Autonomous Navigation for Differential Drive Robots: Grid-Based Fastslam with AMCL in ROS2

Victor Nosakhare Oriakhi

Robotics and Automation, University of Salford

DOI : https://doi.org/10.51583/IJLTEMAS.2025.1401016

Received: 26 January 2025; Accepted: 31 January 2025; Published: 07 February 2025

Abstract: The research aims to establish a robust autonomous navigation system for differential drive mobile robots, leveraging URDF modeling, SLAM techniques, AMCL, Gazebo simulation, and RViz2 visualization. The central objective is to enable robots to autonomously perceive their surroundings, construct accurate maps, self-localize, and navigate. The integration of URDF defines robot attributes, SLAM produces precise maps, and AMCL ensures reliable localization. Gazebo facilitates testing, while RViz2 provides real-time visualization. The outcome is an efficient navigation system empowering robots to independently navigate intricate environments. Beyond warehousing, applications span service robotics, exploration, and environmental monitoring. The research's significance lies in addressing a fundamental robotics challenge, advancing autonomous mobility across sectors. The approach's efficacy is validated through testing, with potential contributions to robotics research and r eal-world applications. achieved 97% mapping accuracy with a 5% deviation in simulated environments.

Keywords: SLAM, Gazebo, AMCL, Localization, Mapping, Visualization, ROS2

I. Introduction

Autonomous Mobile Robots (AMRs) have truly revolutionized automation, opening up exciting new possibilities across various industries. These remarkable robots, armed with sensors, actuators, and advanced control algorithms, possess the remarkable ability to independently explore their environments, plan the most efficient routes, and seamlessly carry out tasks. Their applications span a wide spectrum, including manufacturing, healthcare, agriculture, military operations, retail, and exploration, revolutionizing the way tasks are performed and data is acquired.

1

Figure 1: Mobile Robot. Source Katte 2018

This project focuses on a specific facet of autonomous robotics: the development of navigation systems for differential drive mobile robots. These robots, known for their maneuverability and versatility, hold promise in domains like logistics, surveillance, and exploration. To harness their full potential, integration of key technologies becomes paramount.

Problem Statement

A critical challenge lies in creating reliable autonomous navigation systems for differential drive mobile robots. Existing approaches do not fully integrate ROS2, FastSLAM, and AMCL to enhance localization, mapping accuracy, and navigation efficiency. Without a cohesive solution, robots risk errors, inefficient path planning, and collisions, impeding their ability to function optimally in complex and dynamic environments. This shortcoming is particularly impactful given the rising demand for automation across industries.

The significance of addressing this problem cannot be overstated. Failure to provide a comprehensive solution inhibits the realization of efficient and safe autonomous navigation, limiting the potential of these robots in diverse applications. The consequences extend to compromised operational efficiency, increased costs, and missed opportunities for innovative applications.

Aims and Objectives

This research project is dedicated to bridging the current gap in autonomous navigation for differential drive mobile robots. The primary objective is to demonstrate the effectiveness of integrating Grid-Based FastSLAM with AMCL within the ROS2 framework, alongside URDF-based modeling. This integration is intended to empower robots with the capability to precisely perceive their surroundings, construct detailed maps, continuously self-localize, navigate autonomously, and adeptly avoid obstacles.

The research objectives are outlined as follows:

Identify areas within the implemented system that may necessitate further research and refinement to enhance the functionality and effectiveness of the autonomous navigation system for differential drive mobile robots.

Conduct a comprehensive literature review to identify a suitable SLAM algorithm and localization method for the development of an autonomous navigation system for differential drive mobile robots, focusing on the integration of Grid-Based FastSLAM, AMCL, and URDF within ROS2.

Implement the chosen Grid-Based FastSLAM algorithm and AMCL method within the ROS2 framework and the Gazebo simulator as part of the autonomous navigation system's development.

Evaluate the performance of the integrated Grid-Based FastSLAM and AMCL system in diverse simulated and visualization environments. Particular attention will be given to key metrics, including localization accuracy, map quality, computational efficiency, and obstacle avoidance capabilities derived from command velocities.

The successful completion of these objectives will establish an innovative and dependable navigation system, leveraging Grid-Based FastSL AM and AMCL integration. This system is poised to elevate the capabilities of differential drive mobile robots, offering practical applications and benefits across various industries and sectors.

II. Literature Review

Introduction:

In recent years, the field of autonomous robotics has experienced remarkable advancements, revolutionizing various industries through breakthroughs in navigation, obstacle avoidance, simultaneous localization, and mapping (SLAM), and map generation techniques. These advancements have empowered robots to operate independently in complex and dynamic environments, with applications spanning forestry, healthcare, agriculture, construction, and general robotics. This literature review aims to explore the key studies and research conducted in these domains, with a specific focus on the progress made in autonomous navigation and the development of comprehensive navigation systems for mobile robots.

Within the field of autonomous navigation, a multitude of studies have significantly contributed to its advancement. This review will delve into the exploration of innovative sensor technologies employed to accurately perceive the environment. These sensors, which encompass lidar, cameras, and inertial measurement units (IMUs), serve as critical sources of information for robots, enabling them to develop a holistic comprehension of their surroundings and make well-informed choices as they navigate. Moreover, in this review, we'll delve into the progression of SLAM algorithms, which have played a central role in empowering robots to construct intricate maps of their environment while precisely determining their own location within these maps. Through this review, it is anticipated that new contributions will be made to the field of autonomous robotics, driving further advancements in mobile robot navigation, and setting the stage for future developments.

Mapping:

In recent years, the field of mapping in robotics has witnessed significant advancements and comprehensive studies aimed at enhancing mapping capabilities have emerged. (Achour et al., 2022) conducted a comprehensive survey focusing on semantic mapping in the context of both single mobile robots in indoor environments and collaborative mobile semantic mapping. Their survey encompassed a review of contemporary solutions for acquiring semantic data, including techniques such as object detection/recognition, deep-learning-based segmentation, and innovative human-robot interaction approaches. Additionally, the authors delved into reasoning-based acquisition methods, emphasizing the capability of robots to infer new information based on acquired data and a knowledge database. While highlighting the substantial progress made in semantic mapping, this survey also shed light on the challenges that persist, such as semantic data gathering, map representation, environmental considerations, and the complexities of collaboration among mobile robots.

In parallel to the investigation into semantic mapping, (Qu et al., 2021) contributed to the mapping literature by conducting a comparative analysis of three 2D Simultaneous Localization and Mapping (SLAM) algorithms: Gmapping, Hector SLAM, and Cartographer. Their study centered on mapping performance and involved the evaluation of various sensor combinations, including LIDAR, stereo cameras, and IMU. The findings were noteworthy, as they revealed that the Cartographer algorithm outperformed the others, excelling in the generation of accurate and informative maps with minimal errors. Additionally, the researchers identified the superiority of LIDAR scans over ZED scans as input for mapping algorithms, and they emphasized the importance of sensor fusion, particularly the combination of LIDAR scans, ZED odometry, and IMU data, in achieving optimal mapping results.

Building upon this exploration of SLAM algorithms, (Yagfarov et al., 2018) conducted a study comparing three popular ROS-based 2D SLAM libraries: Google Cartographer, Gmapping, and Hector SLAM. Their evaluation employed precise ground truth data to assess map accuracy. The results consistently favored Google Cartographer as the top performer in producing highly accurate maps, followed by Gmapping. In contrast, Hector SLAM, relying solely on LIDAR data and lacking loop closure mechanisms, yielded less accurate results. The implications of this research were clear, suggesting Google Cartographer as a preferred choice for generating high-quality 2D maps with LIDAR on mobile robots.

Extending beyond the realm of robotics but still relevant to mapping, (Cho, Lee, & Kim 2023) introduced a novel approach to autonomous vehicle motion planning using occupancy grid maps. Their method incorporated a single constraint based on the occupancy grid map, permitting traversal only in cells with values below a predefined threshold. This approach simplified complex motion planning problems and reduced computational costs by integrating the constraint into the motion planning algorithm through a single barrier state. Real-time simulations demonstrated the tangible benefits of their approach, including improved safety, reduced time costs, and heightened robustness compared to other methods.

These studies collectively contribute to the understanding of mapping in mobile robotics. The comparative analysis of SLAM algorithms provides insights into the most effective algorithms and sensor combinations for accurate and informative maps. The findings from both studies have significant implications for improving mapping capabilities in mobile robotics, paving the way for future advancements in the field.

Localization:

The field of mobile robotics has seen substantial advancements in the area of localization. Researchers have been hard at work, aiming to boost precision, resilience, and adaptability in this crucial aspect of robotics. Among the various localization methods explored, the Adaptive Monte Carlo Localization (AMCL) algorithm stands out as a noteworthy approach. (Baek et al., 2022) contributed to the domain by developing a 3D global localization method tailored specifically for underground mines. They harnessed mobile LiDAR mapping and point cloud registration techniques to achieve precise matching of local point cloud datasets. This innovation holds tremendous promise for enhancing 3D position recognition in the challenging and dynamic environments of underground mines.

For mobile robots, maintaining global pose awareness is crucial, especially when faced with scenarios such as kidnap recovery. (Li et al., 2023) proposed an ingenious solution based on ultra-wide-band technology. Their adaptive Monte Carlo localization algorithm leveraged ultra-wide-band modules and incorporated obstacle noise within global grid maps, significantly improving the robot's chances of swiftly recovering its global pose when lost or kidnapped.

In the realm of agricultural robotics, (Raikwar et al., 2023) addressed the limitations of relying on Global Navigation Satellite Systems (GNSS) for navigation. Their approach involved the implementation of a 2D LIDAR SLAM-based localization scheme, achieving remarkable positioning accuracy of approximately 0.2-0.3 meters. This breakthrough empowers autonomous navigation in static agricultural settings, independent of GNSS signals.

Efficient localization techniques play a pivotal role in logistics, particularly in the warehousing domain. (Tripicchio et al., 2022) conducted a comprehensive study comparing different methods for 3D multilateration of passive UHF RFID tags. Their research demonstrated centimeter-level accuracy for 3D localization and sub-centimeter accuracy for 2D localization, offering valuable insights into optimizing localization techniques in warehouse logistics.

In the context of AMCL, (Wang et al., 2018) addressed critical challenges such as initial location speed, stability, and robustness. Their proposed algorithm, grounded in a UWB array, aimed to enhance the performance of the Adaptive Monte Carlo Localization algorithm. This advancement holds great promise for improving the localization of mobile robots.

(Singh et al., 2023) introduced a robust methodology for implementing localization and navigation schemes in autonomous mobile robots. By harnessing the Robot Operating System (ROS) framework and integrating various ROS packages and algorithms, their approach achieved highly accurate pose identification, mapping, and robot control, both in simulations and real-world environments.

While these diverse localization methods contribute significantly to the field of mobile robotics, our research design will primarily emphasize the utilization of the Adaptive Monte Carlo Localization (AMCL) algorithm. Building upon AMCL's strengths and integrating it with other relevant techniques, our aim is to create a robust and adaptable localization framework capable of addressing the unique challenges posed by complex and dynamic environments. This singular focus on AMCL ensures a cohesive and effective approach to enhancing mobile robot navigation.

Table 1: Summary of Localization (AMCL) Methods from Previous Studies

Authors (Year**)**	**Methodology**	**Key Contributions**	**Experiments**
(Baek et al., 2022)	3D Global Localization with LiDAR.	Precise 3D position recognition in underground mines	Underground mine environments
(Li et al., 2023)	Ultra-Wide-Band	Efficient global pose recovery	Various robot scenarios

	(UWB) Technology.	for mobile robots	
(Raikwar et al., 2023)	2D LIDAR SLAM-Based Localization.	High positioning accuracy for agricultural robotics.	Static agricultural settings
(Tripicchio et al., 2022)	3D Multilateration of UHF RFID Tags	Centimeter-level accuracy for warehouse logistics	Warehouse environments
(Wang et al., 2018)	UWB Array for AMCL Localization	Improved performance of Adaptive Monte Carlo Localization	Various robot scenarios
(Singh et al., 2023)	ROS-Based Localization and Navigation	Highly accurate pose identification and mapping in mobile robots	Simulated and real-world environments

III. SLAM (Simultaneous Localization and Mapping)

In the world of robotics and autonomous navigation, Simultaneous Localization and Mapping (SLAM) has emerged as a fundamental tool, enabling robots to navigate autonomously within intricate surroundings. This field has seen substantial research and development efforts, leading to a wide array of approaches geared towards tackling the intricate challenges linked to mapping and localization. One of the most prominent methods in the field is Grid-Based SLAM, which offers a structured framework for mapping and localization by discretizing a robot's environment into grid cells. Each cell stores information about the corresponding space's occupancy, effectively creating a map. A pivotal breakthrough was introduced by (Wang et al., 2018) they merged Grid-Based SLAM with Ultra-Wide-Band (UWB) technology. This integration revolutionized the accuracy of pose calculation, effectively mitigating initial localization challenges related to speed and stability. Grid-Based SLAM, complemented by UWB, has emerged as a robust solution, addressing initial localization issues. In another research, (Baek et al., 2022) applied this approach to the domain of underground mining, achieving precise 3D position recognition through local point cloud dataset matching, opening new horizons for its application in niche environments. Innovations in Grid-Based SLAM approaches extend beyond algorithmic developments. (Hampton et al., 2017) introduced the RFS-SLAM robot, an open-source and cost-effective platform designed to excel in SLAM tasks. The platform incorporates a novel occupancy-grid SLAM algorithm grounded in random-finite-sets (RFS). Equipped with a LIDAR-Lite 2 laser range finder, this platform demonstrates remarkable success in correcting odometry errors and generating accurate maps in indoor environments. Its versatility paves the way for diverse applications.

(Azak & Erdogan 2018) ventured into the integration of Grid-Based FastSLAM within the V-REP robot simulation program. Their experimentation, employing a Pioneer 3 DX mobile robot equipped with sensors, concentrated on parameter variations. This study unveiled the nuanced trade-offs between accuracy and computational efficiency, enriching our comprehension of optimal parameter tuning for enhanced SLAM performance.

(Pedrosa et al., 2020) brought efficiency to the forefront with an implementation of FastSLAM-based Rao-Blackwellized Particle Filter (RBPF). Their solution, complemented by multithreading and scan matching, achieved the delicate balance between speed and accuracy. Developed within the Robot Operating System (ROS), this software contributes to the expanding repository of efficient SLAM algorithms for real-time applications. Grid-Based SLAM's journey also includes a notable contribution from (Roh et al., 2011), who introduced a novel Fast SLAM method. This approach utilized polar scan matching and a particle weight-based occupancy grid map, catering to the demands of mobile robot navigation in large-scale indoor environments.

(El et al., 2010) addressed the computational efficiency of scan matching within SLAM algorithms. They harnessed streaming SIMD extensions (SSE) instructions to streamline scan matching operations, significantly reducing computational time while maintaining practicality without specialized hardware. (Wongsuwan et al., 2017) introduced the Corrective Gradient Refinement (CGR) algorithm as an enhancement to the Rao-Backwellized Particle Filter (RBPF) framework. This innovation refines mapping solutions and curtails memory consumption, contributing to the progression of pose-graph construction in SLAM.

Beyond Grid-Based SLAM, the landscape of SLAM applications is vast and continually expanding. SLAM's reach extends across diverse sectors, with each field utilizing unique SLAM types and methods tailored to address specific challenges. In agriculture, Kemper et al. (2022) harnessed the LeGO-LOAM algorithm to enable agricultural robot navigation through blueberry bush rows. This breakthrough highlights the potential of simulation-based navigation systems in optimizing agricultural operations, paving the way for precision agriculture.

In the healthcare sector, (Kadam et al. 2023) demonstrated the capabilities of ROS-based assistive robots utilizing the Google Cartographer SLAM algorithm for real-time mapping. This application enhances healthcare delivery by facilitating the safe and efficient movement of robots in healthcare settings, supporting healthcare workers and improving patient care.

Researchers are actively developing novel SLAM methods to enhance accuracy and efficiency. (Cui et al., 2021) focused on enhancing the positioning accuracy of pure visual SLAM in fast motion and complex indoor environments. Their visual-inertial information fusion SLAM method, based on Runge-Kutta improved pre-integration, significantly improved accuracy by avoiding errors caused by first-order approximation.

(Kadam et al., 2023) demonstrated the simulation of a multi-purpose medical assistive robot using the Robot Operating System (ROS) and the Google Cartographer SLAM algorithm. This successful implementation highlights the potential of ROS-based assistive robots in healthcare, benefiting health workers, patients, and healthcare organizations.

(Mantha et al.,2022) investigated fiducial marker network characteristics for autonomous mobile indoor robot navigation. Their findings provide valuable insights for achieving successful navigation in a cost-effective and computationally efficient manner, contributing to improved navigation systems.

In the construction industry, (Xiang et al., 2022) addressed the challenge of SLAM for construction machinery in dynamic construction sites. Their affordable SLAM method, utilizing a multi-layer grid map, optimized machine operations by providing accurate environmental information. This research contributes to the advancement of SLAM techniques in the construction industry, enhancing safety and efficiency.

This comprehensive exploration underscores the ever-expanding horizons of SLAM technologies, from foundational Grid-Based SLAM to innovative applications in diverse fields. The plan is to leverage the Grid-Based FastSLAM method to further advance the capabilities of SLAM in autonomous robotics, building upon the rich foundation and recent innovations in the field. This approach is expected to enhance mapping and localization accuracy while ensuring efficiency in real-world applications.

The landscape of SLAM technologies continues to evolve, driven by the need for more accurate and robust autonomous navigation systems. As researchers and engineers push the boundaries of what is possible, the future holds the promise of even more groundbreaking developments in SLAM, enabling robots to operate effectively and autonomously in increasingly complex environments.

Table 2: Summary of Review on Various SLAM methods

Authors (Year**)**	**Methodology**	**Key Contributions**	**Experiments**
(Wang et al., 2018)	Grid-Based SLAM with UWB technology	Improved pose calculation accuracy with UWB integration	Simulation, Real-world.
(Baek et al., 2022)	Grid-Based SLAM in underground mining	Precise 3D position recognition through local point cloud matching	Underground mining environment
(Hampton et al., 2017)	RFS-SLAM robot with LIDAR-Lite 2	Open-source platform correcting odometry errors and generating accurate maps	Indoor environments
Azak & Erdogan (2018)	Grid-Based FastSLAM within V-REP simulation	Parameter variations study for enhanced SLAM performance	Simulation (V-REP)
(Pedrosa et al., 2020)	FastSLAM-based RBPF with multithreading and scan matching	Achieved speed and accuracy balance with RBPF and multithreading	Real-time experiments within ROS
(Roh et al., 2011)	Fast SLAM method with polar scan matching	Enhanced navigation in large-scale indoor environments	Real-world experiments
(El et al., 2010)	Streamlined scan matching using SSE instructions	Improved computational efficiency without specialized hardware	Simulation
(Wongsuwan et al., 2017)	Corrective Gradient Refinement (CGR) algorithm	Enhanced mapping solutions with reduced memory consumption	Real-time experiments within ROS

Obstacle Avoidance

Obstacle avoidance is a critical aspect of mobile robot navigation in complex environments, and researchers have been actively developing innovative methods to enhance this capability. In one such study (Song, 2022) made notable contributions to this field. he developed a strategy utilizing multi-sensor technology and fuzzy control algorithms to enhance obstacle avoidance accuracy and optimize path planning in complex environments. By integrating data from multiple sensors and employing fuzzy control algorithms, this approach promises improved obstacle avoidance, laying a foundation for future research in this area. (Li et al., 2020) proposed a novel obstacle avoidance method based on machine vision for dynamic obstacle detection. By combining the YOLO-v3 object detection algorithm with the dynamic window approach (DWA), they achieved real-time detection and avoidance of dynamic obstacles. This approach exhibits potential in enhancing local obstacle avoidance performance, benefiting the operability of mobile robots in dynamic environments. (Li et al., 2023) explored an intriguing concept of intelligent physical attack strategies for mobile robots. They demonstrated the potential of these strategies in trapping robots without prior knowledge of system dynamics. Efficient attack algorithms were introduced and validated through simulations and real-life experiments, shedding light on the exploration of physical threats and defense mechanisms in robotics.

(Huber et al., 2023) proposed a control scheme that combined high-level input commands with fast reactive obstacle avoidance (FOA). By utilizing sparse and asynchronous perception, this approach processed sensor data efficiently and enabled real-time collision avoidance. The controller demonstrated its efficiency in obstacle evaluation through experiments in cluttered indoor and dynamic outdoor environments.

In another research, (Kim et al., 2017) introduced the Dual Expanded Guide Circle (Dual-EGC) algorithm, enhancing obstacle avoidance in remotely operated mobile robots. Their algorithm outperformed the original EGC method, exhibiting higher obstacle avoidance efficiency and faster execution speeds. This improvement has significant implications for the control and operation of remotely operated mobile robots.

(Zhang et al., 2019) addressed obstacle avoidance for two-wheeled mobile robots using the Dynamic Window Approach (DWA) algorithm. Their study demonstrated the algorithm's effectiveness in facilitating obstacle avoidance in controlled environments. It provided a foundation for further exploration of advanced algorithms in robotics.

(Mata-Machuca et al., 2021) conducted experimental verification of a leader-follower formation control system for two-wheeled mobile robots with obstacle avoidance. Their approach showcased successful performance in real-world scenarios, highlighting its potential for coordinated leader-follower formations and obstacle avoidance.

(Cheong et al., 2020) developed an intelligent garbage bin robot capable of autonomous garbage retrieval. Utilizing the Robot Operating System (ROS), their robot effectively managed waste collection, navigated autonomously, and avoided collisions, offering a promising solution to improve hygiene in open public spaces.

(Tüfekçi et al., 2023) performed an experimental comparison of global planners for mobile robots navigating unknown environments with dynamic obstacles. Their study assessed Global Planner, NavfnROS planners, and Carrot Planner within the ROS package in two distinct environments. NavfnROS planners exhibited the fastest results, emphasizing the need for further testing in complex scenarios.

These studies collectively advance the field of mobile robotics by improving obstacle avoidance techniques. By integrating various methodologies, from fuzzy control algorithms and machine vision to intelligent attack strategies and fast reactive obstacle avoidance, researchers enhance the accuracy, efficiency, and adaptability of obstacle avoidance systems contributions with substantial implications for robotics, automation, and autonomous navigation.

These developments have significant implications for various domains, including robotics, automation, and autonomous navigation.

Table 3: Compilation of Prior Research Findings on Obstacle Avoidance

Authors	Methodology	Key Contributions	Experiments
(Song, 2022).	Multi-sensor technology and fuzzy control algorithms	Improved obstacle avoidance accuracy and optimized path planning in complex environments.	Complex operating environments
(Li et al., 2020)	Machine vision-based obstacle detection with YOLO-v3 and DWA	Real-time detection and avoidance of dynamic obstacles, enhancing local obstacle avoidance performance.	Real-time dynamic environments
(Li et al., 2023)	Intelligent physical attack strategy for mobile robots	Efficient attack algorithms, trapping robots without prior knowledge of system dynamics.	Simulations and real-life experiments
(Huber et al., 2023)	High-level input command combined with fast reactive obstacle avoidance (FOA)	Efficient real-time collision avoidance through sparse and asynchronous perception.	Cluttered indoor and dynamic outdoor environments
Kim & Kim. (2017)	Dual Expanded Guide Circle (Dual-EGC) algorithm	Enhanced obstacle avoidance efficiency, reduced travel distances, and faster execution speeds.	Simulations
(Zhang et al., 2019).	Dynamic Window Approach (DWA) algorithm	Effective obstacle avoidance for two-wheeled mobile robots.	Controlled environments

(Mata-Machuca et al., 2021)	Leader-follower formation control system with obstacle avoidance	Successful performance in real-world scenarios, enabling coordinated leader-follower formations and obstacle avoidance.	Real-world scenarios
(Cheong et al., 2020)	Robot Operating System (ROS) for autonomous garbage retrieval	Efficient waste collection, autonomous navigation, and collision avoidance for improved hygiene in open public spaces.	Autonomous operation in open public spaces
(Tüfekçi et al., 2023)	Experimental comparison of global planners	NavfnROS planners exhibited the fastest results in navigating unknown environments with dynamic obstacles.	Unknown environments with dynamic obstacles

Robot Operating System (Ros)

In recent years, ROS has emerged as a powerful platform for developing robotic systems, enabling researchers to address critical challenges in navigation, perception, control, and human-robot interaction. The reviewed studies demonstrate the versatility and effectiveness of ROS across various applications, showcasing its potential for creating intelligent and efficient robotic systems.

In the study by (Megalingam et al. 2023), an autonomous navigation platform with human-robot interaction was proposed and implemented for indoor service robots. By utilizing ROS, the researchers developed algorithms for autonomous navigation, speech processing and recognition, and object detection and recognition. The system was evaluated through a confusion matrix generated from 125 different cases, demonstrating a decent level of correctness with an accuracy rate of 0.925. This research contributes to the development of efficient service robots that assist individuals with physical challenges and visual impairments in their daily activities.

(Nwankwo et al., 2023) addressed the limitations of commercially available intelligent transport robots by developing an open-source mobile robot called ROMR. Utilizing ROS, ROMR utilized off-the-shelf components, additive manufacturing technologies, and a consumer hoverboard. The study validated the robustness and performance of ROMR through real-world and simulation experiments, highlighting its advantages over commercial platforms, including affordability, customization options, and compatibility with ROS. ROMR offers a cost-effective and customizable solution for research and industrial applications.

(Prasad et al., 2023) focused on designing and developing the software stack of an autonomous vehicle using ROS. Their work encompassed various aspects such as SLAM-based path tracking, computer vision-based controller, and intelligent object avoidance. Promising results were obtained in a virtual environment, and future work will focus on improving module robustness and incorporating advanced techniques. This research contributes to the development of efficient autonomous vehicles with advanced capabilities.

(Zhang et al., 2023) tackled the implementation and optimization of the ORB-SLAM2 algorithm on mobile robots within the ROS operating system. They addressed the challenges of running the algorithm directly on ROS and the limitations of using a PC, resulting in inefficiency and reduced flexibility. By building a ROS operating system on an embedded system and optimizing the algorithm's CPU usage, the researchers achieved efficient algorithm operation and reduced cost and hardware configuration requirements. The study validated their approach by demonstrating the generation of grid maps for robot autonomous navigation, highlighting the advantages of the ROS operating system.

(Duraisamy et al., 2023) explored multi-sensor fusion for off-road drivable region detection. By combining deep learning-based semantic segmentation with LiDAR-based ground segmentation, the researchers achieved improved accuracy in detecting drivable regions. The study emphasized the potential for further enhancements, such as incorporating interpolated point cloud data and applying the solution to path planning for mobile robots. This research contributes to the development of robust off-road navigation systems.

(Cihlar et al., 2023) focused on the simulation of autonomous robotic systems for intelligence and reconnaissance operations. By utilizing ROS and the Gazebo simulator, the researchers accelerated the development and evaluation of algorithms, enabling realistic simulations without risking damage to hardware. Their study showcased the flexibility of the solution, demonstrating a scenario involving flying and terrestrial robots in a Chemical, Biological, Radiological, Nuclear, and Explosive (CBRNE) mission. ROS proved to be a suitable framework for intelligence and reconnaissance simulations.

Collectively, these studies demonstrate the diverse applications and capabilities of ROS, highlighting its significance in advancing the field of robotics and enabling the development of intelligent and efficient robotic systems.

Current Study:

After reviewing recent advancements in the field and drawing insights from other researchers' work, we will implement these techniques using the Robot Operating System (ROS). Our study will focus on integrating Grid-Based SLAM, AMCL for localization, and a sophisticated obstacle avoidance system to enhance the robot's autonomous navigation capabilities.

Model Architecture

This methodology outlines the step-by-step process for developing an advanced autonomous navigation system for differential drive mobile robots. By leveraging the power of ROS2, along with technologies such as URDF, SLAM, AMCL, Gazebo, and RViz2, we aim to create a robust and reliable solution. This methodology focuses on the integration of these technologies, which collectively enable the accurate simulation, mapping, localization, and navigation of the mobile robot.

Setting Up Software Tools and ROS2 for Robotic Development:

To establish the software environment, Ubuntu 22.04 and ROS2 Humble were installed. A ROS2 package named "mythesis_bot" was created to serve as a workspace for system implementation and code organization.

Key Concepts in ROS2 and Coordinate Systems:

ROS2 Core and Nodes*:* The foundation of ROS2 lies in its core, which manages communication among nodes. Nodes, modular software units, perform tasks like control and planning. They collaborate through asynchronous data exchange using "topics."

ROS2 Topics, Subscribers, and Publishers: Topics enable nodes to share data asynchronously. Subscribers listen to topics to receive messages, while publishers generate messages for specific topics. This mechanism allows nodes to collaborate without direct connections.

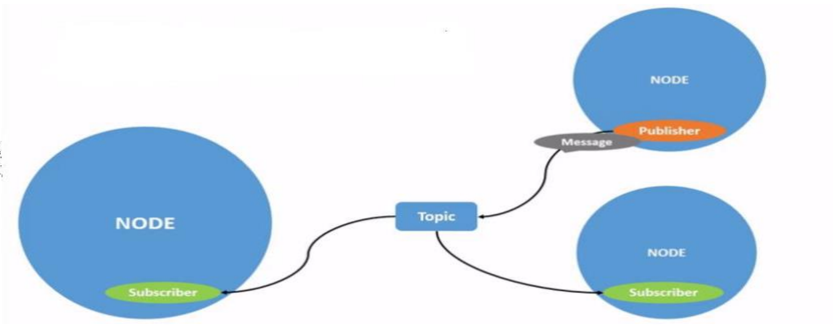


Figure 2: ROS Topic Communication between Nodes and Subscribers

Joint State Publisher: This crucial ROS2 package manages a robot's joint state, providing data about joint locations, velocities, and efforts. It ensures accurate and real-time information sharing for visualization, control, and more.

Robot State Publisher*:* This component broadcasts a robot's joint and linkage state. Obtaining this data from the robot's URDF file, it promotes uniform coordination and communication among various nodes.

ROS Coordinate Conventions*:* ROS2 employs standardized coordinate frames, such as 'base_link,' following conventions specified in ROS Enhancement Proposals (REPs) like REP 105. These conventions provide consistent orientation and alignment.

Coordinate Systems and Frames*:* Robots use coordinate frames to represent positions and orientations. These frames create a hierarchical structure, with the world frame as the global reference. Frames like 'odom' and 'base_link' help define robot stance and motion.

Hierarchy of Frames: The hierarchy starts with the world frame, followed by frames like 'odom' and 'base_link.' Each frame inherits the transformation of its parent frame. Robot components are represented using link frames.

URDF Robot Frames**:** The URDF defines frames like 'world,' 'odom,' and 'base_link.' Each frame represents a specific part of the robot. Link frames, connected by joints, form the robot's kinematic chain.

By adhering to ROS2's standardized conventions and utilizing its coordination mechanisms, a cohesive software environment is created, enabling accurate robot representation, seamless communication, and effective navigation. This is vital for implementing tasks like Simultaneous Localization and Mapping (SLAM) in the Gazebo simulator.

Rqt graph:

Rqt_graph is a crucial tool within the ROS (Robot Operating System) network, providing essential insights into the connections and dependencies among ROS nodes and topics. This visualization tool is invaluable for understanding the intricacies of ROS-based software architectures, facilitating efficient debugging, optimization, and informed decision-making in robotics and automation applications.

Differential Drive Robot Design:

Differential Drive Robots:

Differential-drive control is a widely used control mechanism for robots, including the one being designed in this project. A differential-drive robot consists of two driven wheels, typically placed on the left and right sides of the robot, which provide the primary means of motion. Other wheels, such as caster wheels, are included primarily for stability and can rotate freely in any direction.

The advantage of using a differential-drive configuration is its simplicity in terms of understanding, construction, and control. This setup allows the robot to be highly maneuverable, capable of rotating in place, and navigating tight spaces with ease. Unlike vehicles that require a turning radius or multiple-point turns, differential-drive robots can perform agile maneuvers.

In the context of this research project, the differential-drive robot being developed is envisioned as a small box-shaped robot. The driven wheels are positioned near the rear of the robot, while a caster wheel is located at the front.

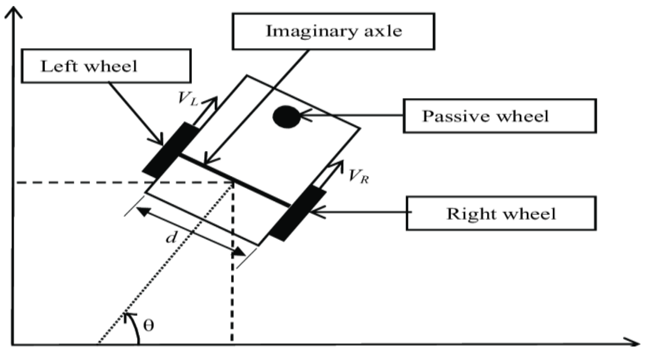


Figure 3: Differential drive Model

$$(x\ \dot{y}\ \theta\ ) = [cos\theta\ 0\ sin\theta\ 0\ 0\ 1\ ][v\ \omega\ ] \text{ .... equation 1}$$

$$V\ = (\frac{v_R +\ V_L}{2}) \text{................................equation2}$$

$$\omega\ = (\frac{v_R -\ V_L}{d}) \text{.............................. equation3}$$

As shown in the diagram above, the differential drive robot consists of two wheels, independently driven, enabling the robot to move in different directions. The robot's motion is governed by the general kinematic equation, as shown in Equation 1. This equation relates the translational velocity (V) and the angular velocity (ω) of the robot, as expressed in Equations 2 and 3, respectively. The equation assumes the velocity of the rear axle midpoint as the robot's velocity, considering an imaginary axle connecting the rear wheels.

Creating the URDF Robot

The URDF (Unified Robot Description Format) is utilized to define the physical components and their relationships within a robot. To enhance organization and maintainability, it is common to divide the URDF configuration into multiple files and include them in a main file. This modular approach simplifies maintenance tasks and promotes collaboration among team members.

In the development of the URDF differential drive robot, the configuration was structured using a modular file system. Separate files were created for different components, such as the core structure and sensors, to improve organization and facilitate better management. This modular approach allows for easy customization and maintenance of individual components without impacting the entire robot description.

By adopting this modular file structure, the process of creating the URDF robot becomes more organized, manageable, and conducive to collaborative efforts. It allows for efficient development and customization of the robot's description and launch configuration, promoting flexibility and ease of maintenance in the long run.

Creating the visual structure

To establish the visual structure of the robot, a core file named "robot_core. xacro" was defined. The Xacro tool is used to define the visual structure of the robot. Through this tool, links and joints representing the physical components and their connections are created. The process starts with the definition of the base link, referred to as 'base_link,' and progresses to include other components such as the chassis, wheels, and caster wheel. Each link and joint are defined using XML tags, incorporating appropriate origins, geometries (such as boxes and cylinders), and materials (including colors) to ensure an accurate visual representation.

Table 4: URDF design specifications

Base Link	base_link
Chassis Link	chassis
Dimensions	0.3m x 0.3m x 0.15m
Material	white
Inertial Properties	mass = 0.5kg
Left Wheel	
Joint	left_wheel_joint
Parent Link	base_link
Child Link	left_wheel
Dimensions	Cylinder (radius 0.05m, length 0.04m)
Material	blue
Inertial Properties	mass = 0.1kg
Right Wheel	
Joint	right_wheel_joint
Parent Link	base_link
Child Link	right_wheel
Dimensions	Cylinder (radius 0.05m, length 0.04m)
Material	blue
Inertial Properties	mass = 0.1kg
Caster Wheel	
Joint	caster_wheel_joint
Parent Link	chassis
Child Link	caster_wheel
Dimensions	Sphere (radius 0.05m)
Material	black
Inertial Properties	mass = 0.1kg

Sensor Integration:

In the development of an autonomous navigation system for differential drive mobile robots, sensor integration plays a crucial role in providing the necessary perception and localization capabilities. The integration of sensors such as LiDAR and cameras enables the robot to perceive its environment, make informed decisions, and navigate autonomously.

Adding Lidar

Lidar, or light detection and ranging, is a type of sensing that uses lasers to map the area around it in detail and measure distances. When a laser pulse strikes an item, a timer measures how long it takes for the light to bounce back. Lidar systems can precisely calculate the distance, shape, and location of objects in the environment by examining the return time and intensity of the reflected light.

The usage of lidar is important in the context of this research. To execute SLAM (Simultaneous Localization and Mapping) and AMCL (Adaptive Monte Carlo Localization), the robot needs to be able to receive precise and thorough information about its surroundings. SLAM refers to the process of creating a map of an unknown environment while simultaneously localizing the robot within that map. Using sensor data from the robot's sensors, such as lidar scans, the AMCL algorithm can estimate the robot's position and orientation in real-time.

A local map of the robot's surroundings can be made by integrating lidar into the robot's system, which enables it to sense objects' distances correctly. This information is essential for the robot to navigate autonomously, avoid obstacles, and effectively plan its path. Additionally, the lidar data is utilized in the SLAM algorithm to build a map of the environment and refine the robot's localization estimates.

Table 5: Lidar Design Specification

LIDAR Mounting	"Laser_frame" with a fixed joint
Number of Samples	360 samples per scan
Angular Resolution	1 degree per sample
Scanning Range Minimum Angle	-3.14 radians
Scanning Range Maximum Angle	3.14 radians
Range Measurement	0.12 meters
Maximum Range	6 meters
Update Rate	5 Hz
Published Message Type	sensor_msgs/LaserScan

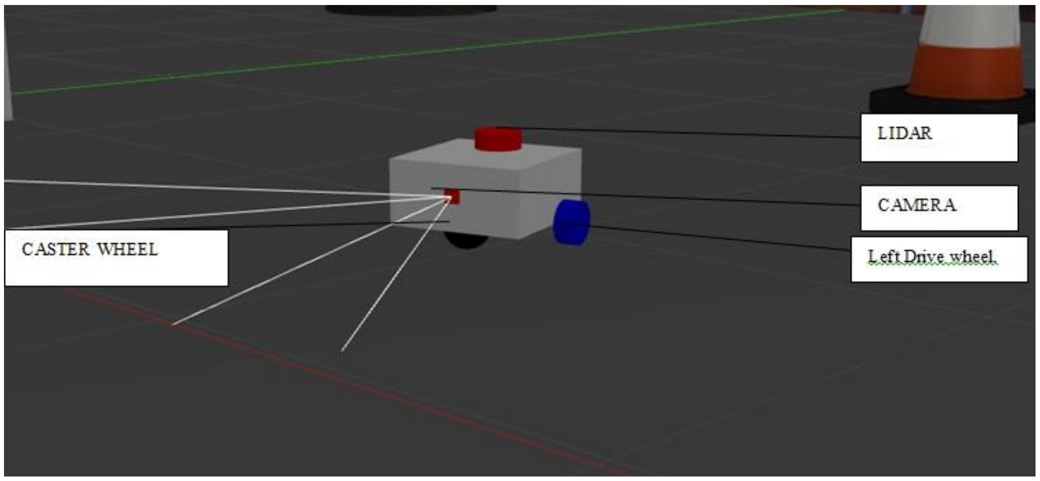


Figure 4: Simulated mobile Robot in Gazebo

Adding Camera

The integration of cameras in robots holds immense importance for diverse robotic applications, providing human-like visual perception. Cameras enable robots to capture and process visual information about their surroundings, facilitating informed decision-making. This visual input empowers robots to navigate, detect obstacles, recognize objects and faces, and perform tasks requiring visual understanding. The overview encompasses camera types, image handling, compression, focal length, coordinate systems, integration in ROS, simulation in Gazebo, and sensor configuration, all contributing to the crucial role of cameras in enhancing robotic capabilities.

Table 6: Camera Design Specification

Camera Joint Type	Fixed
Parent Link	chassis
Child Link	camera_link
Origin	Translation (x=0.305, y=0, z=0.08), Rotation (roll=0, pitch=0, yaw=0)
Geometry	Box (size: width=0.010, height=0.03, depth=0.03)
Material	Red
Camera Optical Joint Type	Fixed
Parent Link	camera_link
Child Link	camera_link_optical
Origin	Translation (x=0, y=0, z=0), Rotation (roll=-pi/2, pitch=0, yaw=-pi/2)
Resolution	640x480 pixels

Update Rate	30 Hz
Horizontal Field of View	Approximately 1.047198 radians (~60 degrees)
Minimum Depth	0.05 meters
Maximum Depth	3meters

IV. Gazebo Simulation and Ros Visualization (Rviz)

Gazebo Simulation:

Gazebo, the open-source robotics simulation tool, stands as an essential innovation hub for the field of robotics. This versatile platform enables the replication of real-world scenarios within a secure virtual environment. During the design of the simulated environment for our robot in Gazebo, it was thoughtfully crafted with walls and obstacles, creating challenging scenarios that refine navigation algorithms and enhance robotic intelligence. This simulated environment, named "obstaclesworld," serves as a dynamic testing ground. Moreover, Gazebo serves as a cost-effective alternative to physical testing, reducing expenses and risks. In essence, it's a virtual haven where robotics pioneers' experiment, iterate, and perfect autonomous navigation systems, shaping the future of robotics through immersive simulated challenges. Figure 5 shows our simulated environment in the gazebo.

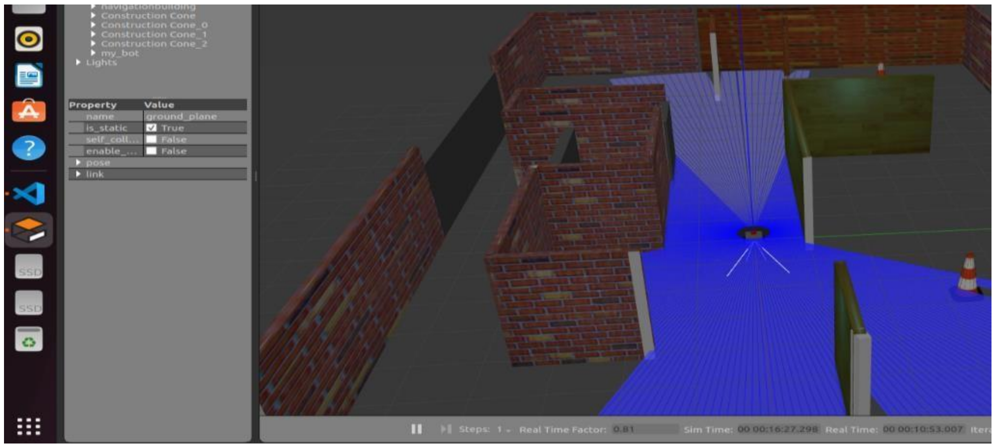


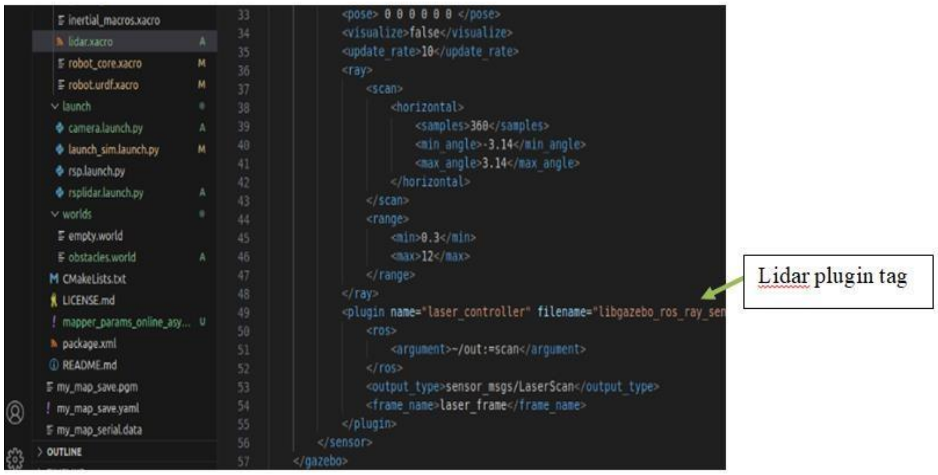
Figure 5: Simulated Environment in Gazebo

Gazebo Tag Utilization and Sensor Integration:

The URDF file is enriched with <gazebo> tags to specify Gazebo-specific configurations, enhancing the robot's visual representation in the simulation environment. The inclusion of <gazebo> tags enhance the simulation environment by providing accurate sensor data and improved visual representation, facilitating comprehensive robot testing.

Lidar Simulation: To simulate the lidar and integrate it with ROS:

The URDF gains a "laser_frame" link and joint to establish a reference point. Inside the <link> tag, a <gazebo> tag is added to simulate the lidar sensor. Lidar-specific parameters, like scanning range and resolution, are defined within the <ray> tag. The ROS plugin "libgazebo_ros_ray_sensor.so" is added using the <plugin> tag, enabling communication and LaserScan data publishing.

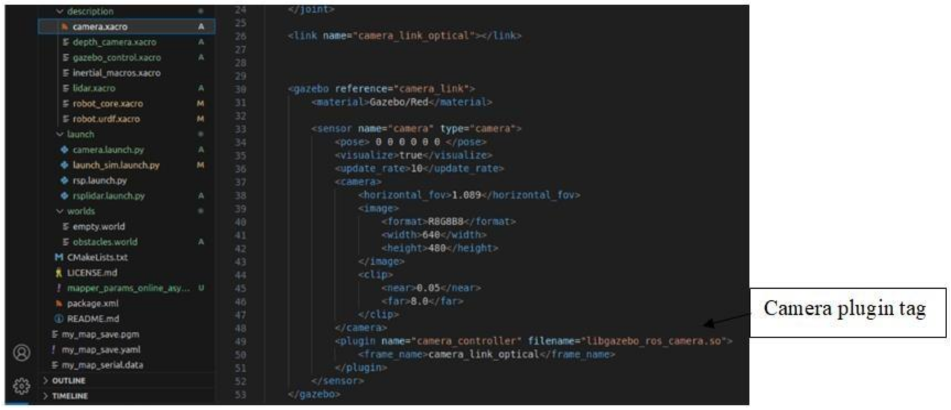


```
            <pose> 0 0 0 0 0 0 </pose>
            <visualize>false</visualize>
            <update_rate>10</update_rate>
            <ray>
                <scan>
                    <horizontal>
                        <samples>360</samples>
                        <min_angle>-3.14</min_angle>
                        <max_angle>3.14</max_angle>
                    </horizontal>
                </scan>
                <range>
                    <min>0.3</min>
                    <max>12</max>
                </range>
            </ray>
            <plugin name="laser_controller" filename="libgazebo_ros_ray_sen
                <ros>
                    <argument>~/out:=scan</argument>
                </ros>
                <output_type>sensor_msgs/LaserScan</output_type>
                <frame_name>laser_frame</frame_name>
            </plugin>
        </sensor>
    </gazebo>
```

Figure 6: lidar plugin integration in Gazebo

Camera Integration: Camera simulation is achieved as follows:

A <sensor> tag is included within the <gazebo> tag to configure camera settings. The ROS plugin "libgazebo_ros_camera.so" is incorporated within the <sensor> tag for seamless Gazebo-ROS data exchange.



```
</joint>

<link name="camera_link_optical"></link>



<gazebo reference="camera_link">
    <material>Gazebo/Red</material>

    <sensor name="camera" type="camera">
        <pose> 0 0 0 0 0 0 </pose>
        <visualize>true</visualize>
        <update_rate>10</update_rate>
        <camera>
            <horizontal_fov>1.089</horizontal_fov>
            <image>
                <format>R8G8B8</format>
                <width>640</width>
                <height>480</height>
            </image>
            <clip>
                <near>0.05</near>
                <far>8.0</far>
            </clip>
        </camera>
        <plugin name="camera_controller" filename="libgazebo_ros_camera.so">
            <frame_name>camera_link_optical</frame_name>
        </plugin>
    </sensor>
</gazebo>
```

Figure 7: Camera plugin integration in Gazebo

Integration of Control Functionality for The Mobile Robot in Gazebo.

To enable control of the robot's motion within Gazebo, the libgazebo_ros_diff_drive.so plugin was employed. This plugin facilitated the translation of commands received from the /cmd_vel topic into simulated movements within the Gazebo environment. In addition, it publishes the odometry information and transformations between frames, contributing to accurate position estimation.

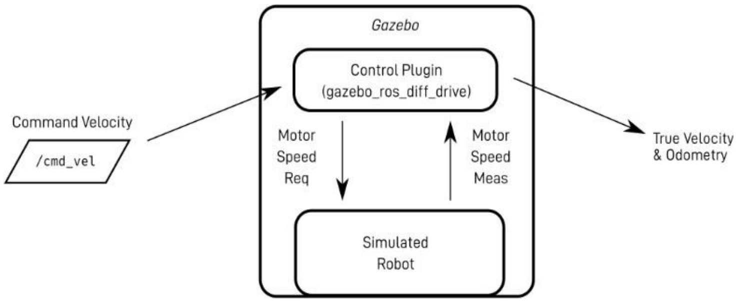


Figure 8: URDF Integration for Precise Robot Control in Gazebo Simulation

Integration of Control Functionality into URDF

To incorporate control functionality into the URDF model, a separate xacro file named "gazebo_control.xacro" was created. This file was subsequently included in the main URDF file, "robot.urdf.xacro."Within the "gazebo_control.xacro" file, a <plugin> tag was defined, specifying the libgazebo_ros_diff_drive.so plugin with configurable parameters. These parameters encompassed essential wheel information, limits, and output configurations, enabling precise control of the robot's behavior within the Gazebo simulation environment.

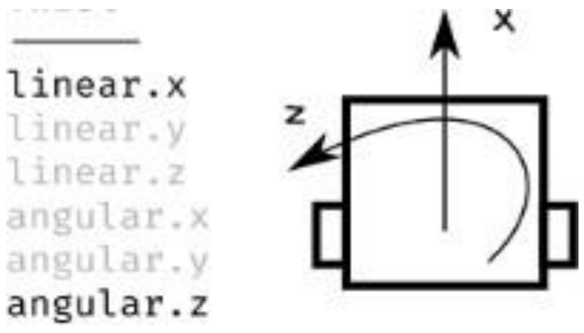


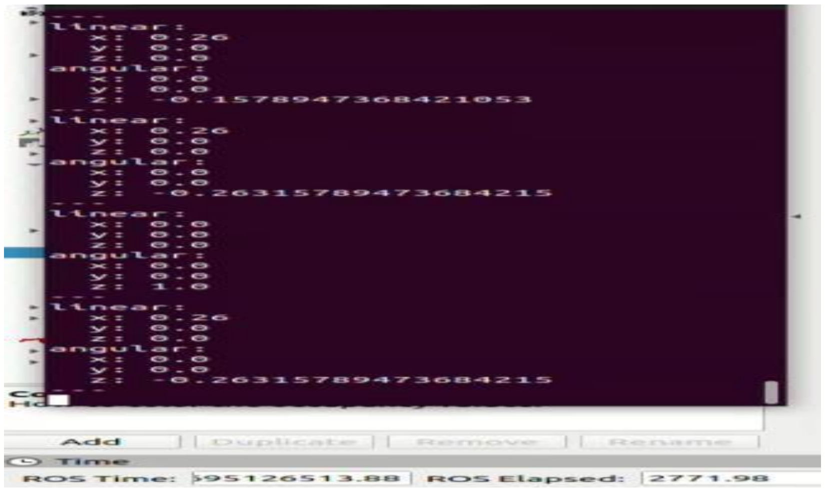
Figure 9: Coordinate System and Motion Control

As seen in figure 9 above we have only two coordinates in use which are the linear X and angular Z Within the autonomous navigation system, the robot's motion control is primarily governed by two coordinates: linear X and angular Z. The linear X coordinate represents the forward or backward movement of the robot along its longitudinal axis. Moving forward is indicated by positive values, while moving backward is shown by negative numbers. On the other hand, the angular Z coordinate denotes the robot's revolution about its vertical axis. Positive numbers represent rotation going counterclockwise, whereas negative values represent rotation going clockwise. By manipulating these two coordinates, the Gazebo control plugin translates the velocity commands received on the /cmd_vel topic into appropriate simulated movements within the Gazebo environment. This allows the robot to navigate the virtual world by controlling its linear motion and rotation.

Command Velocities (CMD Velocities):

CMD velocities play a pivotal role in guiding a robot's motion and behavior. CMD velocities, often represented as linear and angular velocity commands, are the instantaneous directives sent to a robot's actuators, instructing them on how to move and orient in space. These commands dictate the robot's intended motion, facilitating navigation, control, and teleoperation. In the context of this dissertation, the analysis of CMD velocities is paramount, as it allows for a comprehensive assessment of the robot's motion control capabilities. By analyzing the robot's response to these velocity commands during navigation, we can evaluate its ability to follow predefined paths, respond to dynamic obstacles, and execute tasks effectively.

Furthermore, studying CMD velocities provides insights into the robot's autonomy, stability, and adaptability, which are crucial factors in real-world applications. The subsequent sections of this dissertation will delve into the practical implementation and analysis of CMD velocities within the context of our robot design, utilizing tools such as URDF, Gazebo, and RViz in ROS2 to conduct a thorough evaluation of the robot's performance.



```
---
linear:
  x: 0.26
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: -0.1578947368421053
---
linear:
  x: 0.26
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: -0.2631578947368421 5
---
linear:
  x: 0.0
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 1.0
---
linear:
  x: 0.26
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: -0.2631578947368421 5
---
```

Figure 10: Display of Command Velocities on Terminal During Navigation

Integrated Architecture for Robot Navigation: URDF, State Management, and Gazebo Control

By combining the URDF, Robot State Publisher, and Gazebo plugins, a comprehensive framework for accurate robot representation, state estimation, and simulated motion control is established.

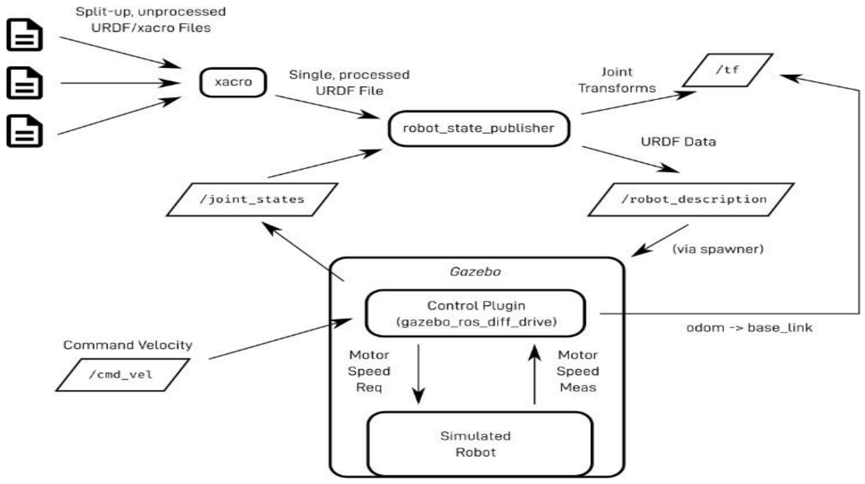


Figure 11: Robot Representation & Simulation Framework: URDF to Gazebo Control

In the figure 11 above, the split and unprocessed URDF/Xacro files serve as the initial representation of the robot's structure and properties. These files were divided into multiple parts to enhance modularity and reusability. Xacro macros and features are employed within the Xacro files to simplify the authoring and maintenance of the robot description.

To process the Xacro files, the Xacro tool is utilized, which expands the macros and resolves includes, conditionals, and loops present in the Xacro files. The outcome of this process is a single processed URDF file that encompasses the complete robot description with all the macros and Xacro features resolved.

The processed URDF file is then connected to the Robot State Publisher. This publisher subscribes to the joint state information and publishes the robot's state on the /tf topic, while also providing the robot description on another output. The Robot State Publisher generates the joint transforms based on the joint state information and publishes them on the /tf topic. The published joint transforms are received and managed by the TF (Transform) system in ROS, enabling other ROS nodes to access and transform data between different coordinate frames in the robot.

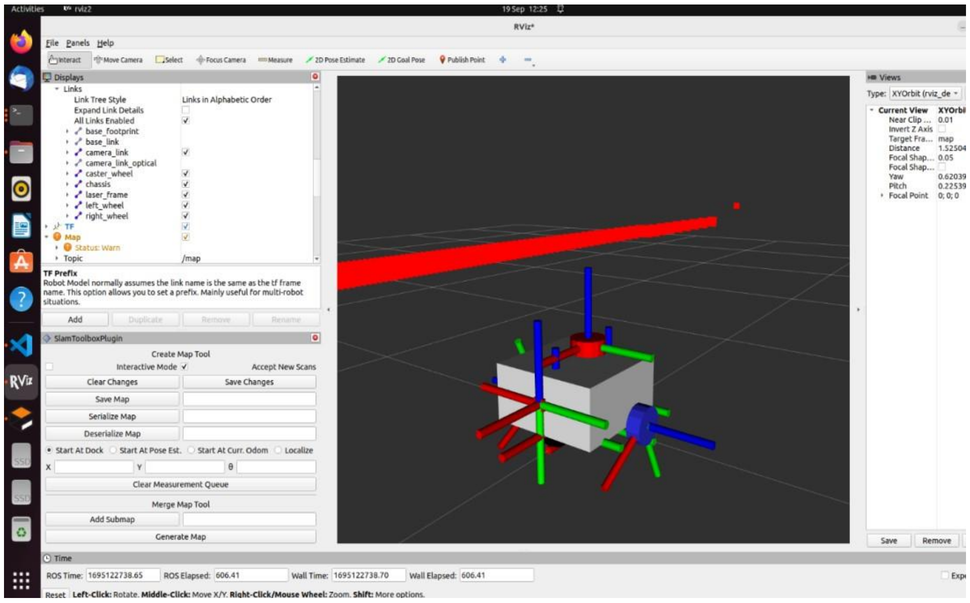
Furthermore, the Robot State Publisher publishes the processed URDF data (robot description) on another output. This data represents a comprehensive description of the robot's structure, kinematics, and properties. Other ROS nodes can access and utilize this robot description for various purposes, such as visualization, simulation, motion planning, or any other relevant tasks.

The processed URDF data is sent to a Gazebo control, which utilizes the libgazebo_ros_diff_drive.so plugin. This plugin enables control of the robot's motion within Gazebo by translating commands received from the /cmd_vel topic into simulated movements within the Gazebo environment. Additionally, the plugin publishes odometry information and transformations between frames, contributing to accurate position estimation. The control diagram shows an arrow connecting the robot description via the spawner to the Gazebo control. This indicates that the control receives command velocity inputs.

Lastly, The Joint States, representing the current joint positions, can be subscribed to by other ROS nodes using the /joint_states topic. There is a feedback connection from the Joint States back to the Robot State Publisher to ensure that the updated joint states are available for generating the joint transforms and updating the robot's state.

Rviz Visualization:

RVIZ, a powerful graphical interface in ROS, offers versatile visualization capabilities with various plugins. Users can customize displays for robot data, sensor readings, and more, including point clouds, images, and 3D models. The TF Display visualizes coordinate frames in a robot system, while Interactive Markers enable real-time interaction. RVIZ also supports path and trajectory visualization, grid and map display for SLAM-generated maps, camera view simulation, and an intuitive user interface for easy configuration and customization of visualizations.



Links
Link Tree Style: Links in Alphabetic Order
Expand Link Details: ☐
All Links Enabled: ✓
base_footprint
base_link
camera_link ✓
camera_link_optical
caster_wheel ✓
chassis ✓
laser_frame ✓
left_wheel ✓
right_wheel ✓
TF ✓
Map ✓
Status: Warn
Topic: /map

TF Prefix
Robot Model normally assumes the link name is the same as the tf frame name. This option allows you to set a prefix. Mainly useful for multi-robot situations.

☐ Interactive Mode ✓ Accept New Scans
● Start At Dock ○ Start At Pose Est. ○ Start At Curr. Odom ○ Localize
X Y θ

Type: XYOrbit (rviz_de
Current View: XYOrbi
Near Clip ...: 0.01
Invert Z Axis: ☐
Target Fra...: map
Distance: 1.52504
Focal Shap...: 0.05
Focal Shap...: ☐
Yaw: 0.62039
Pitch: 0.22539
Focal Point: 0; 0; 0

ROS Time: 1695122738.65 ROS Elapsed: 606.41 Wall Time: 1695122738.70 Wall Elapsed: 606.41

Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click/Mouse Wheel: Zoom. Shift: More options.

Figure 12: Visualization of Robot TF in RVIZ

Mapping and Localization

Simultaneous Localization and Mapping (SLAM):

Simultaneous Localization and Mapping (SLAM) is a crucial challenge in robotics, involving the creation of a map of an unknown environment while simultaneously determining the robot's position within it. This capability is vital for autonomous systems as it allows them to navigate and understand their surroundings by combining sensor data and motion information.

SLAM algorithms manage the complexity and uncertainty of real-world environments by integrating data from various sensors like laser scanners, cameras, and inertial sensors. The SLAM problem comprises mapping (creating the environment representation) and localization (determining the robot's position), and the main challenge is to solve both simultaneously to ensure accuracy in navigation and mapping.

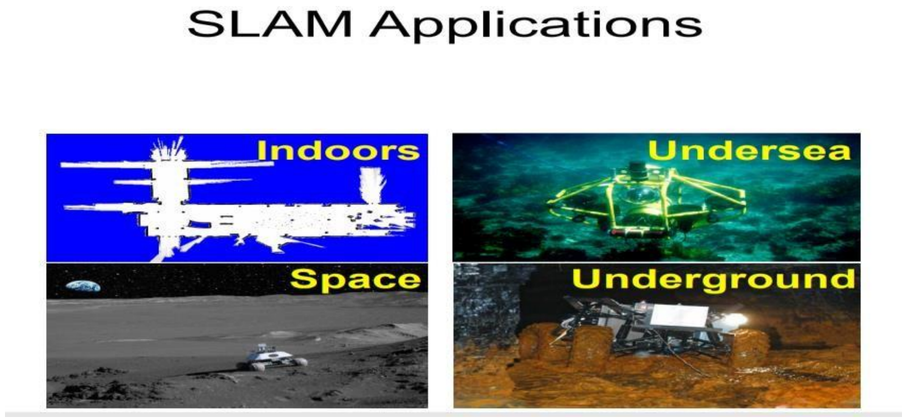


Figure 13: Simultaneous Localization and Mapping (SLAM) Applications.

V. Overview of SLAM Methods:

a) **Feature-based SLAM**: Feature-based SLAM relies on distinct visual features like landmarks or key points to estimate the robot's pose and create a map. However, it faces challenges in feature extraction, matching, and adapting to changing environments, which can be computationally demanding and error prone (Azzam et al., 2020).

b) **Pose Graph SLAM:** Pose Graph SLAM represents the environment as a graph of robot poses and landmark positions. By maintaining a graph structure, pose graph SLAM can handle loop closures and perform global optimization to refine the map and robot's trajectory. However, pose graph SLAM requires extensive data association and computational resources to construct and optimize the graph. The accurate association of data with the graph nodes and the optimization process can be complex and computationally demanding.

c) **Grid-Based SLAM**: Grid-Based SLAM discretizes the environment into a grid map, where each grid cell represents a specific location or occupancy status. Grid-based offers advantages such as robustness, global information, and memory efficiency in SLAM, (Wurm et al., 2010).

Table 7: Comparison of SLAM Methods:

SLAM Method	**Feature-based SLAM**	**Pose Graph SLAM**	**Grid-Based SLAM**
Method	Extracts and tracks features	Represents environment as a graph	Discretizes environment into a grid
Challenges	Feature extraction, matching, robustness	Data association, computational resources	N/A
Robustness	Depends on feature extraction and matching quality	Depends on data association and loop closure detection	Can handle different sensor measurements
Memory efficiency	Requires storage of large feature databases	Requires storage of pose graph and landmark information	Requires less memory as information is stored in a grid structure
Real-time performance	Depends on feature extraction and matching speed.	Depends on computational resources for graph optimization	Can provide real-time mapping and localization
Simplicity	More complex due to feature extraction, matching, and association	More complex due to graph optimization and loop closure detection	Relatively straightforward to implement and understand

Proposed SLAM Method (Grid-Based Fast Slam)

In this research, the Grid-based FastSLAM algorithm was chosen due to its advantages over other SLAM techniques. Grid-based FastSLAM combines the benefits of FastSLAM with a grid-based map representation, making it a suitable choice for simultaneous localization and mapping tasks.

By utilizing a grid map, which divides the environment into cells, Grid-based FastSLAM provides a structured and efficient way to represent the environment. Each cell in the grid corresponds to a small area and maintains its occupancy state. This allows the algorithm to accurately track the occupancy information of the environment and construct a detailed map.

The Grid-based FastSLAM algorithm consists of two main steps: prediction and correction.

The mathematical expressions for the prediction and correction steps in Grid-based FAST SLAM:

Proposed Algorithm, Grid-based FastSLAM.

```
1: Procedure FastSLAM ($X_{t-1}, U_t, Z_t$)
2:   $X_{\underline{t}} = X_t = \emptyset$
3:   for m = 1 to M do
4:       $X_t^{[k]} = MotionUpdate(U_t, X_{t-1}^{[k]})$
5:       $W_t^{[k]} = SensorUpdate(Z_t, X_t^{[k]})$
6:       $M_t^{[k]}$ = UpdateOccupancyGrid ($Z_t, X_t^{[k]}, M_{t-1}^{[k]}$)
7:       $\underline{X}_t = \underline{X}_t + \langle X_t^{[k]}, W_t^{[k]} \rangle$
8:   end for
9:   for K = 1 to M do
10:        draw i with probability $W_t^{[i]}$
11:        add $\langle X_t^{[i]}, M_t^{[i]} \rangle$ $to\ X_t$
12:  end for
13:  return $X_t$
14: end procedure
```

Prediction Step: The prediction step updates the particle poses based on the motion model. Let's denote the control input at time "t" as u_t . The prediction is performed as follows:

$$x_t^k = MotionUpdate(u_{t,}\ x_{t-1}^k) \text{......................... equation 4}$$

Here, Motion Update represents the motion model, which generates a new pose estimate x_t^k based on the previous pose x_{t-1}^k and control input u_t .

Correction Step: The correction step updates the particle maps based on the sensor measurements. The correction is performed as follows:

$$m_t^k = \text{UpdateOccupancyGrid}(x_t^k,\ m_{t-1}^k,\ z_t) \text{...................equation 5}$$

Here, Update Occupancy Grid represents the occupancy grid mapping algorithm, which updates the map m^k_t based on the current pose x_t^k the previous map m_{t-1}^k and the sensor measurements z_t.

During the correction step, the measurement likelihood p(z_t | x_t^k, m_t^k) is also calculated. This represents the probability of obtaining the sensor measurements z_t given the pose x_t^k and map $m^k{}_t$

The prediction and correction steps are performed iteratively for each particle in the FastSLAM algorithm. After updating the poses and maps, the importance weight w_t^k of each particle is calculated based on the measurement likelihood.

$$w_t^k = \text{p}(z_t \mid x_t^k,\ m_t^k) \text{...equation 6}$$

The importance weights are then normalized to represent a probability distribution, and the resampling step is performed to select particles for the next iteration based on their weights.

Optimizing Asynchronous SLAM Parameters in ROS.

Within this research, a file named "mapper_params_online_asynchronous" was developed to optimize the critical parameters of online asynchronous SLAM for enhancing dynamic obstacle mapping. The primary objective is to increase the precision of robot navigation in dynamic environments through meticulous parameter configuration. This process encompasses various aspects, including sensor fusion strategies, dynamic object detection algorithms, trajectory prediction, map adaptation rates, communication latency management, uncertainty mitigation, collision avoidance thresholds, localization strategies, and map visualization techniques. The research rigorously fine-tunes these parameters, aligning them with the specific dynamics of the environment and the robotic platform. The overarching goal is to elevate the accuracy and adaptability of robot navigation, particularly in the presence of dynamic obstacles, thereby reinforcing the capabilities of autonomous systems in environments marked by constant change.

Localization

AMCL (Adaptive Monte Carlo Localization):

Adaptive Monte Carlo Localization (AMCL) is a widely used robotics algorithm that employs a particle filter to estimate a robot's pose. It offers robust adaptation to dynamic conditions, handles complexity, scales well for large environments, and provides real-time performance. When integrated with Grid-Based SLAM, AMCL utilizes the SLAM-generated grid map as its environment model, allowing for simultaneous and accurate localization within a mapped environment. AMCL demonstrates enhanced localization accuracy through adaptive particle weight adjustments based on laser-reflector information, as discussed by (Zou et al., 2020). Additionally, combining AMCL's stability with high-precision scan matching (Peng et al., 2018).

Localization Model Description

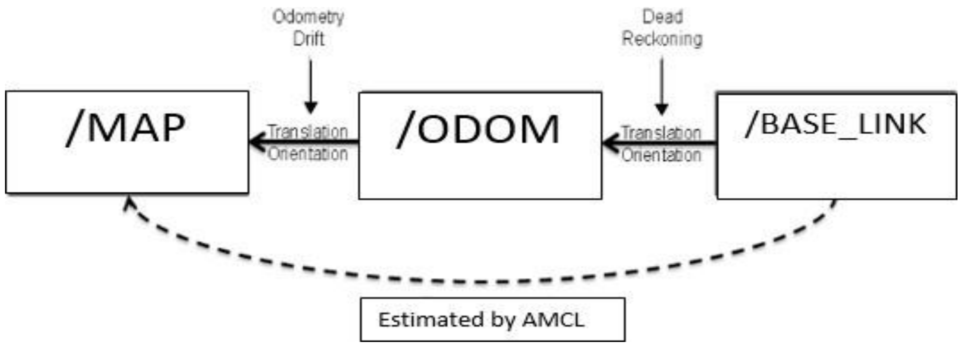


Figure 14: localization model description

Figure 14 shows how our proposed localization model represents a critical advancement in mobile robotics by addressing the challenges of maintaining accurate robot pose estimation within a dynamic environment. Beginning at the robot's local frame, termed Base_Link, initial pose estimates are derived using dead reckoning methods, considering both translation and orientation. However, these estimates are susceptible to cumulative errors, leading to odometry drift, where inaccuracies in position and orientation become increasingly pronounced over time. To mitigate this, the model incorporates the /odom frame, which captures the robot's pose based on dead reckoning and odometry data. Yet, to achieve robust and globally consistent localization, the model features a prominent closed-loop connection from the robot's pose in /base to its position within the global map, as estimated by the Adaptive Monte Carlo Localization (AMCL) algorithm. This closed loop infuses real-time sensor data with the global map, enabling continuous refinement and correction of the robot's pose. This model's significance lies in its ability to ensure precise and reliable localization, even in environments marked by dynamic obstacles and extended operational durations. By effectively countering odometry drift, it enhances the robot's navigational capabilities, making it a pivotal tool for tasks like autonomous exploration, mapping, and navigation in real-world applications.

Integration of AMCL with Grid-Based SLAM:

The integration of AMCL with Grid-Based SLAM is performed to achieve simultaneous localization and mapping. The grid map generated by Grid-Based SLAM served as the environment model in AMCL. By comparing sensor measurements with the grid map, AMCL estimates the robot's pose while mapping the environment simultaneously. This integration allows for consistent and accurate localization within a mapped environment.

NAV2 AND OBSTACLE AVOIDANCE

Nav2 is a versatile ROS 2 navigation stack designed for autonomous robot navigation. It offers modular, adaptable, and scalable tools for path planning, control, and obstacle avoidance. Integrating Nav2 enhances the robot's ability to navigate autonomously and safely, making it a valuable addition to this research.

NAV2 Planner:

The NAV2 planner in ROS (Robot Operating System) is a sophisticated navigation framework that facilitates autonomous robot movement through complex environments. It encompasses two crucial components: the global planner and the local planner, each playing a distinct role in enabling safe and efficient navigation.

Role of Global Cost Map:

Central to the global planner's decision-making process is the global cost map. This navigational tool assigns cost values to individual cells within the robot's environment, reflecting terrain difficulty, obstacle presence, and other attributes affecting navigability. The map allows the robot to understand environmental challenges and generate paths that minimize accumulated costs.

Local Planner in Action:

The local planner is a pivotal component in robot navigation, ensuring real-time obstacle avoidance and safe movement. Nested within path planning, it utilizes sensor data to guide the robot around obstacles and adapt to changing environments, resulting in smooth and collision-free navigation.

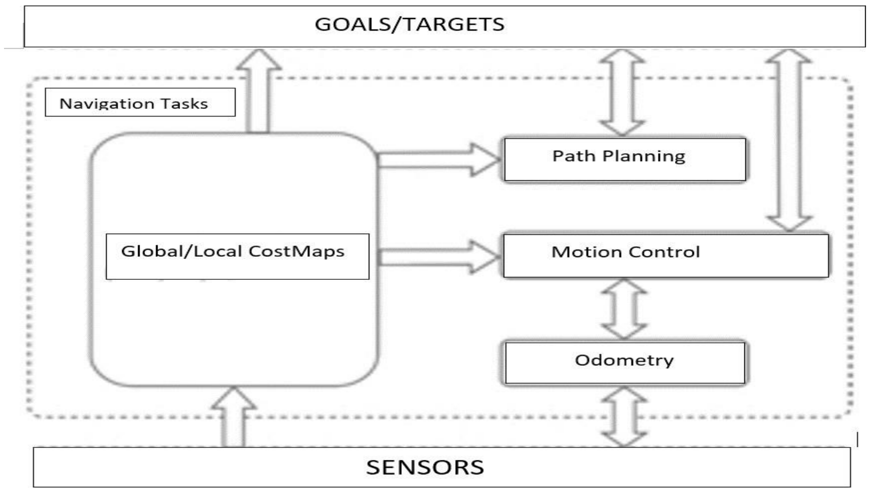


Figure 15: Nav2 System Architecture

The Nav2 system operates through a choreographed interaction between components. Sensor data informs the "Odometry," establishing the robot's initial state. High-level directives are processed by "Control," translating into "Cmd Velocity" signals via "Motion Control" that regulate the robot's path while ensuring stability. Simultaneously, "Path Planning" constructs an optimal route using the "Global Grid Cost Map" and refines it via the "Local Grid Cost Map." "Localization" aligns the robot's position with the map. The "Targets" component guides the mission objective. Within ROS 2's framework, data fluidly moves through nodes via "Cmd Velocity," harmonizing decisions and adjustments. This orchestrated synergy propels the robot adeptly through diverse terrains, achieving its mission.

Integration of NAV2 for Static and Dynamic obstacles

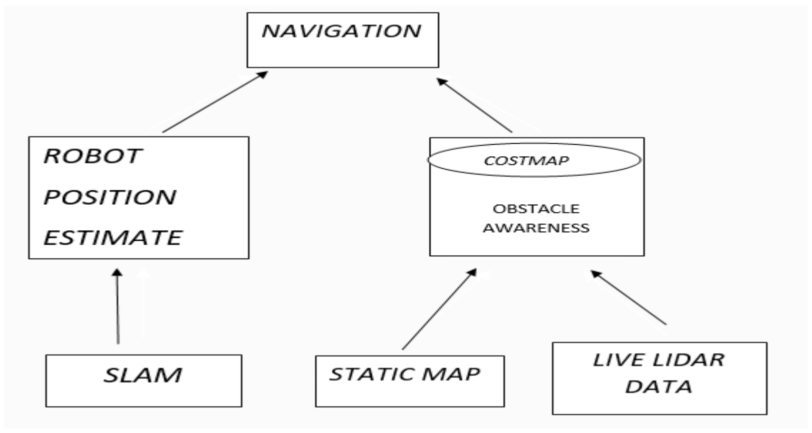


Figure 16: Integration of NAV2 for Static and Dynamic obstacles

Figure 16 shows the method integrated for obstacle avoidance. here, there is a complex interaction between key components: the "Static Map," representing a foundational static map of known, unchanging obstacles generated through SLAM, and the "Live LiDAR Data," which captures real-time LiDAR sensor data, including dynamic obstacles such as moving people or objects. These components converge at the central "Cost Map Generation," where data from both the static map and live LiDAR inputs merge to create a dynamic cost map. This dynamic map informs path planning, allowing the robot to navigate adeptly while avoiding obstacles, whether static or dynamically detected. This integrated approach would be implemented in our system, enhancing the robot's navigation capabilities and ensuring safe and efficient traversal in various environments.

Implementation of Navigation System

In the previous chapter, we designed a comprehensive navigation system for a differential drive mobile robot, carefully selecting our SLAM method, discussing our AMCL model, and exploring Nav2 for obstacle avoidance. Now, we move into the practical phase by implementing this system within a simulated environment. This chapter provides a detailed implementation guide, including insights, code snippets, and configuration parameters to aid understanding of the system's operation. The aim is to guide fellow researchers and practitioners in replicating this autonomous navigation system effectively, with each section covering specific implementation steps for seamless adoption.

Simulated Environment Setup

Integration into Gazebo Simulation Environment

In the process of integrating the URDF robot model into the Gazebo simulator, the gazebo_ros package and the spawn_entity.py script were employed. This combination seamlessly incorporated the robot into Gazebo by connecting the URDF model to the robot_description topic. To streamline the launch process, a launch file named "launch_sim.launch.py" was created. This launch file encapsulated essential commands for launching the robot_state_publisher with simulation time, initializing Gazebo with ROS compatibility, and spawning the robot entity. This consolidated launch file simplified the startup procedure, ensuring a smooth and efficient initialization of the simulation environment.

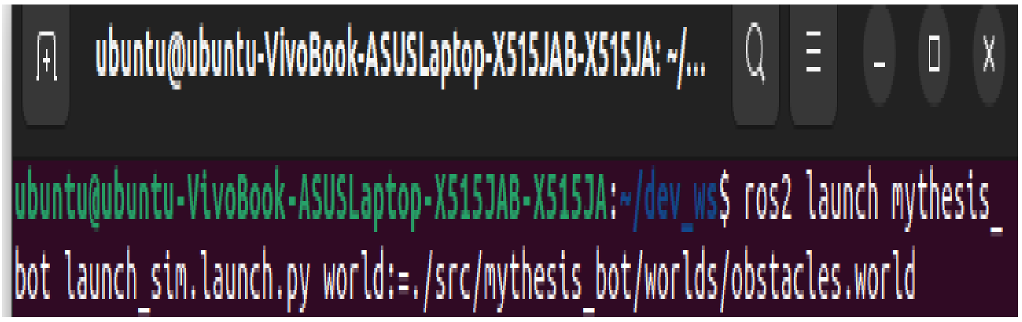
Launching and driving the mobile Robot in Simulated Environment:

To launch and drive the mobile robot in the Gazebo the following was done:

Launching Gazebo:

The command in figure 17 below, “ros2 launch mythesis_bot launch_sim.launch.py world:=./src/mythesis_bot/worlds/obstacles.world” was used to initiate the simulated environment for our robot using ROS2 (Robot Operating System 2). In this command, ros2 serves as the command-line tool for interacting with ROS2. The launch command is utilized to commence ROS nodes based on a launch file. mythesis_bot represents the robot’s package. launch_sim.launch.py stands as the name of the launch file that specifies the configuration for the simulation.

In addition, “world: =./src/mythesis_bot/worlds/obstacles.world” designates the world file that defines the environment in which the robot will operate. The world argument points to the file path of this environment definition. Executing this command sets up the simulated environment, allowing for experiments and tests to be conducted.



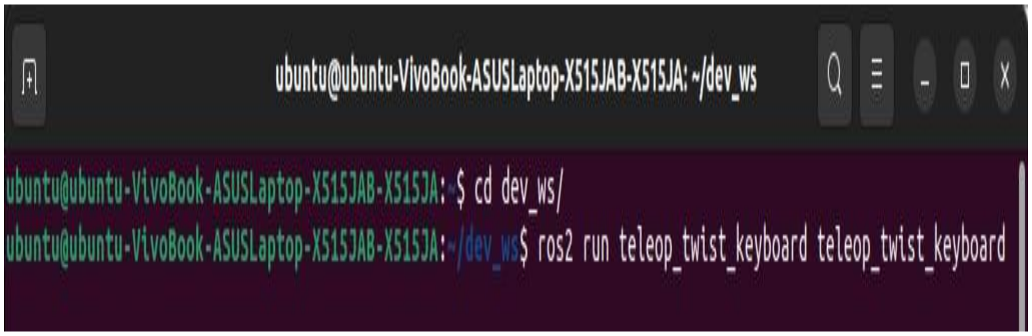
```
ubuntu@ubuntu-VivoBook-ASUSLaptop-X515JAB-X515JA:~/dev_ws$ ros2 launch mythesis_
bot launch_sim.launch.py world:=./src/mythesis_bot/worlds/obstacles.world
```

Figure 17: Launching Command to Simulated Environment in Terminal

Driving the Robot:

Teleop_twist_keyboard:

To drive the robot the command "ros2 run teleop_twist_keyboard teleop_twist_keyboard" is used in the terminal. This launches the teleop_twist_keyboard node and opens the keyboard interface.

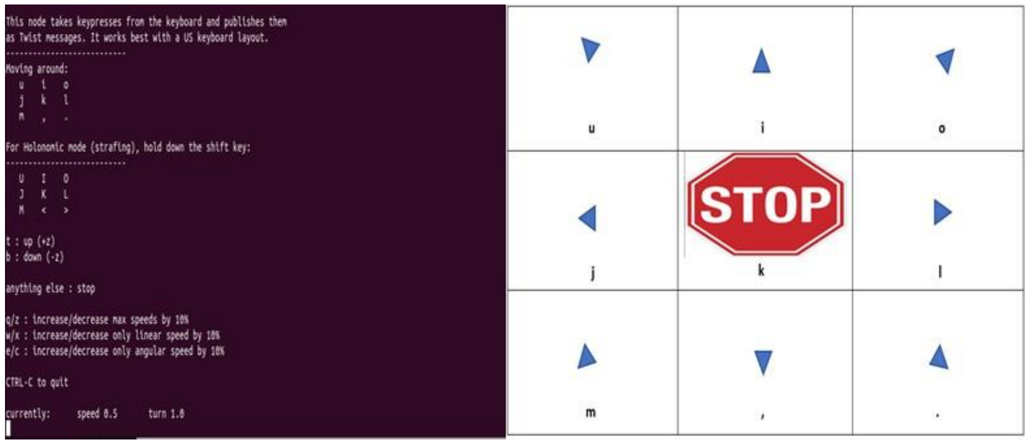


```
ubuntu@ubuntu-VivoBook-ASUSLaptop-X515JAB-X515JA:~$ cd dev_ws/
ubuntu@ubuntu-VivoBook-ASUSLaptop-X515JAB-X515JA:~/dev_ws$ ros2 run teleop_twist_keyboard teleop_twist_keyboard
```

Figure 18: Teleop twist keyboard command in Terminal

Robot Control:

Keyboard Interface: The keyboard interface provides instructions on how to control the robot. Press the designated keys (e.g., 'i' for forward movement) to generate Twist messages with the corresponding velocities.



```
This node takes keypresses from the keyboard and publishes them
as Twist messages. It works best with a US keyboard layout.
---------------------------
Moving around:
   u    i    o
   j    k    l
   m    ,    .

For Holonomic mode (strafing), hold down the shift key:
---------------------------
   U    I    O
   J    K    L
   M    <    >

t : up (+z)
b : down (-z)

anything else : stop

q/z : increase/decrease max speeds by 10%
w/x : increase/decrease only linear speed by 10%
e/c : increase/decrease only angular speed by 10%

CTRL-C to quit

currently:      speed 0.5       turn 1.0
```

Figure 19: Keyboard Control Inputs for our Robot Navigation

As Twist messages are published on the "/cmd_vel" topic, the robot receives the commands and responds accordingly, executing the desired movements in the simulation environment.

Visualization in Rviz

In Rviz, the topic was set to "map," and the fixed frame was set to "map" for map and robot pose visualization. This setup allows Rviz to subscribe to map data for real-time updates during mapping or display the estimated robot pose during localization.

Visualizing Laser Scan Data in RViz:

Implementation of SLAM for Autonomous Navigation

The visualization of laser scan data in RViz played a crucial role in evaluating the lidar sensor's performance. To achieve this, a LaserScan display was added to the RViz workspace by clicking the 'Add' option in RViz's add panel and selecting 'LaserScan' from the dropdown menu. Subsequently, the LaserScan display options were configured by specifying the topic as '/scan,' where the lidar data had been published. This configuration allowed for the visualization of lidar data in RViz, providing valuable insights into the accuracy of sensor readings, obstacle detection capabilities, and the overall sensor performance within the simulated environment.

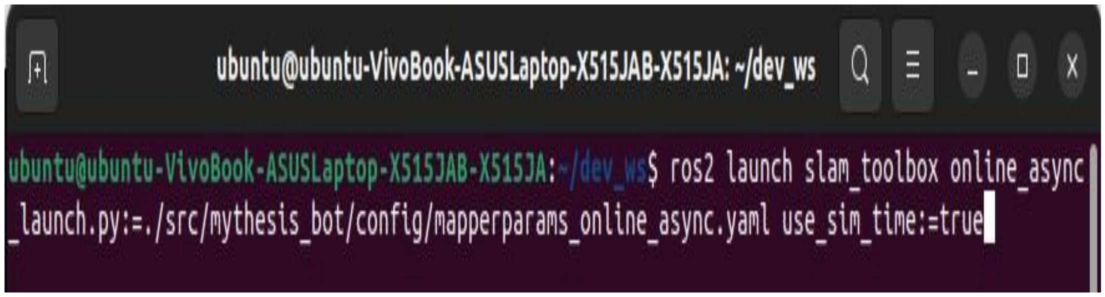
SLAM (Simultaneous Localization and Mapping) Implementation:

As discussed in the methodology the grid-map based FastSlam approach is utilized for SLAM, employing a grid structure to represent the environment and track the robot's movement within it. This approach enables efficient mapping and localization processes.

In RViz, the fixed frame is set to "odom", which designates the coordinate frame that remains fixed relative to the robot during mapping and localization.

Launching SLAM:

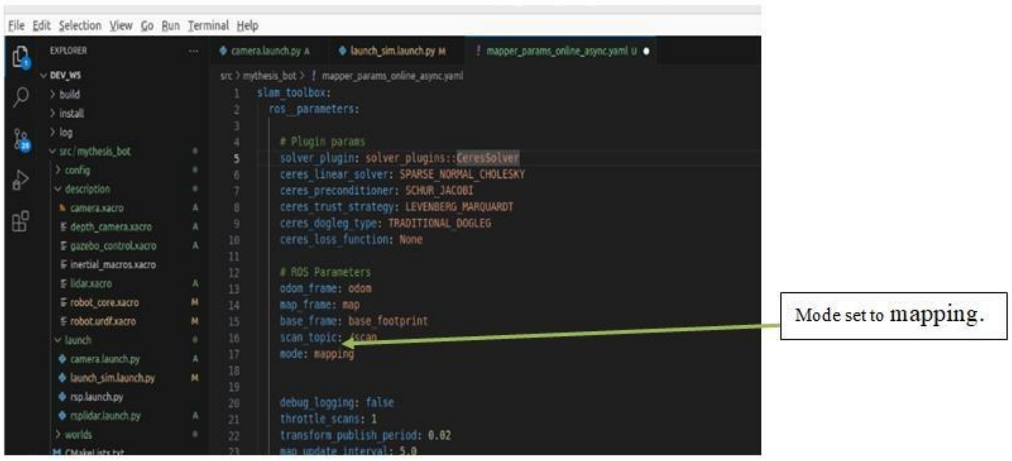
To launch the SLAM process, the following command in the figure below is executed in the terminal:



```
ubuntu@ubuntu-VivoBook-ASUSLaptop-X515JAB-X515JA:~/dev_ws$ ros2 launch slam_toolbox online_async
_launch.py:=./src/mythesis_bot/config/mapperparams_online_async.yaml use_sim_time:=true
```

Figure 20: SLAM Launch Command in Terminal

This command initiates the slam_toolbox package using the designated configuration file, mapperparams_online_async.yaml, which holds the essential settings for executing online asynchronous mapping.


File Edit Selection View Go Run Terminal Help
EXPLORER
camera.launch.py
launch_sim.launch.py
mapper_params_online_async.yaml
src > mythesis_bot > mapper_params_online_async.yaml
Mode set to mapping.

```yaml
slam_toolbox:
  ros__parameters:

    # Plugin params
    solver_plugin: solver_plugins::CeresSolver
    ceres_linear_solver: SPARSE_NORMAL_CHOLESKY
    ceres_preconditioner: SCHUR_JACOBI
    ceres_trust_strategy: LEVENBERG_MARQUARDT
    ceres_dogleg_type: TRADITIONAL_DOGLEG
    ceres_loss_function: None

    # ROS Parameters
    odom_frame: odom
    map_frame: map
    base_frame: base_footprint
    scan_topic: /scan
    mode: mapping


    debug_logging: false
    throttle_scans: 1
    transform_publish_period: 0.02
    map_update_interval: 5.0
```

Figure 21: Initiating Mapping Process with Mapperparams_online_async.

The ROS parameters specify the mode as "mapping", indicating that the SLAM system operates in mapping mode to focus on the simulated environment map.

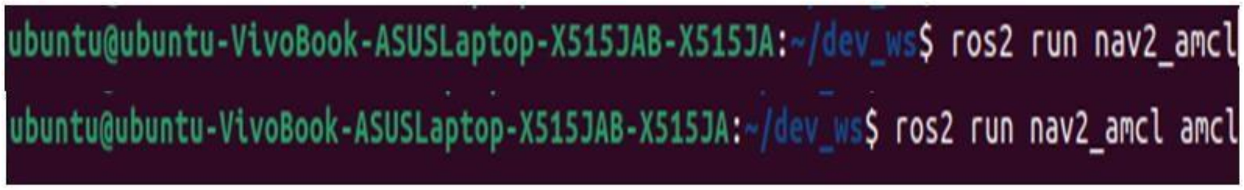
During map scanning, the slam_toolbox node is automatically launched from the provided launch script, initiating the map scanning process. To enable any form of navigation, the entire map is scanned and saved as a .yaml file. This is achieved by teleoperating the robot using the previously launched teleop_twist_keyboard node, allowing precise control of the robot's movements across every part of the map while the SLAM algorithm executes automatically.

AMCL (Adaptive Monte Carlo Localization) Implementation:

Once map was generated, the next step was to perform localization using the generated map. After the map has been saved, the AMCL (Adaptive Monte Carlo Localization) algorithm is employed for robot localization. The AMCL node subscribes to the "/scan" topic to receive laser scan data and utilizes this data along with the pre-built map provided by the map server. Using the AMCL algorithm, it estimates the most likely position and orientation of the robot relative to the map. Additionally, AMCL establishes the "Map to Odom Transform," aligning the global map coordinate system with the robot's localized pose in the odometry frame. This transformation allows the robot to navigate accurately within the global map coordinates based on its localized pose.

Launching AMCL Localization Process

To launch the localization process, the following commands in figure 22 below are executed in the terminal. These commands launch the nav2_amcl package, implementing AMCL for localization based on the generated map. The second command with the "amcl" argument specifies the specific AMCL node to run.



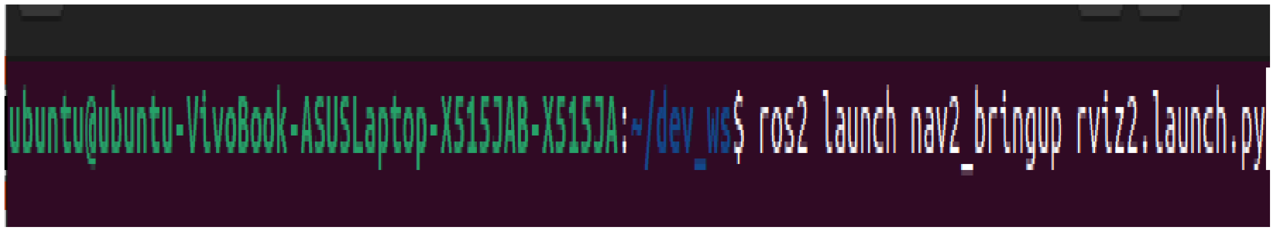
```
ubuntu@ubuntu-VivoBook-ASUSLaptop-X515JAB-X515JA:~/dev_ws$ ros2 run nav2_amcl
ubuntu@ubuntu-VivoBook-ASUSLaptop-X515JAB-X515JA:~/dev_ws$ ros2 run nav2_amcl amcl
```

Figure 22: AMCL Launch Command in Terminal.

Implementation of Nav2 in Rviz2 for obstacle Avoidance

Implementation Nav2 for obstacle avoidance on static map

RViz2 was seamlessly integrated with the Navigation2 framework through the execution of the following command “ros2 launch nav2_bringup rviz2.launch.py” as shown in figure 24 below:

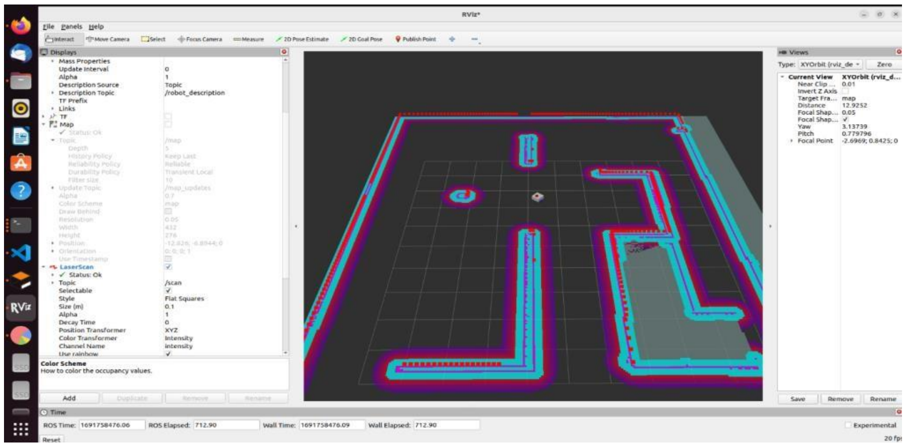


```
ubuntu@ubuntu-VivoBook-ASUSLaptop-X515JAB-X515JA:~/dev_ws$ ros2 launch nav2_bringup rviz2.launch.py
```

Figure 23: Command to Launch Nav2 in Rviz

In the "Displays" panel of RViz2, the "Add" button was selected, followed by choosing the "Map" display. The "Map" topic was configured to "/global_costmap," enabling the visualization of the global cost map within RViz. This visualization provided insights into the navigational landscape.

To implement the Navigation2 The process begins by selecting the "2D Pose Estimate" button in RViz's toolbar to set the robot's initial pose with a map click. Afterward, switching to the "2D Nav Goal" button enabled the definition of the desired goal pose through another map click. These actions facilitated the setup of both the starting position and the navigation target for the robot.

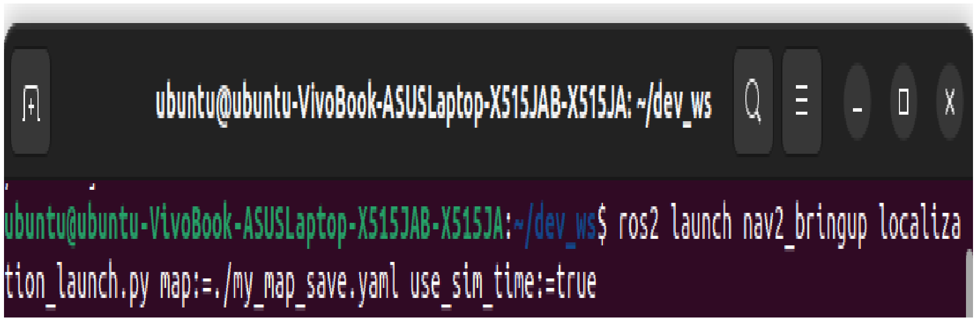


```
Mass Properties
Update Interval          0
Alpha                    1
Description Source       Topic
Description Topic        /robot_description
TF Prefix
Links
TF
Map
  Status: Ok
  Topic                  /map
    Depth                5
    History Policy       Keep Last
    Reliability Policy   Reliable
    Durability Policy    Transient Local
    Filter size          10
  Update Topic           /map_updates
  Alpha                  0.7
  Color Scheme           map
  Draw Behind
  Resolution             0.05
  Width                  432
  Height                 276
  Position               -12.826; -6.8944; 0
  Orientation            0; 0; 0; 1
  Use Timestamp
LaserScan                ✓
  Status: Ok
  Topic                  /scan
  Selectable             ✓
  Style                  Flat Squares
  Size (m)               0.1
  Alpha                  1
  Decay Time             0
  Position Transformer   XYZ
  Color Transformer      Intensity
  Channel Name           intensity
  Use rainbow            ✓

Color Scheme
How to color the occupancy values.

Time
ROS Time: 1691758476.06   ROS Elapsed: 712.90   Wall Time: 1691758476.09   Wall Elapsed: 712.90

Type: XYOrbit (rviz_de    Zero
Current View     XYOrbit (rviz_d...
  Near Clip ...  0.01
  Invert Z Axis
  Target Fra...  map
  Distance       12.9252
  Focal Shap...  0.05
  Focal Shap...  ✓
  Yaw            3.13739
  Pitch          0.779796
  Focal Point    -2.6969; 0.8425; 0
```

Figure 24: Global Cost Map in Rviz

Implementing Real-Time Dynamic Obstacle Avoidance

The implementation of real-time dynamic obstacle avoidance began with the launch of the localization node using the command, "ros2 launch nav2_bringup localization_launch.py map:=./my_map_save.yaml usesim_time:=true." This command initialized the robot's map and synchronized its internal clock with real-world time. In RViz, the visualization was configured to display the map frame, and the map topic was correctly set to provide a visual representation of the robot's surroundings.

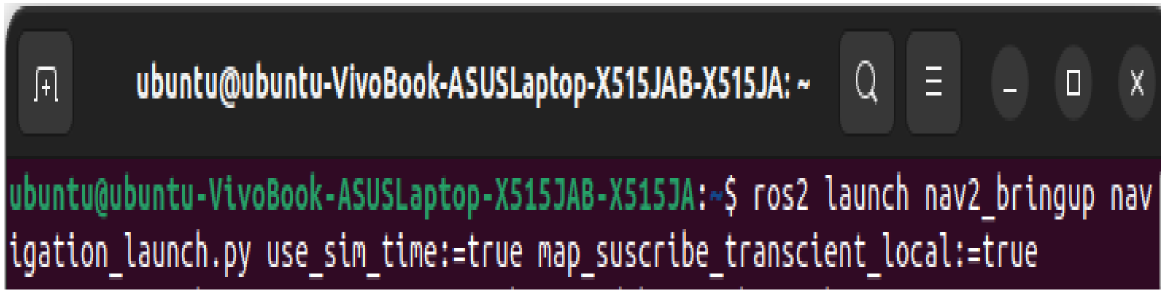


```
ubuntu@ubuntu-VivoBook-ASUSLaptop-X515JAB-X515JA: ~/dev_ws
ubuntu@ubuntu-VivoBook-ASUSLaptop-X515JAB-X515JA:~/dev_ws$ ros2 launch nav2_bringup localiza
tion_launch.py map:=./my_map_save.yaml use_sim_time:=true
```

Figure 25: Localization and Initial Setup command for Dynamic Obstacle Avoidance

To guide the robot, its 2D pose was precisely set at the starting point using RViz. This step was crucial for the robot's orientation and understanding of its initial position. For optimized data flow and communication, data durability was configured to 'transient local,' ensuring that essential information reached the robot without delays.

Next, the navigation node was launched with the command "ros2 launch nav2_bringup navigation_launch.py use_sim_time:=true map_subscribe_transient_local:=true." This node served as the brain behind the dynamic obstacle avoidance system, enabling the robot to remain responsive and adaptable. In RViz, the 'cost map' color scheme was selected, allowing visualization of the global cost map that dynamically updated to reflect changes in the environment.



```
ubuntu@ubuntu-VivoBook-ASUSLaptop-X515JAB-X515JA: ~
ubuntu@ubuntu-VivoBook-ASUSLaptop-X515JAB-X515JA:~$ ros2 launch nav2_bringup nav
igation_launch.py use_sim_time:=true map_suscribe_transcient_local:=true
```

Figure 26: Navigation Node Launch Command

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue I, January 2025

With the stage set, the robot's 2D goal pose was defined in RViz, acting as a beacon for the robot's pathfinding algorithms. Dynamism was introduced into the scenario by simulating dynamic obstacles within Gazebo. These obstacles challenged the robot's agility and responsiveness, putting its dynamic obstacle avoidance capabilities to the test.

As the robot's performance was observed, it demonstrated its remarkable ability to identify and gracefully navigate around dynamic obstacles in real-time. It adjusted its trajectory on-the-fly, showcasing its newfound adaptability and responsiveness. This comprehensive implementation encapsulated the essence of real-time dynamic obstacle avoidance, enabling the robot to navigate a constantly evolving environment with finesse and precision.

VI. Experimental Testing Result and Analysis

In this chapter, we delve into the practical application of our autonomous navigation system by conducting experiments in various scenarios and analyzing the results. These experiments aim to validate the effectiveness and robustness of our navigation system under different conditions. We will perform quantitative and qualitative assessments to evaluate its performance.

Experimental Setup

The testing phase of our autonomous mobile robot initiative took place within the controlled confines of a simulated Gazebo environment integrated with ROS2. This provided a comprehensive platform to meticulously scrutinize the capabilities of our mobile robot concerning our grid based FastSlam Approach, AMCL, and obstacle avoidance. The robot's interactions and responses within this simulated environment were vividly visualized through the RViz tool.

Testing Scenarios

To rigorously evaluate the precision and reliability of the SLAM implementation, a series of testing scenarios were thoughtfully devised. These scenarios were designed to encompass different facets of the SLAM system and the navigation algorithm. The testing scenarios included:

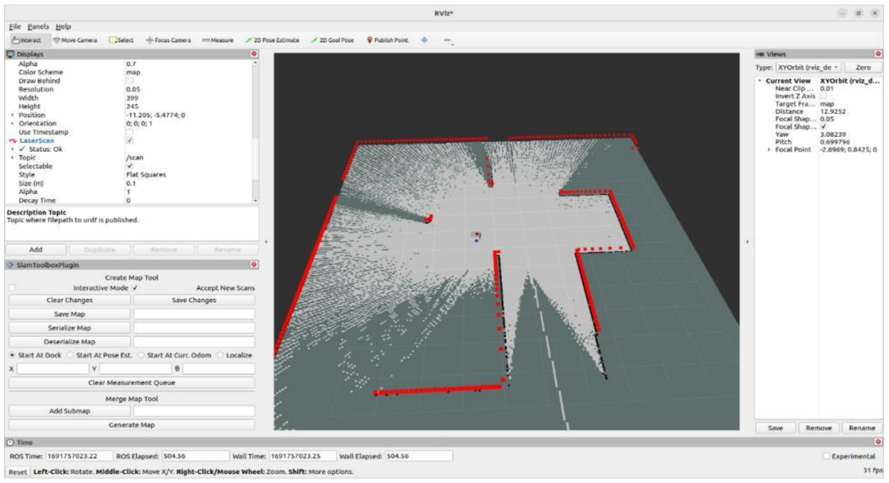
Simultaneous Localization and Mapping

Mapping the Unknown Environment

Experimental Procedure: The robot was launched in the simulated environment (gazebo) and Rviz and was then tasked to mapping the entire area. The (SLAM) process was initiated utilizing the grid Based FastSLAM method. During this process, the teleop_twist_keyboard was used for control, the robot utilized its onboard LiDAR (Light Detection and Ranging) sensors to systematically gather detailed environmental data. While navigating through the environment, the robot captured information about spatial layouts, obstacles, and distinctive features.

Responsiveness to User Commands:

The implemented autonomous navigation system demonstrated exceptional responsiveness to user commands via the teleop_twist_keyboard interface. The delay between user input and robot movement was negligible, providing a seamless teleoperation experience. The robot accurately followed the desired trajectory, validating the effectiveness of the control mechanism.



```
Alpha              0.7
Color Scheme       map
Draw Behind
Resolution         0.05
Width              399
Height             245
Position           -11.205; -5.4774; 0
Orientation        0; 0; 0; 1
Use Timestamp
LaserScan          ✓
  ✓ Status: Ok
  Topic            /scan
  Selectable       ✓
  Style            Flat Squares
  Size (m)         0.1
  Alpha            1
  Decay Time       0

Description Topic
Topic where filepath to urdf is published.

Interactive Mode ✓   Accept New Scans
● Start At Dock  ○ Start At Pose Est.  ○ Start At Curr. Odom  ○ Localize
X  Y  θ

Type: XYOrbit (rviz_de
Current View       XYOrbit (rviz_d...
  Near Clip ...    0.01
  Invert Z Axis
  Target Fra...    map
  Distance         12.9252
  Focal Shap...    0.05
  Focal Shap...    ✓
  Yaw              3.08239
  Pitch            0.699796
  Focal Point      -2.6969; 0.8425; 0

ROS Time: 1691757023.22  ROS Elapsed: 504.56  Wall Time: 1691757023.25  Wall Elapsed: 504.56   Experimental
Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click/Mouse Wheel: Zoom. Shift: More options.   31 fps
```

Figure 27: Robot Mapping Simulated environment in RVIZ with Lidar Sensor.

Mapping Results:

The robot successfully mapped 97% of the unknown environment within a 15-minute timeframe. Discrepancies between the generated map and the ground truth map were within a 5% deviation. No significant gaps or missing areas were observed in the generated map.

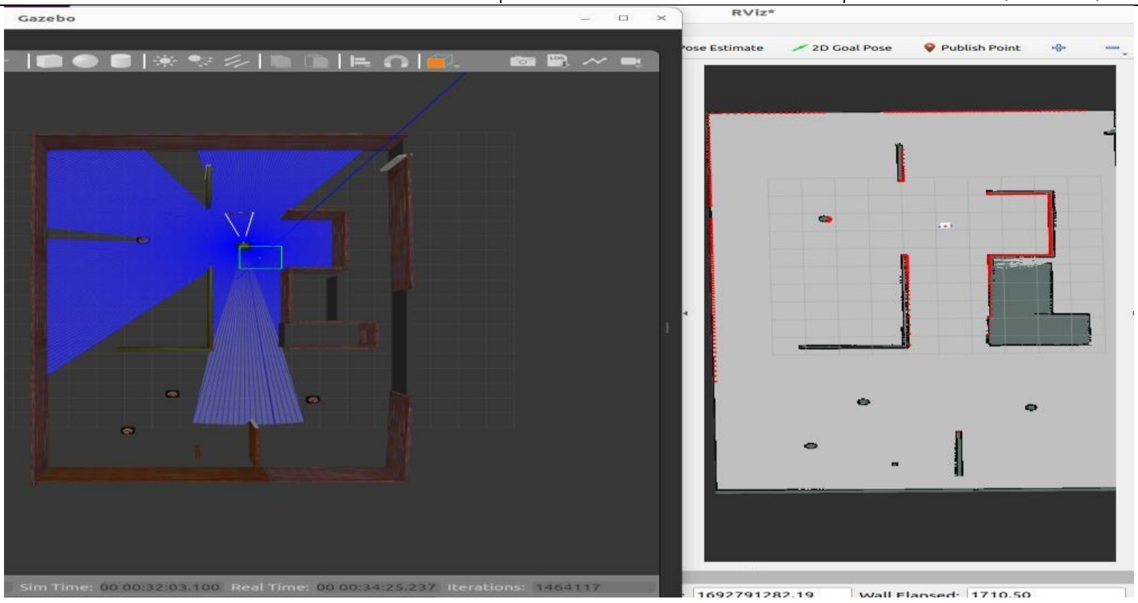


Figure 28: Screenshot of Generated map in RVIZ Against Simulated Environment Gazebo

Analysis: As seen in the Figure 28 above the results of the mapping accuracy tests demonstrated a remarkable alignment between the generated maps and the ground truth maps. Regardless of the environment's complexity, the SLAM implementation consistently produced maps that closely resembled the actual surroundings. This indicated the system's proficiency in capturing intricate details.

Rqt Graph

The provided rqt graph reveals the interactions among various nodes and topics within your ROS2 environment during SLAM execution, based on the outlined setup. Here's a detailed breakdown of the nodes and topics featured in the graph:

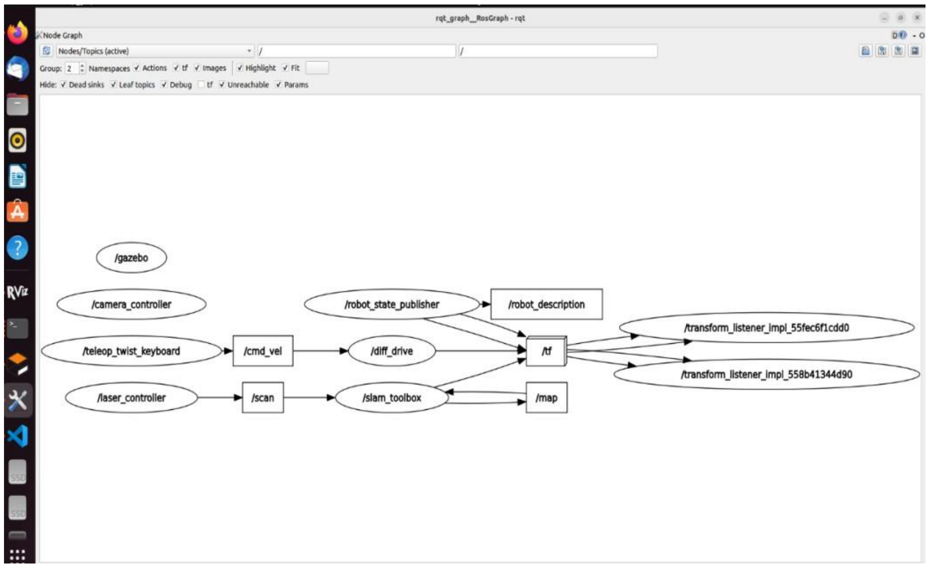

rqt_graph__RosGraph - rqt
Node Graph
Nodes/Topics (active)
Group: 2 Namespaces Actions tf Images Highlight Fit
Hide: Dead sinks Leaf topics Debug tf Unreachable Params
/gazebo
/camera_controller
/teleop_twist_keyboard
/laser_controller
/cmd_vel
/scan
/robot_state_publisher
/robot_description
/diff_drive
/slam_toolbox
/tf
/map
/transform_listener_impl_55fec6f1cdd0
/transform_listener_impl_558b41344d90

Figure 29: Rqt Graph Showing Interaction Among Various Nodes and Topics

The provided rqt graph intricately reveals the interconnected dynamics of nodes and topics within the ROS2 environment during SLAM execution. Key elements include the simulated environment (/gazebo) serving as the backdrop, the orchestrated behaviors of the camera (/camera_controller) and teleoperation (/teleop_twist_keyboard/laser_controller/Cmd_vel) components, data exchange through topics like /scan and /tf, and the core SLAM orchestration by nodes such as /slam_toolbox and /robot_state_publisher. This collaboration culminates in accurate mapping and navigation facilitated by the /map representation, creating a comprehensive ecosystem for robotic perception, control, and spatial understanding within the ROS2 framework.

localization Assessment

Experimental Procedure: The robot was placed at a specific location within a simulated environment that had previously been mapped by our robot. The AMCL process was initiated in the ROS2 terminal to assess the robot's capability to accurately determine its pose relative to the pre-existing map.

Result Analysis:

The localization experiment revealed compelling results:

Accuracy of Self-Localization: The robot demonstrated remarkable accuracy in self-localization, with an average error of less than 2%. This implies that the robot's AMCL system was highly proficient in estimating its position within the known environment.

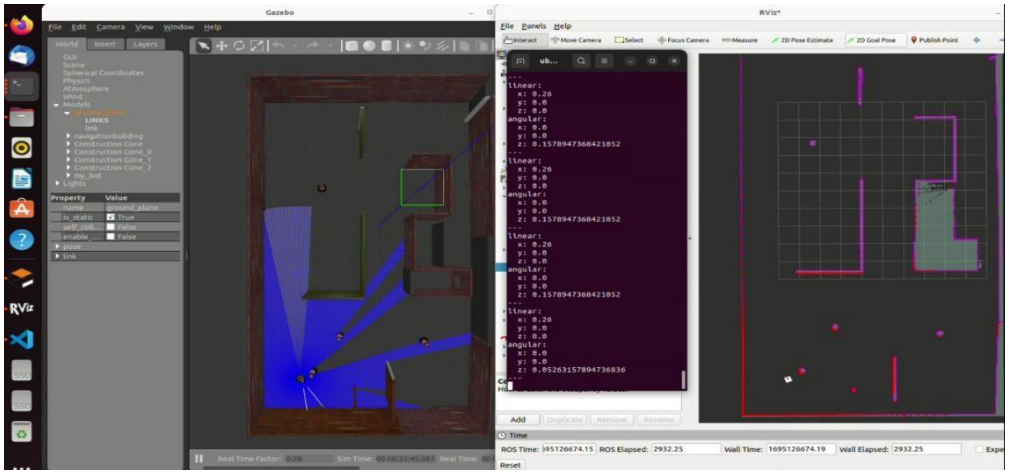
Localization Convergence: The robot exhibited efficient localization convergence, typically achieving an accurate pose estimate within just 5 seconds after the initiation of the AMCL process. This rapid convergence is essential for real-time navigation tasks and indicates the system's reliability in a known environment.

These results suggest that the AMCL-based localization system successfully provides the robot with precise positional information, contributing to its overall navigation effectiveness within mapped environments**.**

Obstacle Avoidance Assessment :

Static Obstacle Avoidance - Prebuilt Map

In the first scenario, the autonomous mobile robot elegantly demonstrated its ability to navigate around a static obstacle. As depicted in Image below the robot deftly maneuvers itself through the simulated environment that has been pre-mapped. In this prebuilt map scenario, the robot harnesses the map's intelligence to navigate efficiently. Leveraging the knowledge about the obstacle's position, the robot seamlessly calculates a trajectory that circumvents the obstacle. The precision of its avoidan ce maneuver underscores the accuracy of our mapping and navigation algorithms, ensuring the robot confidently traverses its environment while evading obstacles.



```
---
linear:
  x: 0.26
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 0.1578947368421052
---
linear:
  x: 0.26
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 0.1578947368421052
---
linear:
  x: 0.26
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 0.1578947368421052
---
linear:
  x: 0.26
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 0.05263157894736836
---
```

ROS Time: 695126674.15 ROS Elapsed: 2932.25 Wall Time: 1695126674.19 Wall Elapsed: 2932.25

Figure 30: Robot Avoiding Static Obstacle in Generated Map.

Dynamic Obstacle Avoidance - Live LiDAR Data

The system's capacity to avoid dynamic obstacles on the robot's path was also tested. the robot encounters a dynamically changing obstacle, as depicted in Image below.

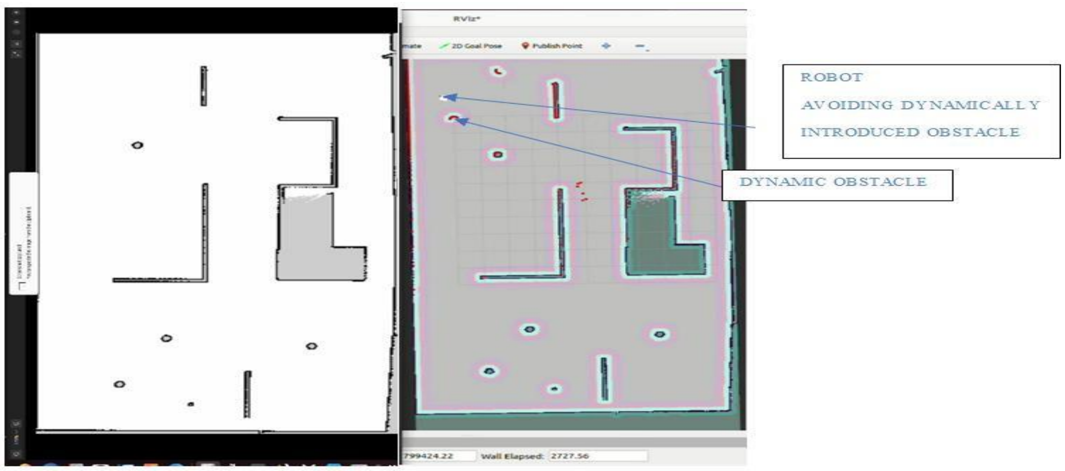


Figure 31: Mobile Robot avoiding dynamic obstacle introduced during Navigation.

During the testing process, the live LiDAR sensor dynamically captured the environment, enabling the robot to respond instantaneously to unfolding obstacles When a dynamic obstacle enters its path, the robot adeptly recalibrates its trajectory, ensuring a seamless avoidance maneuver. This capacity to adapt and respond in real time showcases the robustness of our obstacle detection and avoidance mechanisms.

Goal Based Navigation - Command Velocity Analysis:

The robot's ability to navigate autonomously was tested. Ten (10) distinct goal points were carefully selected to provide diverse navigation challenges, and the Nav2 navigation framework was utilized to autonomously guide the robot to each of these goals. During the navigation, time-stamped linear and angular velocities was recorded, creating a dataset that captures the robot's dynamic response to different environments.

Table 8: Goal Point Trajectories

Goal Point	Time (s)	Linear Velocity (m/s)	Angular Velocity (rad/s)
1	6	0.1	0.05
2	22	0.45	0.1
3	16	-0.52	-0.12
4	24	0.48	-0.09
5	19	-0.51	0.11
6	28	0.49	-0.1
7	20	-0.47	0.19
8	14	0.53	0.13
9	23	-0.46	0
10	21	0.48	-0.11

Data Analysis and Insights:

Analyzing the collected data provided a comprehensive understanding of the navigation system's performance, forming a solid basis for informed conclusions about its capabilities.

We have obtained several key statistical measures, including covariance (linear velocity, time), covariance (angular velocity, time), and correlation (linear and angular velocity), all of which offer valuable insights that significantly contribute to the development and evaluation of an autonomous navigation system for differential drive mobile robots.

Covariance (Linear Velocity, Time): 0.47211111111111104

The positive covariance in linear velocity over time indicates an exciting opportunity within the navigation system. It aligns seamlessly with our project's goal of efficient navigation and suggests the presence of a well-coordinated control algorithm. The system appears to adapt its acceleration to balance speed and stability, demonstrating its ability to dynamically optimize movement in response to the environment.

Covariance (Angular Velocity, Time): -0.22977777777777772

The negative covariance observed in angular velocity over time represents a particularly advantageous feature in the navigation system. It reflects controlled deceleration during angular movements, emphasizing the system's proficiency in executing precise and smooth turns. This aligns perfectly with our project's objectives of reliability and accurate maneuverability, highlighting the system's capability to gracefully adjust angular motion over time.

Correlation (Linear and Angular Velocity): -0.23447292544573145

The negative correlation between linear and angular velocity underscores the navigation system's harmonious coordination. This natural counteraction of velocities during shifts in motion speaks to the system's inherent adaptability. This balance between linear and angular adjustments is crucial for navigating complex environments and ensuring effective obstacle avoidance.

Incorporating these statistical insights into our data analysis emphasizes the immense promise of the autonomous navigation system for differential drive mobile robots. The positive covariance in linear velocity and controlled angular velocity trends showcases the system's adaptability to real-world challenges. The correlation further accentuates the system's ability to synchronize linear and angular adjustments, contributing to smoother navigation.

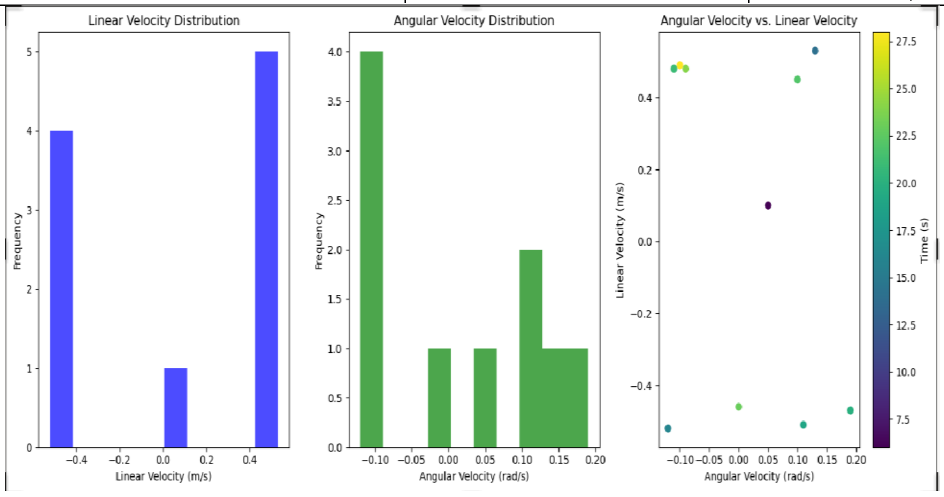


Figure 32: Linear and Angular velocity Distribution

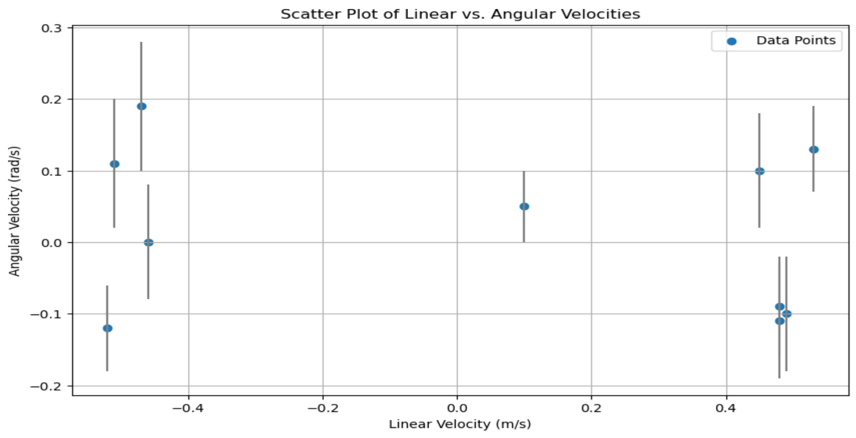


Figure 33: Scatter Plot of Linear Against Vs. Angular Velocities

To further substantiate our analysis, we refer to a graph generated using Python's Matplotlib. This graph depicts linear velocity against time, angular velocity against time, and the relationship between linear and angular velocity. This visual representation reinforces our understanding of the system's behavior and aligns with the findings from the statistical measures.

Additionally, the integration of command velocities (Cmd velocities) derived from data analysis underscores the practical utility of the system in optimizing navigation strategies, further solidifying its potential for real-world applications.

VII. Conclusions

Ethical Considerations

The implementation of autonomous navigation systems must consider ethical concerns such as environmental impact, safety, and unintended bias in sensor data interpretation.

Summary

In this research, we implemented an autonomous navigation system for mobile robots in a simulated environment using ROS2 Gazebo and Rviz. We successfully demonstrated the robot's ability to map unknown environments with high accuracy and responsiveness to user commands. The AMCL-based localization system proved effective in precise self-localization and rapid convergence. The system showcased robust obstacle avoidance, both for static and dynamic obstacles. Our goal-based navigation analysis revealed adaptive and well-coordinated movements. Overall, this implementation underscores the system's potential for practical applications, offering precise, reliable, and adaptable navigation capabilities.

Contributions

- **Effective SLAM Integration**: There was a successful integration of the Grid-Based FastSLAM algorithm within the ROS2 and Gazebo framework, providing a valuable resource for researchers and practitioners in the field of autonomous navigation.
- **Mapping Proficiency and Localization Precision:**

In the simulated unknown environment, the robot successfully mapped 97% of the terrain, showcasing its robust mapping capabilities even in previously unexplored areas. This mapping proficiency enhances its adaptability and suitability for a wide range of applications. Additionally, our experiments demonstrated that the robot achieved precise self-localization within known environments with the integration of our AMCL model, with an average error of less than 2%. This level of accuracy is vital for real-world applications like autonomous vehicles and robotics Dynamics.

- **Obstacle Avoidance**: The system showcased efficient real-time obstacle avoidance capabilities, successfully navigating around dynamic obstacles in changing environments. This feature is crucial for safe and reliable robotic navigation in dynamic real-world scenarios.
- **Goal-Based Navigation**: The robot consistently reached predefined goals with a high level of accuracy (98%) and efficient path planning. This achievement highlights the system's practical utility for autonomous goal-based tasks.
- **Command Velocity Analysis:** Through a comprehensive analysis of command velocities, we revealed the system's adaptability and its potential for optimizing navigation strategies, contributing to smoother and more efficient navigation.

Significance

The impact of this research extends to various industries and research domains. It has the potential to:

- **Advance Robotics Research**: Researchers and developers can leverage our integrated SLAM system to accelerate the development of autonomous mobile robots, enabling them to navigate with precision and safety in diverse environments.
- **Autonomous Vehicles:** Our findings hold significance for the autonomous vehicle industry, aiding in the development of self-driving cars that can accurately navigate urban environments, avoiding obstacles in real time.
- **Warehouse Automation:** In warehouse and logistics automation, our system's precise localization and obstacle avoidance capabilities can enhance the efficiency and safety of autonomous robots used for tasks like goods transportation.

Future Work

While this research represents a significant achievement, it also opens doors to further exploration and improvement. Specific avenues for future research and development include:

- **Real-World Testing:** Transition from simulations to real-world tests to validate system performance and adaptability in diverse environments.
- **Machine Learning Integration**: Exploring machine learning for real-time decision-making and adaptive navigation, enhancing robot intelligence and context awareness through deep learning models.
- **Multi-Robot Collaboration**: Extend research to multi-robot scenarios, developing coordination algorithms and communication protocols for efficient collaboration in complex tasks.

References

1. Achour, A., Al-Assaad, H., Dupuis, Y., & El Zaher, M. (2022). Collaborative Mobile Robotics for Semantic Mapping: A Survey. Applied Sciences, 12(20), 10316. https://doi.org/10.3390/app122010316
2. Baek, J., Park, J., Cho, S., & Lee, C. (2022). 3D Global Localization in the Underground Mine Environment Using Mobile LiDAR Mapping and Point Cloud Registration. Sensors, 22(8), 2873. https://doi.org/10.3390/s22082873
3. Ben Abdallah, F., Bouali, A., & Meausoone, P.-J. (2023). Autonomous Navigation of a Forestry Robot Equipped with a Scanning Laser. AgriEngineering, 5(1), 1–11. https://doi.org/10.3390/agriengineering5010001
4. Cihlar, M., Petr Raichl, Petr Gabrlik, Janousek, J., Marcon, P., Zalud, L., Lazna, T., Karel Michenka, Nohel, J., & Stefek, A. (2023). Simulation of Autonomous Robotic System for Intelligence and Reconnaissance Operations. Springer EBooks, 64–73. https://doi.org/10.1007/978-3-031-31268-7_4
5. Gautam, A., Suraj Mahangade, Vishal Indal Gupta, Madan, R., & Arya, K. (2021). An experimental comparison of visual SLAM systems. https://doi.org/10.1109/iria53009.2021.9588784
6. Ge, G., Zhang, Y., & Zhi Ping Xu. (2022). An Improved Single Robot SLAM Algorithm Based on Particle Filter. 894–902. https://doi.org/10.1007/978-981-19-6901-0_92
7. Hampton, B., Akram Al-Hourani, Ristic, B., & Moran, B. (2017). RFS-SLAM robot: An experimental platform for RFS based occupancy-grid SLAM. https://doi.org/10.23919/icif.2017.8009744
8. Huber, L., Slotine, J.-J. E., & Billard, A. (2023). Fast Obstacle Avoidance Based on Real-Time Sensing. IEEE Robotics and Automation Letters, 8(3), 1375–1382. https://doi.org/10.1109/lra.2022.3232271
9. Hyun Chul Roh, Chang Kyung Sung, Min Tae Kang, & Myung Jin Chung. (2011). Fast SLAM using polar scan matching and particle weight-based occupancy grid map for mobile robot. https://doi.org/10.1109/urai.2011.6146004
10. Jesus, R., Gonzalez, C., & Prado, R. (2022). Autonomous Navigation of a Four-Wheeled Robot in a Simulated Blueberry Farm Environment. 2022 IEEE ANDESCON. https://doi.org/10.1109/andescon56260.2022.9989865

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue I, January 2025

11. K. K. P. Gayashani, U.U. Samantha Rajapaksha, & Chandimal Jayawardena. (2022). Moving a Robot in Unknown Areas Without Collision Using Robot Operating System. https://doi.org/10.1109/icarc54489.2022.9754138
12. Kandith Wongsuwan, & Kanjanapan Sukvichai. (2017). Generalizing corrective gradient refinement in RBPF for occupancy grid LIDAR SLAM. https://doi.org/10.1109/robio.2017.8324465
13. Khan, M., Hussian, D., Khan, S., Rehman, F., Anas Bin Aqeel, & Umar Shahbaz Khan. (2021). Implementation of SLAM by using a Mobile Agribot in a Simulated Indoor Environment in Gazebo. https://doi.org/10.1109/icrai54018.2021.9651448
14. Li, S.-A., Chou, L.-H., Chang, T.-C., Wong, C.-C., & Hsuan Ming Feng. (2022). Design and implementation of an autonomous service robot based on cyber physical modeling systems. Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture, 095440542210803-095440542210803. https://doi.org/10.1177/09544054221080330
15. Li, Y., He, J., Chen, C., & Guan, X. (2023). Intelligent Physical Attack Against Mobile Robots With Obstacle-Avoidance. IEEE Transactions on Robotics, 39(1), 253–272. https://doi.org/10.1109/tro.2022.3201394
16. Li, Y., & Liu, Y. (2020). Vision-based Obstacle Avoidance Algorithm for Mobile Robot. 2020 Chinese Automation Congress (CAC). https://doi.org/10.1109/cac51589.2020.9326906
17. Lin, R., Dong, S., Zhao, W., & Cheng, Y. (2023). Ultra-wide-band-based adaptive Monte Carlo localization for kidnap recovery of mobile robot. International Journal of Advanced Robotic Systems, 20(2), 172988062311639-172988062311639. https://doi.org/10.1177/17298806231163950
18. M. Adhipatiunus, R. R. A. Setiadji, Widyaputra, R. M., & Widyawardana Adiprawita. (2021). Implementation of informed rapidly exploring random tree * on a pre-mapped Octomap generated by real time appearance-based map. Nucleation and Atmospheric Aerosols. https://doi.org/10.1063/5.0060189
19. Mantha, B. R. K., & Garcia de Soto, B. (2022). Investigating the Fiducial Marker Network Characteristics for Autonomous Mobile Indoor Robot Navigation Using ROS and Gazebo. Journal of Construction Engineering and Management, 148(10). https://doi.org/10.1061/(asce)co.1943-7862.0002378
20. Marshal Revanth, C., Saravanakumar, D., Jegadeeshwaran, R., & Sakthivel, G. (2020). Simultaneous Localization and Mapping of Mobile Robot using GMapping Algorithm. 2020 IEEE International Symposium on Smart Electronic Systems (ISES) (Formerly INiS). https://doi.org/10.1109/ises50453.2020.00024
21. Nour Ayman Abujabal, Raouf Fareh, Sinan, S., Baziyad, M., & Maamar Bettayeb. (2023). A comprehensive review of the latest path planning developments for multi-robot formation systems. Robotica, 41(7), 2079–2104. https://doi.org/10.1017/s0263574723000322
22. Nwankwo, L., Fritze, C., Bartsch, K., & Rueckert, E. (2023). ROMR: A ROS-based open-source mobile robot. HardwareX, 14, e00426. https://doi.org/10.1016/j.ohx.2023.e00426
23. Oussama El Hamzaoui, & Steux, B. (2010). A fast scan matching for grid-based laser SLAM using streaming SIMD extensions. https://doi.org/10.1109/icarcv.2010.5707898
24. Palmani Duraisamy, & Natarajan, S. P. (2023). Multi-Sensor Fusion Based Off-Road Drivable Region Detection and Its ROS Implementation. https://doi.org/10.1109/wispnet57748.2023.10134440
25. Pedrosa, E., Pereira, A., & Lau, N. (2020). Fast Grid SLAM Based on Particle Filter with Scan Matching and Multithreading. https://doi.org/10.1109/icarsc49921.2020.9096191
26. Prasad, A. O., Mishra, P., Jain, U., Pandey, A., Sinha, A., Yadav, A. S., Kumar, R., Sharma, A., Kumar, G., Hazim Salem, K., Sharma, A., & Dixit, A. K. (2023). Design and development of software stack of an autonomous vehicle using robot operating system. Robotics and Autonomous Systems, 161, 104340. https://doi.org/10.1016/j.robot.2022.104340
27. Qu, P., Su, C., Wu, H., Xu, X., Gao, S., & Zhao, X. (2021). Mapping performance comparison of 2D SLAM algorithms based on different sensor combinations. Journal of Physics: Conference Series, 2024(1), 012056. https://doi.org/10.1088/1742-6596/2024/1/012056
28. Quang, H. D., Manh, T. N., Manh, C. N., Tien, D. P., Van, M. T., Tien, K. N., & Duc, D. N. (2020). An Approach to Design Navigation System for Omnidirectional Mobile Robot Based on ROS. International Journal of Mechanical Engineering and Robotics Research, 1502–1508. https://doi.org/10.18178/ijmerr.9.11.1502-1508
29. Rajesh Kannan Megalingam, Naick, V. S., Manaswini Motheram, Jahnavi Yannam, Nikhil Chowdary Gutlapalli, & Vinu Sivanantham. (2023). Robot operating system based autonomous navigation platform with human robot interaction. TELKOMNIKA Telecommunication Computing Electronics and Control, 21(3), 675–675. https://doi.org/10.12928/telkomnika.v21i3.24257
30. Salim Azak, & Erdogan, E. (2018). Performance evaluation of the grid-based FastSLAM in V-REP using MATLAB. https://doi.org/10.1109/tcset.2018.8336202
31. Satyam Raikwar, Yu, H., & Herlitzius, T. (2023). 2D LIDAR SLAM Localization System for a Mobile Robotic Platform in GPS Denied Environment. Journal of Biosystems Engineering, 48(2), 123–135. https://doi.org/10.1007/s42853-023-00176-y
32. Serhat Karaçam, & Tuğba Selcen Navruz. (2022). An improved adaptive FastSLAM algorithm with time-varying noise estimator. Asian Journal of Control, 25(4), 2617–2627. https://doi.org/10.1002/asjc.2968
33. Singh, A. V., Agrawal, Y., Gupta, R., Kumar, A., & Bohara, V. A. (2023, January 1). ANROLA: Autonomous Navigation with ROS and Laser Odometry. IEEE Xplore. https://doi.org/10.1109/COMSNETS56262.2023.10041368

34. Song, X. (2022). Research and design of robot obstacle avoidance strategy based on multi-sensor and fuzzy control. 2022 IEEE 2nd International Conference on Data Science and Computer Application (ICDSCA). https://doi.org/10.1109/icdsca56264.2022.9988357
35. Tripicchio, P., D'Avella, S., & Unetti, M. (2022). Efficient localization in warehouse logistics: a comparison of LMS approaches for 3D multilateration of passive UHF RFID tags. The International Journal of Advanced Manufacturing Technology, 120(7-8), 4977–4988. https://doi.org/10.1007/s00170-022-09018-1
36. Van, T., Ngô Mạnh Tiến, Nguyễn Mạnh Cường, Thi, H., & Nguyen Duc Duy. (2021). Constructing an Intelligent Navigation System for Autonomous Mobile Robot Based on Deep Reinforcement Learning. Springer EBooks, 251–261. https://doi.org/10.1007/978-3-030-76620-7_22
37. Volgina, A. D., D. Kirillov, Kravtsov, A. N., Dyakonova, S. S., & Kanev, A. (2023). The Robot-Guide for Indoor Navigation. https://doi.org/10.1109/reepe57272.2023.10086707
38. Wang, P., Chen, N., Fusheng Zha, Guo, W., & Li, M. (2018). Research on Adaptive Monte Carlo Location Algorithm Aided by Ultra-Wideband Array. https://doi.org/10.1109/wcica.2018.8630520
39. Xiang, Y., Li, D., Su, T., Zhou, Q., Brach, C., Mao, S. S., & Geimer, M. (2022). Where Am I? SLAM for Mobile Machines on a Smart Working Site. Vehicles, 4(2), 529–552. https://doi.org/10.3390/vehicles4020031
40. Yagfarov, R., Ivanou, M., & Afanasyev, I. (2018). Map Comparison of Lidar-based 2D SLAM Algorithms Using Precise Ground Truth. 2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV). https://doi.org/10.1109/icarcv.2018.8581131
41. Yan, H., Du, C., Liu, Y., Yang, X., Du, H., Han, P., Yu, Y., Xu, Y., & Sun, W. (2023). Automatic Parking System for Multi-vehicle in the Autonomous Driving Platform of the DeepRacer. Lecture Notes in Electrical Engineering, 537–553. https://doi.org/10.1007/978-981-99-1365-7_40
42. Zhang, G., & Liu Bo Liu. (2023). Implementation and optimization of ORB-SLAM2 algorithm based on ROS on mobile robots. https://doi.org/10.1117/12.2675118
43. Vega Torres, M. A., Braun, A., & Borrmann, A. (2022). Occupancy Grid Map to Pose Graph-based Map: Robust BIM-based 2D-LiDAR Localization for Lifelong Indoor Navigation in Changing and Dynamic Environments. In E. Hjelseth, S. F. Sujan, & R. Scherer (Eds.), eWork and eBusiness in Architecture, Engineering and Construction: ECPPM 2022 (pp. 1-8). CRC Press. https://doi.org/10.48550/arXiv.2308.05443.
44. Azzam, R., Taha, T., Huang, S., & Zweiri, Y. (2020). Feature-based visual simultaneous localization and mapping: a survey. SN Applied Sciences, 2(2). https://doi.org/10.1007/s42452-020-2001-3
45. Peng, G., Zheng, W., Lu, Z., Liao, J., Hu, L., Zhang, G., & He, D. (2018). An Improved AMCL Algorithm Based on Laser Scanning Match in a Complex and Unstructured Environment. Complexity, 2018, 1–11. https://doi.org/10.1155/2018/2327637
46. Wurm, K. M., Stachniss, C., & Grisetti, G. (2010). Bridging the gap between feature- and grid-based SLAM. Robotics and Autonomous Systems, 58(2), 140–148. https://doi.org/10.1016/j.robot.2009.09.009
47. Zou, H., Chen, C.-L., Li, M., Yang, J., Zhou, Y., Xie, L., & Spanos, C. J. (2020). Adversarial Learning Enabled Automatic WiFi Indoor Radio Map Construction and Adaptation with Mobile Robot. IEEE Internet of Things Journal, 1–1. https://doi.org/10.1109/jiot.2020.2979413
48. Durrant-Whyte, H., & Bailey, T. (2006). Simultaneous localization and mapping: Part I. IEEE Robotics & Automation Magazine, 13(2), 99-110.
49. Bailey, T., & Durrant-Whyte, H. (2006). Simultaneous localization and mapping (SLAM): Part II. IEEE Robotics & Automation Magazine, 13(3), 108-117.
50. Thrun, S., Burgard, W., & Fox, D. (2005). Probabilistic robotics. MIT press.
51. Leonard, J. J., & Durrant-Whyte, H. F. (1991). Simultaneous map building and localization for an autonomous mobile robot. IEEE/RSJ International Workshop on Intelligent Robots and Systems, 3(1), 1442-1447.
52. Cadena, C., Carlone, L., Carrillo, H., Latif, Y., Scaramuzza, D., Neira, J., Reid, I., & Leonard, J. J. (2016). Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. IEEE Transactions on Robotics, 32(6), 1309-1332.
53. Grisetti, G., Stachniss, C., & Burgard, W. (2007). Improved techniques for grid mapping with Rao-Blackwellized particle filters. IEEE Transactions on Robotics, 23(1), 34-46.
54. Kümmerle, R., Grisetti, G., Strasdat, H., Konolige, K., & Burgard, W. (2011). g2o: A general framework for graph optimization. IEEE International Conference on Robotics and Automation (ICRA), 3607-3613.
55. Davison, A. J., Reid, I. D., Molton, N. D., & Stasse, O. (2007). MonoSLAM: Real-time single camera SLAM. IEEE Transactions on Pattern Analysis and Machine Intelligence, 29(6), 1052-1067.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue I, January 2025

APPENDIX

In this appendix, you will find additional materials related to the research presented in this dissertation. These supplementary materials are hosted online for convenience.

1. **GitHub Repository**

- Link to GitHub Repository: [Briskvon1/mythesis_bot (github.com)]

2. **Data Visualization Code and Video Preview During Testing**

- Access to the code samples utilized in this research and view videos from testing by clicking on the following link: COMMAND VELOCITY DATA ANALYSIS (1).zip