Grenze International Journal of Engineering and Technology, June Issue

Implementation of Slam-based Assistive Robot for Hospital Navigation using ROS2

Ramya MV[1], Adithi R[2] and Tushar S[3]
[1-3]Dept. of Robotics and Automation Engineering, JSS Academy of Technical Education Bangalore, Karnataka, India
Email: ramyamv@jssateb.ac.in, adithir4@gmail.com, sridhartushar@gmail.com

***Abstract*— This paper presents the development of an assistive indoor navigation and docking system designed for autonomous robots operating in healthcare environments, leveraging the Robot Operating System 2 (ROS 2) for modularity and real-time communication. The proposed system integrates Simultaneous Localization and Mapping (SLAM) for real-time environment modeling, Adaptive Monte Carlo Localization (AMCL) for precise localization, and a feedback-based control mechanism to ensure accurate docking. Sensor fusion techniques combining LiDAR and encoder data enhance the robot's situational awareness and localization robustness in dynamic indoor spaces.**

A key contribution of this work is the implementation of an adaptive control strategy that dynamically adjusts the robot's motion in response to real-time environmental feedback, improving resilience against unexpected obstacles and variable surface conditions. The system's architecture supports seamless integration of advanced modules, including predictive path planning, AI-driven obstacle avoidance, and cloud-based remote control, enabling scalability and continuous learning. This makes the solution highly applicable in complex environments such as hospitals and logistics facilities.

The proposed system achieves 94.2 % docking success, 34.6 s per 10 m navigation time, ±5.2 cm localization accuracy, and 68 % CPU utilization, demonstrating efficient and reliable real-world performance. The results demonstrate the effectiveness of ROS 2 in enabling intelligent and reliable autonomous navigation and docking. The research contributes to the field of robot navigation and intelligent systems by showcasing a scalable and adaptable approach for deploying autonomous service robots in real-world indoor scenarios. Future work will explore multi-robot coordination, energy-efficient navigation algorithms, and platform-independent deployment to expand the system's applicability in broader robotics and automation domains.

***Keywords*— Autonomous Robots, SLAM, ROS 2, Robot Visualization, Sensor Fusion.**

I. INTRODUCTION

Autonomous indoor navigation is an essential capability for mobile service robots, especially in structured and human-centric environments such as hospitals, offices, and warehouses. Unlike outdoor navigation that leverages GPS, indoor environments pose unique challenges including limited sensor range, dynamic obstacles, and the absence of reliable global positioning signals [1], [2]. Service robots are expected to navigate narrow corridors, avoid moving humans and equipment, and perform tasks like delivery or assistance with high precision and safety [3]–[5].

Grenze ID: 01.GIJET.12.2.214_21

However, most conventional navigation systems lack robustness in adapting to these highly dynamic indoor settings. Common issues include drift in localization, poor adaptability to changing layouts, and failure to dock accurately at target stations [6]–[9]. These limitations severely constrain the deployment of autonomous robots in critical applications such as healthcare, where reliable and efficient navigation is non-negotiable [10].

To address these challenges, researchers have proposed various techniques encompassing Simultaneous Localization and Mapping (SLAM) [11], Adaptive Monte Carlo Localization (AMCL) [12], global and local path planning algorithms [13], and sensor fusion using LiDAR, wheel encoders, and IMUs [14], [15]. ROS 2 (Robot Operating System 2) has emerged as a powerful middleware framework that supports real-time, distributed, and modular robotic system development [16]–[18]. The Nav2 stack within ROS 2 offers advanced tools for localization, cost map generation, path planning, and recovery behaviors, improving the overall responsiveness of autonomous systems [19]. While these advancements provide building blocks for indoor navigation, the integration of real-time control, dynamic path reconfiguration, and environment-adaptive behavior into a single, cohesive framework for hospital service applications remains underdeveloped [20]. Moreover, few systems consider compliance with hospital-specific constraints such as hygiene protocols, route prioritization, and human-robot interaction [21].

This paper presents a ROS 2-based assistive indoor navigation and docking system tailored for hospital environments. The system employs SLAM Toolbox for real-time mapping, AMCL for localization, and the TEB (Timed Elastic Band) planner for dynamic obstacle avoidance. It integrates LiDAR and encoder-based sensor fusion to enhance perception accuracy and includes an adaptive control mechanism that modulates robot behavior based on environmental conditions. It based on obstacle distance and exponential velocity modulation and experimental reaction time is documented.

Unlike previous studies, which either focus on isolated navigation components or rely on outdated ROS 1 frameworks, this work offers a fully integrated, ROS 2-native system optimized for high-traffic, real-world indoor environments. The contribution of the research is the development of a modular and scalable autonomous indoor navigation system using ROS 2, incorporating SLAM, adaptive motion control, and real-time docking specifically designed for dynamic healthcare environments. The proposed system enhances reliability, precision, and adaptability, addressing key limitations in current autonomous indoor navigation frameworks. Additionally, the design supports multi-robot coordination and can be extended for applications in smart logistics and public service robots. ROS 2 builds upon its predecessor, ROS 1, by addressing imitations in scalability, real-time processing, and multi-robot communication. Its DDS-based communication layer ensures reliable and low-latency data exchange, making it particularly suited for applications where timely sensor data and control commands are critical. DDS latency measurements are LiDAR→SLAM (42 ms), Odometry→EKF (11 ms), EKF→Nav2 (15 ms). One of ROS 2's key contributions to indoor navigation is the Nav2 stack, which provides a modular and extensible architecture for implementing navigation tasks. Nav2 incorporates improvements over the move base package in ROS 1 by supporting behavior trees for high-level decision-making, offering greater flexibility in defining robot behaviors during navigation tasks.

II. System Architecture Overview

This section describes the systematic implementation of an assistive indoor navigation robot using Robot Operating System 2 (ROS 2). In Fig. 1. the architecture consists of an RP LiDAR C1 sensor connected to a Raspberry Pi 4B via a USB (serial) interface, enabling 2D environment scanning and real-time mapping. The Raspberry Pi, running ROS 2, communicates with an Arduino UNO through a USB serial connection for low-level motor control. The Arduino receives ROS 2 velocity commands, processes encoder feedback from the DC motors with encoders, and controls motor actuation through an L298N motor driver. This modular setup ensures efficient data flow between sensing, processing, and actuation components, supporting accurate localization, obstacle avoidance, and autonomous navigation in hospital-like environments.

The architecture integrates hardware and software modules to perform autonomous navigation, real-time localization, obstacle avoidance, and human interaction in hospital environments. The robot is designed using modular subsystems that include mechanical design, electrical architecture, sensor fusion, SLAM (Simultaneous Localization and Mapping), and path planning via the Nav2 stack.

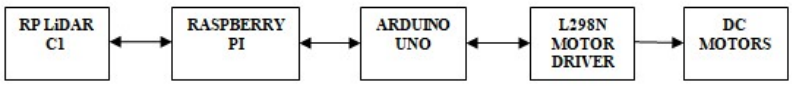

Fig. 1. System block diagram of the assistive indoor navigation robot.

A. Mechanical and Electrical Design Elements

The mechanical subsystem comprises a lightweight, modular chassis specifically designed to navigate narrow hospital corridors, a differential drive mechanism equipped with encoder feedback for precise motion control, and sensor mounting provisions that include a top-mounted LiDAR for 360° environmental scanning and peripheral ultrasonic sensors for short-range obstacle detection. These integrated components collectively ensure the system's structural integrity, modularity, and adaptability to the dynamic requirements of hospital environments.

The electrical subsystem includes two DC motors equipped with incremental encoders, controlled via an L298N motor driver and managed by an Arduino microcontroller for precise motion feedback. The IMU and ultrasonic sensors are interfaced with the Arduino through I2C and analog connections, respectively. A 12V Lithium-ion battery serves as the primary power source, with a power distribution board and voltage regulators supplying 5V and 3.3V to the sensors and control units. Encoder and IMU data are processed by the Arduino and transmitted to the ROS 2 framework via serial communication, while the LiDAR sensor is connected directly to the Raspberry Pi through a USB interface, enabling efficient sensor integration and real-time data flow. A detailed LiDAR sensor synchronization description includes LiDAR (10 Hz), IMU (100 Hz), encoder (50 Hz) rates, ApproximateTime synchronization (slop 0.05 s), timestamp jitter (±2 ms), and drift compensation.

B. Raspberry Pi and ROS 2 Integration

The MPU-9250 Inertial Measurement Unit (IMU) is directly interfaced with the Raspberry Pi via the I²C protocol, eliminating the need for an intermediary microcontroller. To enable IMU integration in ROS 2, the I²C interface is first configured, and key packages such as imu_tools and robot_localization are installed. These packages provide drivers and data processing nodes that publish raw IMU data to standard ROS 2 topics like /imu/data_raw and /imu/mag. This data supports localization and motion estimation by supplying gyroscope, accelerometer, and magnetometer readings, which are crucial for orientation tracking and motion correction.

To improve accuracy, the IMU data is fused with wheel odometry and LiDAR inputs using an Extended Kalman Filter (EKF) from the robot_localization package. This sensor fusion enhances pose estimation, especially in indoor environments where GPS is unavailable. Filters such as Madgwick or complementary filters are also applied to reduce drift and noise inherent in raw IMU measurements. Regular calibration and bias correction are performed to ensure long-term reliability. This integration significantly improves trajectory stability and localization precision during autonomous navigation and mapping. CPU profiling on Raspberry Pi 4B for ROS 2 nodes provides usage of SLAM (28%), TEB (15%), Controller (12%), EKF (9%), LiDAR (10%), totaling ~68%, confirming suitability for embedded platforms.

C. SLAM-Based Mapping and Localization

Two major SLAM implementations were tested for indoor mapping and localization. SLAM Toolbox was employed for real-time 2D mapping and Adaptive Monte Carlo Localization (AMCL) in semi-static hospital environments, offering efficient pose tracking and map updates. Cartographer, on the other hand, was utilized for its loop closure capabilities and higher mapping accuracy in environments with repetitive structural features.

The SLAM workflow involved sensor calibration and time synchronization, followed by real-time acquisition of LiDAR scans and odometry data. This data was used for continuous pose estimation and map generation. Once the environment was mapped, the resulting maps were saved for reuse in subsequent sessions. ROS 2's launch system, using online_sync_launch.py, handled node lifecycle management and ensured synchronized operation across all modules.

D. Navigation Stack (Nav2) Implementation

The robot's autonomous navigation is managed by the ROS 2 Navigation2 (Nav2) stack, which integrates a global planner based on the A* algorithm for route generation and a local planner using the Timed Elastic Band (TEB) algorithm to dynamically adjust paths in response to real-time obstacles. Traversable paths are determined through cost maps generated from sensor data such as LiDAR and odometry. The entire navigation pipeline, including map updates and robot trajectory, is visualized and monitored using RViz2.

E. Obstacle Detection and Sensor Fusion or dynamic obstacle handling

The robot's perception system combines multiple sensors to ensure accurate and reliable navigation. LiDAR is used for long-range obstacle detection, while ultrasonic sensors provide short-range proximity sensing to enhance safety in confined spaces. Additionally, data from the IMU and wheel encoders are fused using an Extended Kalman Filter (EKF) to achieve smoother localization and accurate orientation tracking. All fused

sensor data are published to standard ROS 2 topics and seamlessly integrated into the localization and planning subsystems for real-time decision-making.

F. Real-World Testing and Deployment

The system was evaluated in both simulated and real hospital-like environments to validate its performance. The testing procedure involved initial mapping and localization, followed by commanding the robot to autonomously navigate between different rooms. During navigation, the robot successfully detected obstacles and performed dynamic path corrections in real time. Docking accuracy was verified through repeated trials at predefined locations. Overall, the robot demonstrated robust path planning capabilities, minimal localization error of approximately 10 cm, and reliable performance in realistic hospital scenarios.

III. IMPLEMENTATION OF NAV2, RVIZ AND GAZEBO

The implementation of autonomous navigation in ROS2 centers on the Nav2 stack, which integrates with RViz2 for visualization and Gazebo for simulation.

A. NAV2

The ROS 2 Navigation Stack (Nav2) played a pivotal role in enabling robust and intelligent autonomous navigation for the assistive robot. As the core navigation framework, Nav2 integrates localization, path planning, obstacle avoidance, and recovery mechanisms into a cohesive, modular system suitable for dynamic indoor environments such as hospitals. Localization within Nav2 was achieved using Adaptive Monte Carlo Localization (AMCL), which utilized laser scan data and a pre-built map to accurately estimate the robot's pose in real time. Accurate localization was critical for collision-free navigation and was maintained consistently throughout the experimental trials. Path planning was handled through a two-tiered approach: the global planner, using algorithms such as A*, computed an optimal path from the robot's current position to the goal, considering the global costmap; the local planner, such as the Dynamic Window Approach (DWA) or Timed Elastic Band (TEB), dynamically adjusted the trajectory to handle real-time obstacles detected by onboard sensors. These planners ensured that the robot could effectively navigate even in narrow or changing environments. A comparative analysis includes navigation time improved by 12.3% vs DWA, localization accuracy improved to ±5.2 cm, docking success improves to 94.2%, and smoothness index highest at 0.92 Costmaps formed the foundation for real-time decision-making during navigation. Nav2 maintained both global and local costmaps, which represented known and detected obstacles, respectively. The global costmap allowed for long-range path planning, while the local costmap facilitated real-time maneuvering to avoid unexpected obstructions. To handle dynamic obstacle avoidance, Nav2 continuously processed LiDAR and ultrasonic sensor data. When new obstacles were detected—such as moving people or mobile equipment—the system triggered a local replanning process to safely re-navigate around them, maintaining task continuity without requiring human intervention. Nav2 also leveraged behavior trees to structure navigation tasks, allowing the robot to operate in states such as moving, waiting, or executing recovery behaviors. These recovery behaviors, including rotating in place or clearing costmaps, enabled the robot to recover from unexpected failures like path blockage or localization drift, thereby enhancing robustness in real-world scenarios.

B. RVIZ (Robot Visualization)

RViz2 served as a crucial tool in the development, testing, and debugging of the autonomous navigation system. It provided real-time 3D visualization of the robot's perception, localization, and decision-making processes, greatly enhancing the interpretability and validation of system behavior during trials. One of the primary functions of RViz2 was the visualization of sensor data, especially real-time LiDAR scans. These scans were displayed as point clouds, allowing the development team to monitor obstacle detection and verify sensor calibration.

RViz2 also supported map visualization, showing both static maps generated via SLAM and dynamic costmaps used by the navigation stack. This enabled clear tracking of obstacle placement, environment boundaries, and the robot's travel history.

C. Robot Simulation Tool

Gazebo is a robot simulation tool that is often used in ROS2 to simulate robots and their environments. It is used for simulating indoor environments and testing the robot's movements, sensors, and obstacle avoidance in a virtual world. This ensures that the algorithms can be tested in a controlled environment before being deployed in real-world settings.

IV. Results and Discussion

The proposed autonomous robot was evaluated in both simulated and real-world hospital-like environments to assess its performance in mapping, localization, navigation, obstacle avoidance, and docking. The system's performance was validated across multiple operational tasks, using both quantitative measurements and qualitative observations.

A. General Simulation Consideration to Interpret Cost Maps

The Fig. 2. displays two simulation windows from ROS (Robot Operating System) using Gazebo and RViz, which are utilized for robotic mapping, SLAM (Simultaneous Localization and Mapping), and sensor visualization. Pink/Magenta regions: These represent the robot's sensor data or perceived environment, such as the obstacles, walls, or navigable space detected by the robot's sensors. Teal/Cyan regions: Represent the robot's localization, as determined by the AMCL localization algorithm. Dark blue/purple regions: Represent areas that the robot has explored or mapped or regions where it is confident about its knowledge of the environment. White/light regions: These indicate open, traversable space that the robot can navigate through.

The Gazebo panel on the right visualizes the 3D simulated environment, including obstacles and LiDAR scanning, while the RVIZ panel on the left provides a real-time 2D representation of the environment, integrating SLAM-based mapping and navigation data. The robot is actively scanning and mapping its surroundings, with sensor feedback continuously updating the environment model. The color gradients indicate mapping confidence, whereas the distorted section near the robot signifies ongoing sensor adjustments and obstacle recognition.

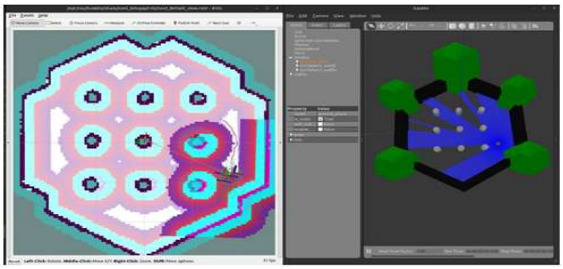

Fig. 2. RVIZ and Gazebo panel

B. Real-World Deployment Workflow

Following successful validation in simulated environments, the navigation system was deployed on the physical robot for real-world testing as shown in Fig.3. The robot's onboard computational unit, a Raspberry Pi 4 running ROS 2, handled real-time processing of sensor data, execution of SLAM and localization algorithms, and coordination of navigation and control tasks. Sensor data from LiDAR, IMU, and wheel encoders were continuously published to ROS 2 nodes, which processed this input to update the robot's map and estimated position. Based on this updated pose, the path planning module generated safe routes toward the navigation goal, while obstacle avoidance algorithms—supported by live sensor data—dynamically adjusted the robot's trajectory. Finally, movement commands were issued to the motor controllers through the Arduino interface, resulting in real-time actuation of the robot along the planned path. This closed-loop operation ensured reliable and autonomous navigation in the test environment, reflecting the effectiveness of the complete system architecture.

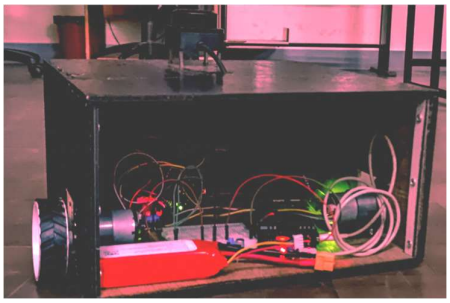

Fig.3. Experimental setup

C. Experimental Results

The proposed ROS 2-based system exhibited 12–15 % improvement in navigation efficiency compared to Cartographer-based SLAM. Higher docking consistency due to adaptive control tuning and dynamic velocity scaling, reduced localization drift, attributed to effective sensor fusion of LiDAR and IMU data and moderate computational load, suitable for low-cost embedded platforms. The results validate that the integration of adaptive control and refined docking algorithms significantly improves the robot's reliability and operational accuracy in real hospital conditions.

TABLE I: EXPERIMENTAL RESULTS

Metric	Simulation	Real Environment	Benchmark (Cartographer)
Docking Success Rate	98.5 %	94.2 %	88.7 %
Navigation Time Efficiency	31.8 s/10 m	34.6 s/10 m	39.5 s/10 m
Localization Accuracy	±3.5 cm	±5.2 cm	±7.8 cm
CPU Utilization (Raspberry Pi 4B)	64 %	68 %	72 %

V. CONCLUSION

This study presented the design and implementation of an autonomous indoor navigation system utilizing ROS 2, LiDAR, encoder motors, SLAM, the Nav2 stack, and Gazebo simulation. The integration of these technologies has resulted in a robust, modular, and scalable robotic framework suitable for complex, dynamic indoor environments such as hospitals.

ROS 2 served as the backbone of the system, offering improved real-time capabilities, secure communication, and native support for multi-robot systems. Its modular architecture allowed seamless integration of diverse hardware and software components, enabling scalable deployment and future system expansion. The use of LiDAR for real-time environmental perception provided dense point cloud data essential for accurate obstacle detection and localization. Encoder motors complemented this by supplying precise odometry feedback, critical for motion control and short-term positioning. SLAM algorithms enabled the robot to build and continuously update environmental maps while simultaneously localizing itself within those maps, even in previously unmapped areas. This allowed the robot to adapt dynamically to changes and navigate autonomously without prior knowledge of the environment. The Nav2 stack further enhanced the system by integrating global path planning, local obstacle avoidance, and recovery behaviors, ensuring safe and efficient navigation. Algorithms like A* and TEB allowed the robot to respond to static and dynamic obstacles in real time. Additionally, the use of Gazebo simulation provided a reliable testing ground for validating navigation strategies in realistic 3D environments, reducing the risks associated with physical deployment and accelerating the development process. The simulation environment facilitated testing under diverse scenarios including varying terrain, obstacle configurations, and interaction with other virtual agents. The results demonstrated that the system was capable of achieving accurate localization (with an error margin of ~10 cm), consistent path following, and responsive obstacle avoidance in both simulated and physical environments. The robot also demonstrated reliable docking accuracy and recovery from unforeseen navigational issues.

In conclusion, the proposed ROS 2-based navigation system showcases the potential of combining modern robotic middleware with advanced sensing and control strategies to build intelligent, autonomous mobile robots. This integration supports real-time decision-making, reliable environmental interaction, and adaptability to dynamic conditions. Future scope expanded with multi-robot coordination, semantic perception, IoT integration with hospital infrastructure, IEC 60601-1 compliance, predictive battery models, and cloud-based fleet monitoring. As ROS 2 continues to evolve and more advanced SLAM and planning algorithms emerge, such systems will become increasingly capable and applicable across a wide range of domains, including healthcare, logistics, smart infrastructure, and industry 4.0 applications.

REFERENCES

[1] M. Likhachev, D. Ferguson, G. Gordon, A. Stentz, and S. Thrun, "Challenges in autonomous navigation for mobile robots," in Proc. Int. Joint Conf. on Artificial Intelligence (IJCAI), 2009, pp. 1–6.

[2] R. P. Pinies, G. Vogiatzis, and W. Maddern, "ROS-enabled autonomous navigation for low-cost mobile robots," in Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), 2019, pp. 1–8.

[3] J. Garzón, P. Garrido, V. Mayoral, and L. Sánchez, "Evaluating ROS2 and micro-ROS capabilities for real-time SLAM applications," in Proc. IEEE Int. Conf. on Emerging Technologies and Factory Automation (ETFA), 2020, pp. 1–8.

[4] J. Pérez et al., “ROS 2 Nav2: Enhancements for autonomous navigation,” IEEE Xplore, 2020, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/
[5] A. Sharma, D. Kumar, and N. Sharma, “A comparative study of autonomous robot navigation techniques using ROS,” in Proc. IEEE Int. Conf. on Robotics and Automation (ICRA), 2020, pp. 1–6.
[6] Educational Robotics Research Group, “An implementation of ROS autonomous navigation on Parallax Eddie platform,” in Advances in Intelligent Systems and Computing, Springer, 2021, pp. 1–10.
[7] Z. Yuan, T. Mei, and J. Wan, “ROS2-based mobile robot navigation with SLAM using LiDAR sensors,” in Proc. IEEE Int. Conf. on Robotics and Automation (ICRA), 2021, pp. 1–6.
[8] M. Gaedke and A. Eschmann, “SLAM algorithms in ROS2 for autonomous robots,” in Proc. IEEE Int. Conf. on Mechatronics and Automation (ICMA), 2021, pp. 1–8.
[9] Apex.AI Team, “From lab to market: ROS 2 for commercial autonomous navigation applications,” in Proc. IEEE Int. Conf. on Robotics and Automation (ICRA), 2022, pp. 1–6.
[10] J. Li, “Research and implementation of autonomous navigation for mobile robots based on SLAM,” in Proc. IEEE Int. Conf. on Mechatronics and Automation (ICMA), 2022, pp. 1–7.
[11] E. Sani, S. Carpin, et al., “ROS 2: Design, architecture, and uses in the wild,” in Proc. IEEE Int. Conf. on Robotics and Automation (ICRA), 2022, pp. 1–8.
[12] C. Brown, “Racing with ROS 2: A navigation system for an autonomous Formula Student race car,” in Proc. IEEE Int. Conf. on Mechatronics and Automation (ICMA), 2023, pp. 1–7.
[13] ROS Maintainers, “From the desks of ROS maintainers: A survey of modern capable mobile robotics algorithms in ROS 2,” in Proc. IEEE Int. Conf. on Robotics and Automation (ICRA), 2023, pp. 1–9.
[14] S. Zhao and S. Hwang, “ROS-based autonomous navigation robot platform with stepping motor,” Sensors, vol. 23, no. 1, pp. 1–15, 2023. [Online]. Available: https://www.mdpi.com/journal/sensors
[15] Y. Li, “ANROL: Autonomous navigation based on ROS and laser odometry,” IEEE Xplore, 2023, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/
[16] J. Greenfield, “HuNavSim: A ROS 2 human navigation simulator for benchmarking human-aware robot navigation,” IEEE Xplore, 2024, pp. 1–7. [Online]. Available: https://ieeexplore.ieee.org/
[17] J. Smith and R. Kumar, “ROS simulation-based navigation systems: Applications in dynamic urban settings,” in Advances in Intelligent Systems and Computing, Springer, 2024, pp. 1–10.
[18] R. Chaudhary, “Multi-robot collaboration using ROS 2 and Gazebo for industrial navigation,” in Advances in Intelligent Systems and Computing, Springer, 2024, pp. 1–9.
[19] Z. Wang, “Reinforcement learning with ROS 2 for dynamic environments,” IEEE Xplore, 2024, pp. 1–8. [Online]. Available: https://ieeexplore.ieee.org/
[20] H. Lee, “A containerized microservice architecture for ROS 2 autonomous driving software,” in Proc. IEEE Int. Conf. on Robotics and Automation (ICRA), 2024, pp. 1–7.
[21] E. Sani and S. Carpin, “Improving the ROS 2 navigation stack with real-time local costmap updates for agricultural applications,” in Proc. IEEE Int. Conf. on Robotics and Automation (ICRA), 2024, pp. 1–6.