane through Rviz.

C. Performance Analysis

An experimental setup is required to evaluate the performance of Gmapping and Hector Mapping, where the robot navigates a predefined environment manually and autonomously using each SLAM algorithm. The environment should have known landmarks and obstacles to provide a basis for accurate comparison. The actual path of the robot during manual navigation should be recorded precisely using ground truth data, which will then be compared with the paths generated by the SLAM algorithms.

Absolute error, absolute relative error, and percentage error are the key metrics used for this comparison. Absolute error measures the direct difference between the robot's actual position during manual and autonomous navigation, as provided by the SLAM algorithm. A lower absolute error indicates a higher accuracy in pinpointing the position of the robot in the map. It is calculated using Eqn. 6, where $y_{estimated}$ does the SLAM algorithm estimate the position and y_{real} is the actual position of the robot. Lower absolute error indicates higher accuracy in pinpointing the s location of the robot on the map. The absolute relative error captures the robot's pose estimation consistency over time. It calculates the difference between consecutive robot poses provided by the SLAM algorithm relative to the actual position. This metric is useful for understanding the stability and reliability of the SLAM algorithm's performance over time. It is calculated using Eqn. 7. The percentage error represents the relative error as a percentage, making it easier to interpret the magnitude of the error in a standardised format. It is calculated using Eqn. 8.

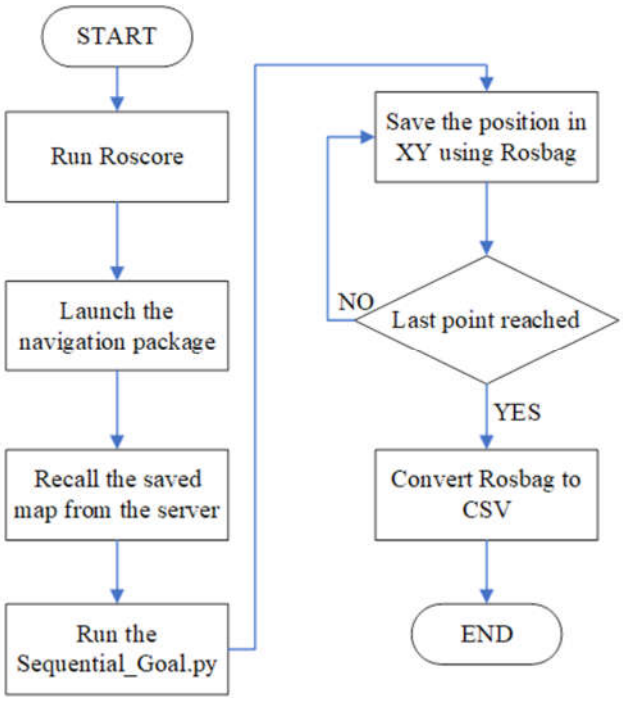

Figure 6: Experimental procedure to autonomously navigate Bveeta Mini using sequential goal operation.

Figure 7. The experimental work is done in a smaller corridor on a slippery surface

$$\varepsilon_{abs} = |(y_{estimated} - y_{real})| \quad (6)$$

$$\varepsilon_{rel} = \left|\frac{(y_{estimated} - y_{real})}{y_{real}}\right| \quad (7)$$

$$\varepsilon_{rel}(\%) = \left(\left|\frac{(y_{estimated} - y_{real})}{y_{real}}\right| \times 100\right) \quad (8)$$

Gmapping, which uses a particle filter approach, is known for its robustness in dealing with noisy data and dynamic environments. This algorithm can handle highly uncertain environments, making it suitable for applications where the robot encounters varying conditions. In environments with moderate noise and dynamic elements, Gmapping tends to

produce lower absolute error due to its ability to filter out noise effectively (Meng et al., 2020). It maintains consistent pose estimation over time, resulting in lower relative errors. The percentage error remains relatively low, indicating high accuracy and reliability in various conditions.

On the other hand, Hector Mapping, which relies on scan matching, is highly efficient in environments with high-frequency LIDAR scans. This algorithm excels in scenarios where rapid and accurate map updates are required (Mallma., 2024). Due to its precise scan-matching capabilities, Hector Mapping often produces a lower absolute error in high-resolution scan environments. However, Hector Mapping's performance may vary more in highly dynamic environments, potentially resulting in higher relative errors than Gmapping (Thale et al., 2020). The percentage error can be slightly higher in dynamic environments but remains competitive in stable environments with frequent scan updates.

Comparing the two, in relatively static environments with well-defined features, Hector Mapping may outperform Gmapping due to its precise scan matching, resulting in lower absolute and relative errors. However, Gmapping tends to be more robust in dynamic environments where noise and moving objects are present, resulting in lower percentage errors and more consistent performance. Hector Mapping is typically more computationally efficient, making it suitable for applications requiring real-time updates and high-frequency data processing. In conclusion, the performance analysis of Gmapping and Hector Mapping reveals that each SLAM algorithm has its strengths and weaknesses. Gmapping excels in dynamic environments with moderate noise, providing robust and consistent localisation and mapping. Hector Mapping, on the other hand, performs exceptionally well in static or semi-static environments with high-resolution LIDAR scans, offering precise and efficient mapping. By comparing the absolute error, absolute relative error, and percentage error, researchers can determine the most suitable SLAM algorithm for their specific application. Both algorithms provide valuable tools for autonomous navigation, with the choice largely depending on the particular requirements and conditions of the operating environment.

IV. RESULTS AND DISCUSSION

This section will describe the results and testing of the project. The results will be analysed and compared throughout the discussion. The difference between Hector SLAM and Gmapping was tested to see their effects on the performance of the robot.

A. Hector Mapping

Figure 8 shows the result of the GUI plugin for visualising the ROS computation graph, namely rqt_graph, to perform the Hector SLAM mapping from the collaborated nodes. The hector_mapping node is responsible for constructing the map by processing laser scan data and simultaneously tracking the robot's position, which is crucial for the navigation task. The robot_state_publisher, a sub-node of tf_static, published a fixed transformation between the laser scanner frame (in this work is the LiDAR sensor) and the base_link frame of Bveeta Mini. This transformation is crucial for accurately aligning the laser scans with the robot's base in the map, influencing the position and orientation of obstacles within the map. In this work, the retrieved raw scan data from LiDAR was at a rate of 11.5 Hz. The teleop_twist_keyboard node provides the velocity value to the Bveeta Mini's motor through the cmd_vel topic. As a result, the generated map of the experimental area is shown in Figure 9.

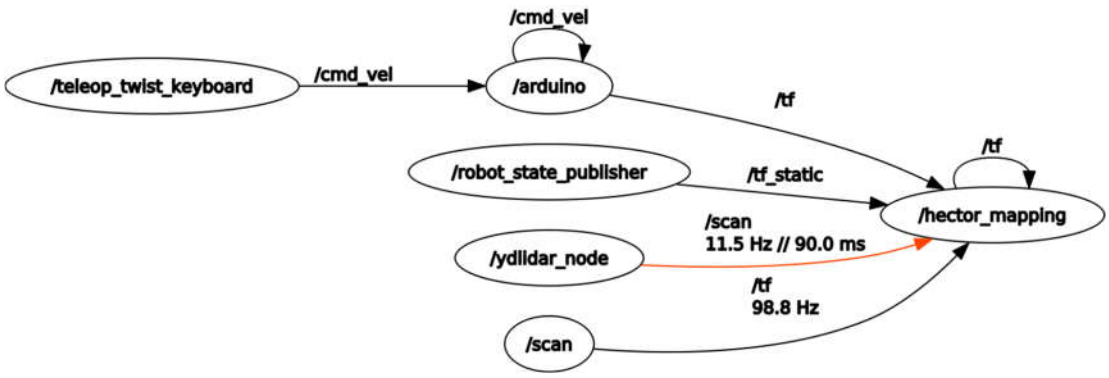

Figure 8. Rqt graph for autonomous navigation using Hector SLAM mapping.

The rectangular map produced by Hector SLAM likely represents the experimental area in this work, indicating that it could effectively track the robot's movements and correlate them with the laser scan data. Then, it compares consecutive laser scans to find the best possible alignment between them. By determining how much the robot needs to be moved (translated and rotated) to achieve the best alignment between the scans, Hector SLAM can estimate the change in the robot's pose. It iteratively updates the robot's pose estimation based on the scan-matching results. This update considers translations (movement along the x and y axes) and rotation (around the z-axis). The transform tree, tf, and tf_static keep track of coordinate frames from the continuous chain of transformations from the laser scanner to the robot's base, base_link, and ultimately to the global map frame. Hector SLAM calculates the laser scan's position relative to the map's coordinate frame by combining the estimated pose updates with the known transforms. This effectively determines the

robot's updated x and y coordinates within the generated map. The map consists of an occupied space (in blue) corresponding to the walls and obstacles detected by the LiDAR sensor during the mapping process. The remaining area appears white, signifying free space. However, in this work, the map lacks an obstacle in the middle of the experimental area due to the limited space.

In this work, the Hector SLAM primarily relies on one-line LiDAR data, which might not be able to capture intricate details or objects below a certain height. Overall, the Hector SLAM map captures the basic layout of the environment and might have slight inaccuracies due to the use of low-resolution LiDAR. Furthermore, the parameters within Hector SLAM, such as noise, must be filtered to create more accurate maps, especially in a complex environment. Nevertheless, the generated map can be used as a reference map for autonomous navigation using Hector SLAM, and its result is depicted in Figure 10. The errors comparing the robot's position during manual mapping and autonomous navigation using Hector SLAM are tabulated in Table 3.

f

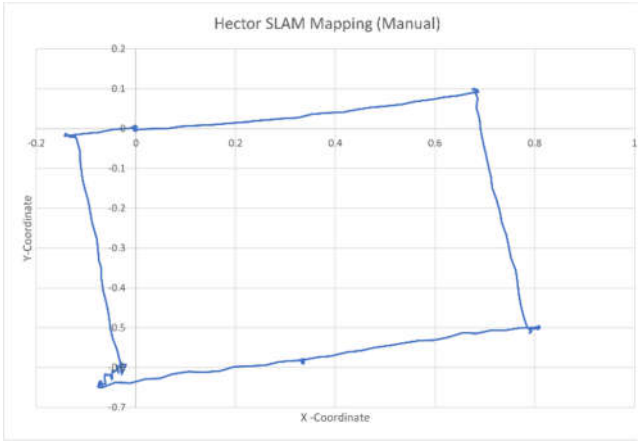

Figure 9. Manual Hector SLAM Mapping to create a reference map.

The result of autonomous navigation using Hector SLAM mapping shows a similar layout to the reference map in Figure 9 but with some prominent discrepancies, particularly in the route of Point A-B, Point B-C, and Point D-A. In these routes, the Hector SLAM map deviates from the straight lines of the reference map, exhibiting a more curved and irregular path. The computed errors between the manual map generation and the autonomous navigation in Table 3 confirm this deviation. The percentage of the absolute relative error, εrel (%), for Point A-B, Point B-C, and Point D-A is 142.9851%, 21.94870%, and 54.50178%, respectively. The LiDAR itself might be a plausible cause for this relatively large error. It can struggle with certain objects and surfaces. For example, it might have difficulty with highly reflective surfaces or objects with very thin edges. When this happens, it has difficulty matching the current map with the saved map at the beginning of the route. Therefore, it struggles to find and follow the mapped points between Points A – B. Furthermore, the slippery surface caused the robot to slip during navigation and deviate the robot from its original path. A huge discrepancy can be seen in Table 3, with the relative error of 142.9851%. When the robot finally arrived at point B, it successfully matched the point location of Point B from the current scan with the saved map. Therefore, the relative error was reduced to 21.9487% According to Figure 10, in the bottom left corner of the image, the reference map shows a right-angled corner, while the Hector SLAM map shows a more curved shape. This discrepancy could be caused by the LiDAR sensor struggling to precisely capture the corner due to its limitations and matching its position during the scanning.

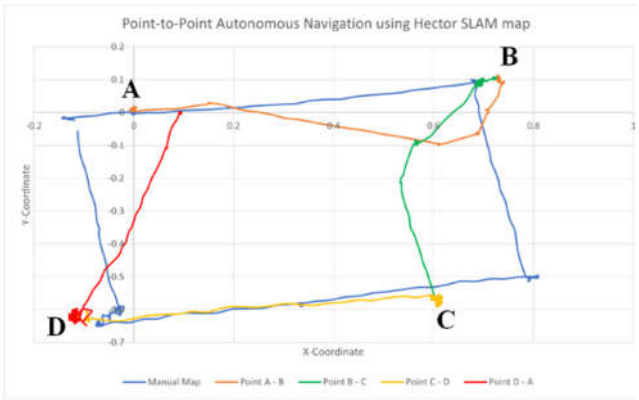

Figure 10. Autonomous navigation using Hector SLAM map.

Table 3 Numerical analysis of autonomous navigation using Hector SLAM

Navigation Path	ε_{abs}	ε_{rel}	$\varepsilon_{rel}(\%)$
Point A - B	0.0496	1.4299	142.9851
Point B - C	0.1638	0.2195	21.94870
Point C - D	0.0153	0.0270	2.698974
Point D - A	0.0455	0.5450	54.50178

Even though Hector SLAM mapping does not require odometry value during robot pose estimation, the sensor can accumulate errors over time, especially if the wheels slip or the terrain is uneven or slippery as tiles were the ground platform of the experimental area, which can be resulting a major drift to the mobile robot (Li et al., 2021). In the bottom right corner of the image, the reference map shows a straight line, while the Hector SLAM map shows a slightly curved path. Odometry errors during navigation could have caused this discrepancy. Furthermore, SLAM algorithms like Hector SLAM can be susceptible to errors due to constant iteration and map refining based on LiDAR. Like Hector SLAM, SLAM algorithms are constantly iterating and refining the map based on sensor data. However, they can be more susceptible to errors in complex environments or with limited sensor data. The scan matching problem in Hector SLAM is commonly described as a nonlinear optimisation problem (Yong et al., 2023). The deviation of the Hector SLAM map from the reference map in this work could be due to limitations in the Hector SLAM algorithm.

B. Gmapping

The rqt_graph for the Gmapping experiment is shown in Figure 11. Compared to Hector SLAM, the rqt_graph for Gmapping has more nodes. The slam_gmapping node requires data from LiDAR and odometry to determine the position of the mobile robot. Like Hector SLAM, the teleop_twist_keyboard node is reused to map the experimental area manually. Notably, the Gmapping algorithm publishes the robot's pose (position and orientation) information on the /tf topic, typically obtained from odometry. This information is essential for Gmapping to relate the laser scans to the robot's movement accurately. Figure 12 shows the generated map based on the manual navigation using the Gmapping algorithm.

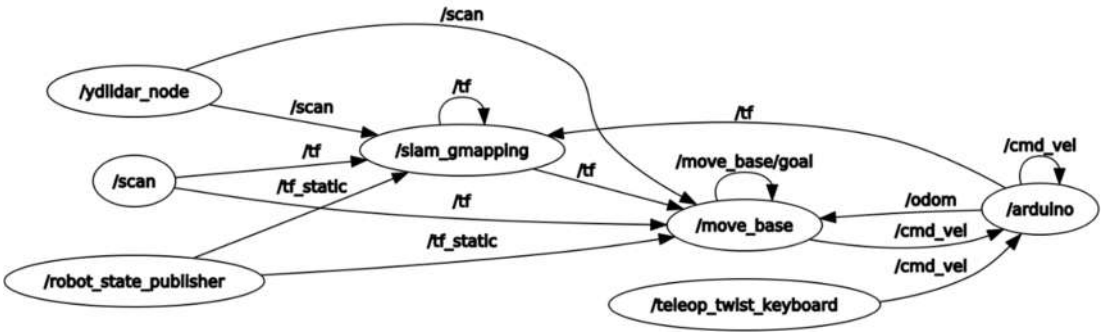

Figure 11. Rqt graph for autonomous navigation using Gmapping.

Like Hector SLAM, the Gmapping algorithm can also produce a rectangular map representing the shape of the experimental area. It indicates that Gmapping could effectively track the robot's movements based on the odometry data and correlate them with the laser scan data. Gmapping receives odometry data, which provides estimates of the robot's movement based on how much the robot's wheels have turned or changes in acceleration and orientation. Based on this odometry data, Gmapping generates an initial guess of the robot's pose (x, y, and orientation) relative to its starting position. At the same time, it receives laser scan data representing the environment around the robot. It compares the current laser scan against the partially built map and attempts to find the best match. This involves testing different translations (shifting the scan in x and y) and rotations to discover the most likely position where the scan originated. Gmapping combines the odometry-based initial estimate with the laser-scan-matching results using probabilistic techniques to calculate a refined estimate of the robot's pose. This refined pose is considered more accurate than relying on odometry alone, as it incorporates information about the environment's structure from the laser scans (Li et al., 2021).

Figure 13 shows the result of the generated map through autonomous navigation using the Gmapping algorithm. The robot's initial position largely deviates from the reference map with the discrepancies of 6550.482% from its relative starting point, which can be seen in Table 4. This massive deviation is likely due to the memory and computational constraints of the Gmapping algorithm, which uses fewer particles and does not identify loops in small-scale map construction (Meng et al., 2020). Furthermore, the Gmapping algorithm relies heavily on the information provided by the odometer (Tian et al., 2023). As mentioned before, odometry data can accumulate errors over time, as in this work, the surface of the experimental work is slippery due to the type of tiles used. When the robot's wheel's slip, the wheels' spin without effectively moving the vehicle forward. It distorts the data from the encoder, which will affect the wheel odometer data and ultimately lead to the unsatisfactory effect of the robot's mapping (Guo et al., 2022). As a result, the generated map using autonomous navigation through the Gmapping algorithm is unreliable due to the accumulated error in the robot's position. To overcome this issue, a filter algorithm can be used on the encoder data and integrated with the Inertial Measurement Unit (IMU) (Juharia et al., 2020).

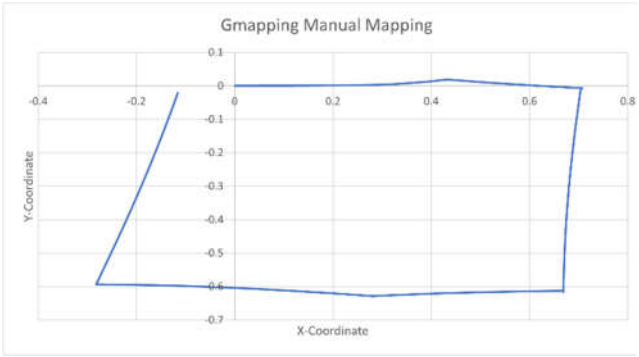

Figure 12. Manual Gmapping Mapping to create a reference map

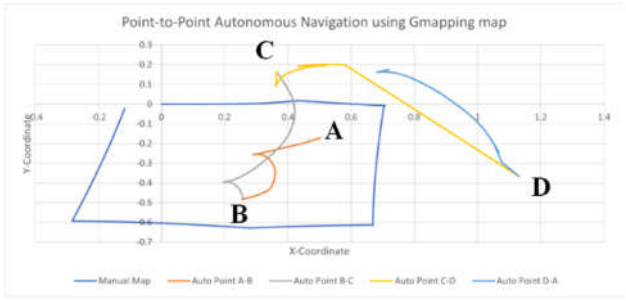

Figure 13. Autonomous navigation using Hector SLAM map

The experimental data offer a thorough comparative examination of the Hector SLAM and Gmapping algorithms for self-navigating a mobile robot on a slippery surface. The main goal was to evaluate the precision and dependability of these SLAM algorithms in difficult circumstances, employing metrics such as absolute error, absolute relative error, and percentage error.

Table 4 Numerical analysis of autonomous navigation using Gmapping

Navigation Path	ε_{abs}	ε_{rel}	$\varepsilon_{rel}(\%)$
Point A - B	0.2923	65.5048	6550.4820
Point B - C	5.2709	0.9578	95.7793
Point C - D	0.8030	1.3039	130.3900
Point D - A	1.2689	13.2588	1325.8832

The findings indicated that Hector SLAM outperformed Gmapping regarding accuracy across all navigation pathways. The average relative error for Hector SLAM was 55.53%, which, although greater than the desired level, was considerably reduced compared to the error rates of Gmapping. The exclusive utilisation of LiDAR data by Hector SLAM for map creation and robot positioning conferred a clear benefit in this experimental configuration. Hector SLAM achieved more consistent and dependable navigation performance by mitigating its reliance on odometry, which is prone to errors caused by wheel slippage on slippery surfaces. Nevertheless, the Hector SLAM method demonstrated significant constraints. Specifically, variations were noted in the routes between Points A and B, B and C, and D and A. These discrepancies are probably caused by the limits of the LiDAR sensor, which may have difficulty accurately detecting thin or shiny surfaces and might collect inaccuracies over time owing to environmental conditions. Furthermore, the iterative process of scan matching in Hector SLAM, in which the algorithm consistently improves the robot's estimation of its position, can result in inaccuracies if the initial conditions are not ideal or if the environment poses significant difficulties, such as reflective surfaces or narrow corners.

On the other hand, Gmapping exhibited substantial inaccuracies, especially in the initial position of the robot, with a concerning relative error percentage of 6550.482% for the navigation path between points A and B. The significant difference can be attributed to Gmapping heavily depending on odometry data. When a surface is slippery, the wheels can slip and cause inaccuracies in odometry measurements. These inaccuracies can result in significant mistakes in position, which Gmapping's probabilistic method cannot sufficiently correct. The considerable dependence on odometry for the initial determination of the robot's position implies that any inaccuracies in the readings from the wheel encoders directly affect the map's accuracy and the robot's perceived position.

The experimental results highlight the underlying difficulties of Gmapping when traction is hindered. Gmapping incorporates LiDAR data for environmental mapping; however, its reliance on odometry limits its ability to handle real-world situations that involve changing surface conditions effectively. The algorithm's computing constraints worsen this problem, requiring a trade-off between the particles utilised for mapping and the accuracy of loop closure and small-scale map generation.

The observed discrepancies between Hector SLAM and Gmapping are in line with their fundamental ideas from a theoretical standpoint. The architecture of Hector SLAM utilises high-frequency LiDAR scan matching, allowing it to accurately estimate its position without relying on odometry. This method is especially beneficial in situations where wheel slippage is common since it separates the robot's evaluation of its position from the sometimes inaccurate wheel encoder information. On the other hand, Gmapping's use of the Rao-Blackwellized Particle Filter, which combines odometry and LiDAR data, has both advantages and disadvantages. Although the integration can improve accuracy in stable conditions, it becomes a disadvantage in situations where odometry is not dependable. The experimental results indicate that Hector SLAM is a more appropriate option for situations that require reliable navigation on slick terrain.

Incorporating Inertial Measurement Units (IMUs) into SLAM algorithms like Gmapping can improve their resilience and yield more accurate posture estimation. Inertial Measurement Units (IMUs) can provide additional information to rectify inaccuracies in odometry, hence enhancing the overall precision of mapping. Moreover, the efficiency of Hector SLAM could be improved by including sophisticated noise filtering methods for LiDAR data, especially in situations characterised by reflecting surfaces or narrow corners. Ultimately, the selection of a SLAM algorithm should be based on the particular environmental factors and application needs, notwithstanding the advantages offered by both algorithms. Hector SLAM's autonomy from odometry renders it a more desirable choice for difficult terrains, while Gmapping could enhance its reliance on potentially inaccurate odometry data by employing sensor fusion techniques

V. CONCLUSION

This study evaluated and compared the performance of two widely used SLAM algorithms, Hector SLAM and Gmapping, in navigating a slippery tiled surface. The experiments revealed that Hector SLAM, which relies solely on LiDAR data, achieved higher accuracy in such low-traction conditions than Gmapping, which depends on odometry data. The findings highlight the limitations of odometry-reliant algorithms in environments where wheel slippage occurs, emphasising the need for alternative methods or sensor fusion approaches to mitigate these challenges.

While this research provides valuable insights into the capabilities and limitations of Hector SLAM and Gmapping under slippery conditions, the findings' generalizability is constrained by using a single surface type. To address this limitation, future studies should explore the performance of these algorithms across a range of surface conditions, such as sandy, gravel, or muddy terrains, as well as high-friction surfaces like asphalt or rubberised flooring. Such experiments

would provide a more comprehensive understanding of the algorithms' robustness and adaptability in diverse real-world scenarios.

Additionally, future work could investigate the integration of inertial measurement units (IMUs), advanced noise filtering techniques, and automated distance and area calculations to enhance the accuracy of both algorithms, particularly in environments with variable traction. By expanding the scope of experimental conditions and exploring advanced algorithmic modifications, subsequent research can contribute to developing more reliable and versatile autonomous navigation systems for mobile robots operating in complex and dynamic environments.

AUTHOR CONTRIBUTIONS

Muhammad Syafiq Ifwat Safizan: Methodology, Software, Validation, writing – original draft, Writing – review & editing**. Norashikin M. Thamrin:** Supervision, writing – original draft, Writing – review & editing**. Khairul Anuar Juhari:** Technical assistant, Industrial collaboration.

ACKNOWLEDGEMENT(S)

The authors thank the School of Electrical Engineering, College of Engineering, Universiti Teknologi MARA for providing the research facilities and publication fund. This research was not funded by any grant.

REFERENCES

Ahuja, G., Sharma, S., Sharma, M., & Singh, S. (2022). Assisted Living Robots: Discussion and Design of a Robot for Elder Care. International Conference on Internet of Things and Connected Technologies, 11–26, Singapore, Springer Nature Singapore.

Anl, E., & Doğan, H. (2022). Design and implementation of a cost effective vacuum cleaner robot. Turkish Journal of Engineering, 6(2), 166–177.

Asgharian, P., Panchea, A. M., & Ferland, F. (2022). A review on the use of mobile service robots in elderly care. Robotics, 11(6), 127.

Ayad, M., MacKay, J., & Clarke, T. (2023). Implementation of a Tag Playing Robot for Entertainment. Future of Information and Communication Conference, 416–426.

Berx, N., Decré, W., Morag, I., Chemweno, P., & Pintelon, L. (2022). Identification and classification of risk factors for human-robot collaboration from a system-wide perspective. Computers & Industrial Engineering, 163, 107827.

Bisht, R. S., Pathak, P. M., & Panigrahi, S. K. (2022). Design and development of a glass façade cleaning robot. Mechanism and Machine Theory, 168, 104585.

Bizbot Technology. (2024). Bveeta mini type R007. https://github.com/skj84/bveeta-R007. Assessed on August 10, 2023

Chahal, N., Bisht, R., Rana, A. K., & Srivastava, A. (2023). Robotic Arm: Impact on Industrial and Domestic Applications. Handbook of Computational Sciences: A Multi and Interdisciplinary Approach, 323–339.

Chow, J. F., Kocer, B. B., Henawy, J., Seet, G., Li, Z., Yau, W. Y., & Pratama, M. (2019). Toward underground localisation: Lidar inertial odometry enabled aerial robot navigation. ArXiv Preprint ArXiv:1910.13085.

Dilip, G., Guttula, R., Rajeyyagari, S., Pandey, R. R., Bora, A., R Kshirsagar, P., Sundramurthy, V. P., & others. (2022). Artificial intelligence-based smart comrade robot for elders healthcare with strait rescue system. Journal of Healthcare Engineering, 2022.

Farooq, M. U., Eizad, A., & Bae, H.-K. (2023). Power solutions for autonomous mobile robots: A survey. Robotics and Autonomous Systems, 159, 104285.

Ferreira, J. F., Portugal, D., Andrada, M. E., Machado, P., Rocha, R. P., & Peixoto, P. (2023). Sensing and Artificial Perception for Robots in Precision Forestry: A Survey. Robotics, 12(5), 139.

Gehrke, S. R., Phair, C. D., Russo, B. J., & Smaglik, E. J. (2023). Observed sidewalk autonomous delivery robot interactions with pedestrians and bicyclists. Transportation Research Interdisciplinary Perspectives, 18, 100789.

Guo, W., Qiu, J., Xu, X., & Wu, J. (2022). Talbot: A track-leg transformable robot. Sensors, 22(4), 1470.

Hakli, R., & others. (2023). Helping-as-Work and Helping-as-Care: Mapping Ambiguities of Helping Commercial Delivery Robots. Social Robots in Social Institutions: Proceedings of Robophilosophy 2022, 366, 239.

Javaid, M., Haleem, A., Singh, R. P., Rab, S., & Suman, R. (2022). Significant applications of Cobots in the field of manufacturing. Cognitive Robotics, 2, 222–233.

Juharia, K. A., Ramlib, R., Harisb, S. M., Ibrahimc, Z., & Mohamedd, A. Z. (2020). Development of Floor Mapping Mobile Robot Algorithm Using Enhanced Artificial Neuro-Based SLAM (ANBS). Jurnal Kejuruteraan, 3(1), 59–64.

Li, Z.-X., Cui, G.-H., Li, C.-L., & Zhang, Z.-S. (2021). Comparative Study of Slam Algorithms for Mobile Robots in Complex Environment. 2021 6th International Conference on Control, Robotics and Cybernetics (CRC), 74–79.

Ma, B. J., Kuo, Y.-H., Jiang, Y., & Huang, G. Q. (2023). RubikCell: Toward Robotic Cellular Warehousing Systems for E-Commerce Logistics. IEEE Transactions on Engineering Management.

Maheswari, B. U., Imambi, S. S., Hasan, D., Meenakshi, S., Pratheep, V. G., & Boopathi, S. (2023). Internet of things and machine learning-integrated smart robotics. In Global Perspectives on Robotics and Autonomous Systems: Development and Applications (pp. 240–258). IGI Global.

Mallma, A. (2024). A comparison study between RGB Camera based mapping and LiDAR based mapping algorithms using ROS and Gazebo. Proceedings of Umea's 27th Student Conference in Computing Science USCCS 2024, 73.

Mehrotra, T., & Shetty, S. (2023). An Innovation of Energy Harvesting for Small Scale Robotics in Automation Industry. 2023 International Conference on Distributed Computing and Electrical Circuits and Electronics (ICDCECE), 1–6.

Meng, Z., Wang, C., Han, Z., & Ma, Z. (2020). Research on SLAM navigation of wheeled mobile robot based

on ROS. 2020 5th International Conference on Automation, Control and robotics Engineering (CACRE), 110–116, IEEE

Mikołajczyk, T., Mikołajewski, D., Kłodowski, A., Łukaszewicz, A., Mikołajewska, E., Paczkowski, T., Macko, M., & Skornia, M. (2023). Energy Sources of Mobile Robot Power Systems: A Systematic Review and Comparison of Efficiency. Applied Sciences, 13(13), 7547.

Rani, S., Srivastava, G., & others. (2024). Secure hierarchical fog computing-based architecture for industry 5.0 using an attribute-based encryption scheme. Expert Systems with Applications, 235, 121180.

Sharma, N., Pandey, J. K., & Mondal, S. (2023). A review of mobile robots: Applications and future prospect. International Journal of Precision Engineering and Manufacturing, 24(9), 1695–1706.

Skarga-Bandurova, I., Krytska, Y., Shorokhov, M., Suvorin, O., Barbaruk, L., & Ozheredova, M. (2019). Towards development iot-based water quality monitoring system. Proceedings - 2019 International Conference on Future Internet of Things and Cloud Workshops, FiCloudW 2019, 140–145.

Thale, S. P., Prabhu, M. M., Thakur, P. V., & Kadam, P. (2020). ROS based SLAM implementation for Autonomous navigation using Turtlebot. ITM Web of Conferences, 32, 1011, EDP Sciences.

Tian, C., Liu, H., Liu, Z., Li, H., & Wang, Y. (2023). Research on Multi-Sensor Fusion SLAM Algorithm Based on Improved Gmapping. IEEE Access, 11, 13690–13703.

Tourani, A., Bavle, H., Sanchez-Lopez, J. L., & Voos, H. (2022). Visual SLAM: what are the current trends and what to expect? Sensors, 22(23), 9297.

Tran, D. T., Truong, D. H., Le, H. S., & Huh, J.-H. (2023). Mobile robot: automatic speech recognition application for automation and STEM education. Soft Computing, 1–17.

Yaqoob, I., & Imran Sarwar Bajwa. (2023). Sensor Fusion-Based Approach for Real Time Navigation in Autonomous Mobile Robots Using Mobile Stereonet in Warehouse. SSRN, 1–18.

Yong, Z., Renjie, L., Fenghong, W., Weiting, Z., Qi, C., Derui, Z., Xinxin, C., & Shuhao, J. (2023). An Autonomous Navigation Strategy Based on Improved Hector SLAM With Dynamic Weighted A* Algorithm. IEEE Access, 11, 79553–79571.

Zennaro, I., Finco, S., Calzavara, M., & Persona, A. (2022). Implementing E-commerce from logistic perspective: Literature review and methodological framework. Sustainability, 14(2), 911.