Comparative Benchmarking of SLAM, Global Path Planning and Local Planning Capabilities of ROS and ROS2 Navigation Frameworks

Surabhi Gupta[1], Amit Barde[2], Siddhant Swan[3], Sapna Wagaj[4]
[1,2,3,4]Dept. Electronics and telecommunication engineering, Vishwakarma Institute of Technology, Pune, India

***Abstract*—This research paper comprehensively benchmarks the SLAM, global path planning, and local planning capabilities of the ROS Navigation Stack and ROS2 Navigation2 (Nav2) for autonomous mobile robot navigation. As robots operate in increasingly complex environments, evaluating navigation frameworks is critical. The study highlights Nav2's strengths in global planning and competitive local planning performance, offering insights for optimizing navigation solutions across applications. The study reveals Nav2's superior global planning performance with the SMAC planner and generally competitive local planning capabilities compared to ROS's Navigation Stack, albeit with an isolated instance of collision observed with the DWB planner.**

Keywords—ROS, ROS2, navigation, SLAM, path planning

I. INTRODUCTION

The field of autonomous mobile robotics has witnessed remarkable advancements in recent years, driven by the ever-increasing demand for autonomous systems capable of operating in diverse and complex environments. From industrial automation and logistics to exploration and search-and-rescue operations, the applications of autonomous mobile robots continue to expand, pushing the boundaries of what is possible. At the heart of these autonomous systems lies the critical challenge of enabling robust and efficient navigation capabilities. Navigation, in the context of mobile robotics, encompasses a multitude of intricate tasks, including simultaneous localization and mapping (SLAM), global path planning, and local path planning. SLAM algorithms are responsible for constructing a coherent representation of the environment while concurrently determining the robot's location within that map. Global path planning involves the computation of an optimal path from the robot's current position to a desired goal location, taking into account the static obstacles present in the mapped environment. Local path planning, on the other hand, focuses on generating feasible trajectories and velocity commands that allow the robot to navigate safely through its immediate surroundings, accounting for dynamic obstacles and kinematic constraints.

The seamless integration and coordination of these navigation tasks are essential for enabling truly autonomous operation of mobile robots. Failure in any one of these components can compromise the overall navigation performance, potentially leading to collisions, suboptimal paths, or even complete mission failure. Consequently, the development of robust and efficient navigation solutions has become a critical area of research and development within the robotics community.

The Robot Operating System (ROS) has emerged as a widely adopted middleware framework for developing robotic applications, providing a rich ecosystem of tools, libraries, and packages. Among its core components, the ROS Navigation Stack and the more recent Navigation2 (Nav2) Stack, part of the ROS2 framework, stand as comprehensive solutions for mobile robot navigation. These stacks offer a collection of algorithms and functionalities aimed at addressing the challenges associated with localization, mapping, and path planning. The ROS Navigation Stack, introduced in the earlier iterations of ROS, has been extensively utilized in various robotics projects and research endeavors. It offers a well-documented and robust framework for integrating SLAM algorithms, global path planners, and local trajectory planners, enabling autonomous navigation in diverse environments. However, as the field of robotics continues to evolve rapidly, the limitations of the Navigation Stack have become increasingly apparent, particularly in terms of flexibility, extensibility, and compatibility with emerging hardware and software platforms.

To address these limitations and unlock new possibilities in mobile robot navigation, the ROS2 community has developed the Navigation2 Stack, a modern and modular navigation solution designed to leverage the enhanced capabilities of the ROS2 framework. Built upon the foundations of its predecessor, Nav2 incorporates several architectural improvements, including a behavior tree-based approach, modular design, and support for multiple local trajectory and path planners within a single navigation task. These advancements aim to provide developers with greater flexibility and adaptability, enabling the tailoring of navigation strategies to specific application contexts and facilitating the integration of cutting-edge algorithms and techniques. As the adoption of ROS2 and Nav2 continues to grow, a comprehensive understanding of their capabilities, performance, and limitations becomes crucial for researchers, developers, and practitioners in the field of robotics. This research paper aims to conduct a comparative benchmarking study, evaluating the SLAM, global path planning, and local planning capabilities of both the ROS Navigation Stack and the ROS2 Navigation2 Stack. Through rigorous experimentation and analysis, this study seeks to provide valuable insights into the strengths and weaknesses of each framework, enabling informed decision-making and guiding future development efforts in the domain of autonomous mobile robot navigation.

II. LITERATURE REVIEW

This proposal [1] introduces Navigation2 as a modern navigation solution building on ROS Navigation's legacy. Utilizing behavior trees and dynamic environment methods, it operates on ROS2 for safety and lifecycle management. Its application alongside pedestrians showcases its relevance for comparative studies between navigation stacks in ROS and ROS2. The paper [2] introduces a real-time plane segmentation method for indoor navigation systems for the visually impaired, utilizing surface normal estimation in range images. Efficiency and accuracy challenges are addressed through integral images for normal estimation efficiency and dynamic smoothing region determination for enhanced accuracy. Compared to PCL methods, it offers superior performance across depths (1~8m) within a ROS framework at 30fps, showcasing its relevance for comparative studies between navigation stacks in ROS and ROS2. This paper [3] discusses an autonomous mobile robot's control system, combining STM32-based hardware circuitry and ROS for software control. The lower computer handles motor control and sensor data, while the upper computer utilizes ROS for SLAM and autonomous navigation. This setup's relevance to comparative studies lies in evaluating ROS and ROS2 navigation stacks' effectiveness and compatibility with hardware systems like STM32. The paper [4] explores global path planning for mobile robots using a hybrid A* algorithm and genetic algorithm approach, preceded by MAKLINK graph theory and the Dijkstra algorithm. Results show improved performance over Dijkstra in solution quality and computational time. This study's relevance to comparative studies lies in assessing how such path planning algorithms integrate with ROS and ROS2 navigation stacks for real-world robotic applications. The paper [5] focuses on real-time positioning and map building for mobile robots, employing an improved artificial potential field method for local path planning. It utilizes FastSLAM algorithm via particle filter for map construction and integrates autonomous positioning, path planning, obstacle avoidance, and motion simulation on ROS within the Gazebo platform. This study's relevance to comparative studies lies in evaluating how ROS and ROS2 navigation stacks accommodate advanced algorithms like FastSLAM and artificial potential fields for robust robotic navigation.

This paper [6] investigates ROS-based visual SLAM methods for mobile robot applications in indoor environments, comparing trajectories from various sensors processed by different SLAM algorithms. Methods include monocular ORB-SLAM, monocular DPPTAM, stereo ZedFu, and RTAB-Map, verified against LIDAR-based Hector SLAM and ground truth measurements. This study is pertinent to comparative studies between ROS and ROS2 navigation stacks, as it assesses the compatibility and performance of SLAM algorithms within each framework for real-world robotic navigation. This paper [7] introduces a

SLAM technique without odometer information, modifying the HECTOR SLAM method for different hardware and excluding IMU devices. It optimizes the method for Commercial Off-The-Shelf (COTS) hardware, enhancing its applicability. This study's relevance to comparative studies between ROS and ROS2 navigation stacks lies in assessing how such SLAM techniques integrate with each framework, considering their hardware requirements and optimization capabilities.

The paper [8] explores SLAM implementation using ROS and Arduino, detailing an inexpensive differential drive robot for mapping in domestic environments. It offers a theoretical explanation of the Rao-Blackwellization particle filter algorithm and provides a cost-effective approach with code and guidelines for 2D mapping. In a comparative study between ROS and ROS2 navigation stacks, this work highlights the adaptability of ROS for integrating with low-cost hardware like Arduino, offering insights into compatibility and performance across different platforms. This paper [9] introduces a novel SLAM framework combining ant system and LMB filter, facilitating joint estimation of feature locations and numbers. It employs a real-time moving ant estimator for vehicle trajectory estimation, outperforming PHD-SLAM and LMB-SLAM with better map quality and trajectory accuracy. In a comparative study between ROS and ROS2 navigation stacks, this work emphasizes the adaptability of ROS to innovative SLAM algorithms, suggesting potential improvements and integration strategies for future developments. The paper presents a ROS-based template for implementing the agent-based subsumption model in mobile robot control systems, leveraging ROS's reusable function units and messaging mechanism. It formalizes behaviors as ROS-based finite state machines and represents inhibitors and suppressors among layers as ROS nodes, facilitating easy instantiation. This work's relevance [10] to a comparative study between ROS and ROS2 navigation stacks lies in evaluating how ROS's messaging mechanism and software resources support the implementation of complex control systems, providing insights into potential improvements or adaptations for ROS2 integration. This paper [11] discusses combining ROS with MATLAB and Simulink to address the lack of graphical analysis and operation interfaces in ROS. MATLAB's powerful data processing and visualization capabilities, along with Simulink's visual interface, offer a convenient and efficient solution for analyzing and tracking moving objects in robot design. In comparative studies between ROS and ROS2 navigation stacks, this approach highlights the potential for integrating ROS with external tools like MATLAB for enhanced data processing and visualization, suggesting avenues for improvement or expansion in ROS2.

The paper [12] presents autonomous mobile robot implementations using ROS, focusing on safety and low power consumption. It utilizes 2D LiDAR and RGB-D camera with ROS 2D navigation stack, showcasing two setups: one on Raspberry Pi 3 with only 2D LiDAR and another on Intel NUC with 2D LiDAR and RGB-D camera. The experiments demonstrate dynamic obstacle avoidance capabilities, relevant for comparative studies between ROS and ROS2 navigation stacks to assess their performance and compatibility with different hardware setups.

This paper [13] details an autonomous mobile robot's control system using STM32 and ROS, covering hardware circuit design, lower computer tasks like motor control and sensor data transfer, and upper computer tasks including SLAM and autonomous navigation via ROS. This setup's relevance to a comparative study between ROS and ROS2 navigation stacks lies in assessing how each framework accommodates complex control systems, including SLAM and autonomous navigation, while leveraging different hardware platforms like STM32. This paper [14] presents a control system hardware based on ROS for cooperative robots operating in unknown areas, employing SLAM for localization and mapping. It introduces a new hybrid architecture utilizing one PC as master and Odroid-U3 embedded systems for communication, eliminating the need for expensive external sensors. This work's relevance to a comparative study between ROS and ROS2 navigation stacks lies in assessing how each framework supports such hybrid architectures and their compatibility with different hardware configurations for multi-robot control in complex environments.

The paper [15] investigates discrepancies between planned and traveled paths of a virtual differential drive robot in Gazebo-ROS simulator using ROS navigation stack. It conducts experiments navigating the robot through fixed destinations while recording and comparing planned and actual path coordinates, considering scenarios with dynamic obstacles. This study's relevance to a comparative study between ROS and ROS2 navigation stacks lies in evaluating how each stack handles path planning and execution in simulated environments, identifying strengths and weaknesses for real-world application.

III. METHODOLOGY/EXPERIMENTAL

A. *Components & Software Setup*

Within the context of the research project, the fundamental hardware infrastructure comprises a differential drive robot, encompassing essential components such as the chassis, motors, wheels, and encoders. This configuration enables precise control over locomotion and facilitates odometric measurements through the monitoring of wheel rotation. Augmenting this setup is a lidar sensor, serving as the primary means of environment perception and mapping. The LDROBOT D200 LiDAR sensor with a range of 8m was utilized. It has a scanning angle of 360° and a scanning frequency of 6 Hz. Facilitating the computational tasks is the Raspberry Pi 4 Model B, serving as the onboard computing platform. Equipped with a quad-core ARM Cortex-A72 CPU and available with 2GB, 4GB, or 8GB of RAM, the Raspberry Pi 4 offers substantial processing power for executing ROS 2 and ROS frameworks. Its array of USB ports enables seamless integration with sensors and peripherals, while GPIO pins facilitate connectivity with external hardware elements. To regulate the robot's motors, a motor controller such as the L298N Dual H-Bridge Motor Driver is employed. This component interfaces with the Raspberry Pi, converting digital commands into analog signals to govern motor speed and direction. Additionally, auxiliary hardware components including connectors, cables, breadboards, and mounting hardware are indispensable for the assembly and integration process.

In the ROS (Robot Operating System) setup, the core software framework used is ROS 1, which is widely adopted in the robotics community. ROS 1 provides a comprehensive ecosystem of libraries, tools, and packages for building robotic systems. For navigation tasks, the ROS Navigation Stack is utilized, which includes components such as the Trajectory Rollout planner and the Dynamic Window Approach (DWA) planner for generating feasible trajectories. Additionally, the stack incorporates the gmapping package for SLAM (Simultaneous Localization and Mapping) and the move_base package for high-level navigation planning. These components work together to enable autonomous navigation and mapping capabilities for the robot. On the other hand, in the ROS 2 setup, the focus shifts to the next-generation robotics framework, ROS 2. ROS 2 offers several advantages over ROS 1, including improved performance, better real-time capabilities, and enhanced security features. In the research project, the Humble framework is utilized as a part of the ROS 2 Navigation Stack. Humble is a lightweight navigation system designed specifically for ROS 2, offering efficient and scalable solutions for navigation tasks. It includes components such as the DWB (Dynamic Window Approach) planner, which computes optimal trajectories for the robot, and the AMCL (Adaptive Monte Carlo Localization) package for accurate localization. Additionally, the SLAM Toolbox is integrated into the ROS 2 setup for simultaneous localization and mapping.

B. *The ROS (1) NavStack*

The navigation stack constitutes a coherent amalgamation of packages within the framework of Robot Operating System (ROS) 1, dedicated to facilitating mobile robot navigation. Its operational framework involves the assimilation of data streams originating from odometry readings, sensory inputs, and prescribed goal poses. Subsequently, it generates and issues velocity commands conducive to ensuring the safe movement of the mobile robot's base. The navigation stack operates through the coordination of multiple packages, with the 'move_base' package serving as its central hub. This package encompasses a range of functionalities including state machines, planners, internal cost maps, and recovery behaviors essential for effective navigation.

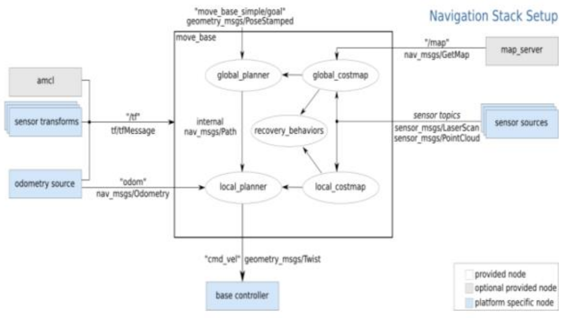

Fig 1. The NavStack Design

In the illustrated design, white nodes represent implemented mandatory components, blue nodes denote essential components requiring setup for each specific robot platform, and gray nodes indicate optional components that have already been integrated. The navigation system comprises several essential components crucial for its effective operation. Firstly, the Odometry Source component serves as a foundational element, requiring access to odometry information formatted as nav_msgs/Odometry. This information is indispensable for accurately determining the robot's pose and motion within its environment. Secondly, sensor sources play a vital role by providing laser scan or point cloud data related to the surrounding environment. This data is instrumental during both local and global path planning stages, particularly for obstacle avoidance, where it informs the generation of cost maps. The third component, Sensor Transforms, is tasked with ensuring proper coordination between various coordinate frames. This necessitates the publication of transforms by the robot, notably including the odom->base_link transform derived from odometry readings. Additionally, if a map is utilized, a map->odom transform becomes requisite for seamless integration. The fourth component, the Base Controller, interfaces with the move_base package to execute navigation commands. It requires a robot base controller capable of interpreting velocity commands formatted as geometry_msgs/Twist, thereby facilitating the robot's movement according to the navigation plan. Lastly, while optional, the AMCL (Adaptive Monte Carlo Localization) component becomes necessary if a map->odom transform is employed. Similarly, the Map Server component, also optional, provides the functionality of loading maps when required, further enhancing the system's adaptability and functionality.

The navigation system possesses the flexibility to initialize with or without a static map. In the absence of a static map, the system relies solely on information regarding encountered obstacles, forming optimistic global plans concerning unexplored areas. Subsequently, it adjusts its path planning strategy as additional obstacles are encountered, necessitating periodic replanning. Primarily, the move_base component assimilates pertinent data, generating both global and local path plans by considering corresponding global and local cost maps. It subsequently issues velocity commands to the robot's actuators, persisting until the navigation objective is achieved.

The navigation system comprises several fundamental components essential for its operational integrity. The first component is the Global Costmap, which functions as a comprehensive representation of the environment in a two-dimensional voxel grid format. It serves as the basis for global or full-length path planning, continuously integrating real-time obstacle data obtained from sensors to ensure accuracy and relevance. Complementing the Global Costmap is the Local Costmap, which focuses on the immediate surroundings of the robot. This map provides a detailed depiction of the local region visible to the robot and is dynamically updated with real-time obstacle information. Its primary role lies in facilitating local or short-distance path planning, offering a nuanced understanding of the immediate terrain. The Global Planner stands as a critical component, employing the A* algorithm to chart a path from the robot's current pose to its designated end goal pose. While serving as a high-level navigation guide, the Global Planner utilizes inputs such as the Global Costmap, robot localization data, and goal pose to generate an optimal trajectory. Conversely, the Local Planner operates on a shorter timescale, employing the Dynamic Window Approach (DWA) to plan short-distance paths. Its objective is to translate the high-level path guidance provided by the Global Planner into actionable velocity commands for the robot. To accomplish this, the Local Planner relies on inputs from the Local Costmap and guidance from the Global Planner. Lastly, the navigation system incorporates Recovery Behaviors, which activate in response to obstacles or potential failures encountered by the robot. These behaviors are designed to navigate the robot out of challenging situations, ensuring continued progress towards its objectives and maintaining operational resilience. In summary, the integration of odometry, sensor data, costmaps, and goal positions enables both the Global and Local Planners to generate velocity commands tailored to guide the robot effectively along its intended path, ensuring robust navigation capabilities.

Issues with the Navigation Stack in ROS 1 are discernible upon scrutiny. The move_base package is reliant on a monolithic and unconfigurable state machine, thereby limiting the flexibility available to developers seeking to tailor the system to meet specific application requirements. This constraint hampers the exploration of diverse developmental avenues, consequently impeding the realization of optimal solutions. Furthermore, the move_base functionality is inherently restricted to differential drive and holonomic wheeled robots. This limitation imposes constraints on the scope of applicability, rendering the navigation stack unsuitable for broader classes of robotic platforms. Additionally, the move_base package mandates the utilization of a singular set of global and local planning algorithms concurrently. This rigidity precludes the dynamic loading of alternative plugins for planning algorithms, thereby constraining the adaptability of the system to accommodate diverse operational contexts and custom applications.

C. *The ROS2 Nav2 Stack*

ROS 2 was conceived with the explicit objective of transcending the confines of laboratory environments, ushering the Robot Operating System into real-world applications. Navigation 2, building upon the foundation of ROS 2 and drawing inspiration from the achievements of its predecessor, the nav stack, aspires to extend the boundaries of navigational capabilities. By leveraging the enhanced reliability, security, and speed inherent in ROS 2, Navigation 2 endeavors to address the aforementioned limitations encountered within the nav stack. Departing from the unyielding nature of monolithic state machines, Navigation 2 adopts a Behavior Tree-based approach to navigation. This paradigm shift affords developers greater configurability and adaptability, fostering the exploration of diverse navigational strategies tailored to specific application contexts. Moreover, Navigation 2 adopts a modular architecture comprising independent servers responsible for planning, control, and recovery tasks. This modularity enables the seamless integration, removal, substitution, or augmentation of individual components, thereby enhancing the system's versatility and extensibility. In contrast to its predecessor, Navigation 2 introduces support for multiple local trajectory and path planners within the context of a single navigation task. This capability empowers developers to deploy a diverse array of planning algorithms, facilitating the optimization of navigation strategies in varying environments and scenarios.

Nav2 incorporates two key design patterns. The BT Navigator serves as the central component, housing the behavior tree essential for navigation behavior implementation. Task-Specific Asynchronous Servers, operating as ROS 2 nodes, host multiple algorithm plugins tailored to specific tasks, enhancing the framework's adaptability. Additionally, Nav2 employs Managed (Lifecycle) Nodes for deterministic behaviors and consolidates all relevant packages like AMCL and Map Server, ensuring a cohesive framework for efficient navigation system management. The Behavior Navigation Server assumes the role of the principal component and primary interface within the system, serving as the locus for navigation behavior implementation. Tasked with orchestrating the navigation process upon receipt of a goal pose from the user, it employs a Behavior Tree to govern the navigation task. To facilitate communication with the BT Navigator, ROS 2 action servers are employed, utilizing a NavigateToPose action message to initiate navigation requests. Subsequently, the Behavior Tree housed within the Behavior Navigation Server interfaces with additional action servers embedded within the controller, planner, behavior, and smoother servers. These servers collectively manage control efforts, compute plans, execute recovery maneuvers, and perform other pertinent tasks integral to the navigation process. The Behavior Tree nodes within the BT Navigator establish communication with modular servers, including the controller, planner, behavior, smoother, and potentially custom servers, to coordinate and facilitate robot navigation. Each action server is assigned a distinct .action type to enable interaction with the BT Navigator and other components of the navigation system.

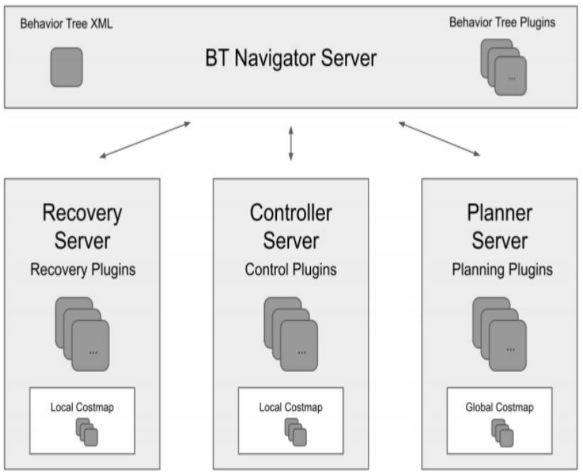

Fig 2. Nav2 Core Overview

These servers serve as hosts for a variety of algorithm plugins pertinent to their respective tasks, affording the Behavior Tree Node the flexibility to dynamically select an appropriate plugin at runtime. Moreover, they adhere to a standardized plugin interface, facilitating the creation and selection of new algorithms during runtime. All depicted servers within the design are realized as ROS 2 Managed (Lifecycle) Nodes. The Lifecycle Manager assumes responsibility for orchestrating the program lifecycle of both the BT Navigator and subsequent server Nodes. It systematically guides each server through the managed node lifecycle stages, namely: inactive, active, and finalized, ensuring orderly and controlled execution.

D. Mapping using both stacks

We have developed an expansive Gazebo world resembling a large factory site. The primary objective of this endeavor was to create an environment of significant scale, thereby presenting a formidable challenge for Simultaneous Localization and Mapping (SLAM) techniques. Autonomous Mobile Robots (AMRs) commonly operate within such expansive settings, particularly in factory floors and warehouses. These environments pose several inherent challenges for robotic systems. Mapping extensive spatial domains entails complexities beyond mere size considerations. Factors such as dynamic environmental conditions and potential symmetries within the arena further complicate the task of re-localization. Moreover, the collaborative and imperfect nature of human activities and other objects present within the environment introduce additional hurdles for global and local planners alike. Consequently, our designed Gazebo world serves as a testbed for evaluating the efficacy of navigation algorithms and strategies within large-scale and dynamic environments characteristic of real-world industrial settings. Through rigorous experimentation and analysis within this simulated environment, we aim to refine and enhance the navigation capabilities of autonomous robotic systems, thereby addressing the challenges inherent to deployment in complex industrial environments. We used a differential drive LiDAR-based mobile robot for the experiment.

1) GMapping (SLAM – ROS NavStack)

ROS employs the gmapping package, a particle filter-based Simultaneous Localization and Mapping (SLAM) solution tailored for mobile robotic platforms. Default parameters were utilized to evaluate the mapping performance within the context of our experimentation. The performance exhibited by gmapping within our simulated environment proved to be notably commendable. The generation of the map was achieved seamlessly, without necessitating any parameter tuning.

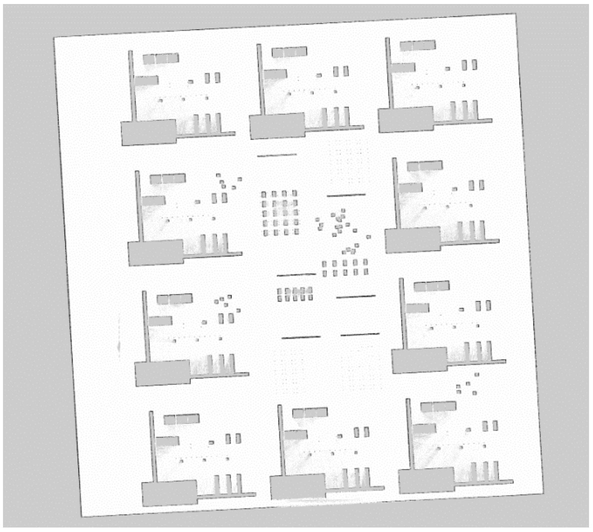

Fig 3. Map created via GMapping

2) SLAM ToolBox

The SLAM Toolbox offers a comprehensive array of features, encompassing functionalities such as relocalization, continued mapping, long-term mapping, and map merging. However, for the purpose of this comparison, our attention is specifically directed towards SLAM capabilities. The slam_toolbox package adopts a pose graph SLAM methodology, leveraging the karto scan matcher algorithm. Notably, this package is purported to exhibit particular efficacy within expansive indoor environments. Although the package demonstrates competence in mapping extensive spatial domains, the process of saving the map using the map_server ROS2 package encountered obstacles, as documented (refer to the issue here). While the SLAM Toolbox represents a notable stride towards establishing a dependable SLAM solution, its efficacy in real-world scenarios remains contingent upon accruing sufficient empirical experience.

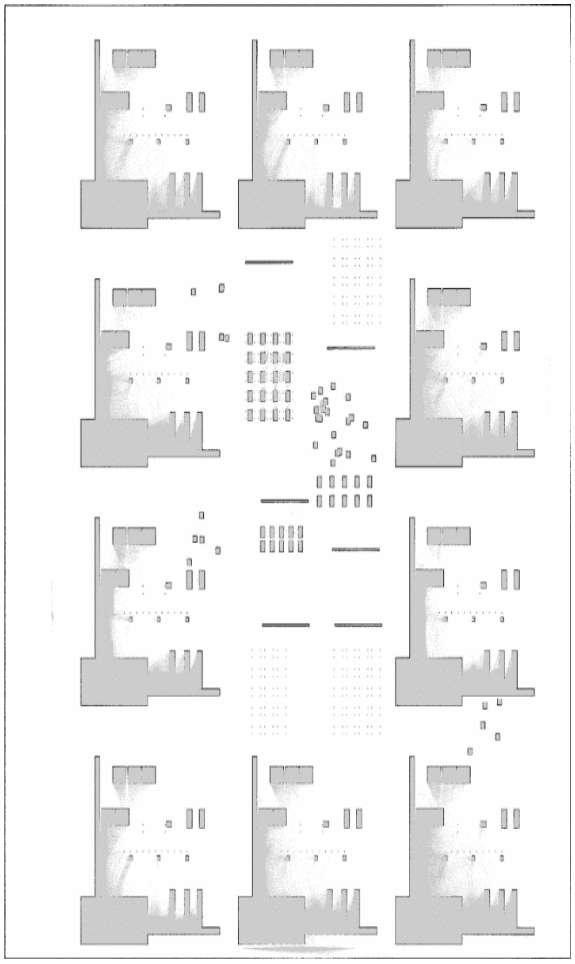
Fig 4. Map created by SLAM Toolbox

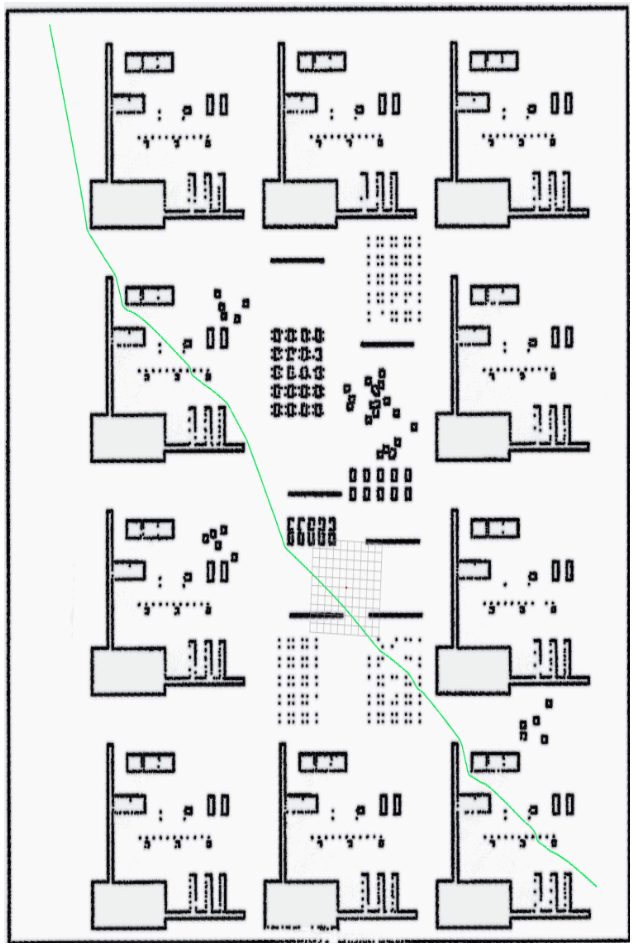
Fig 5. NavFN Global Plan

IV. RESULTS AND DISCUSSIONS

The global and the local planners for both the systems were evaluated.

1) NavFN (Global Planner – NavStack)

NavFn employs Dijkstra's planner to calculate the shortest path from the starting point to the designated goal. While the exploration of A* planning, incorporating heuristics, remains an alternative under consideration, NavFn offers limited flexibility in governing the nature of the global plan generated. Specifically, it prioritizes the generation of the shortest cost path, with the cost being determined by the associated costmap. However, NavFn does not account for factors such as path smoothness, the number of turns, or constraints such as the radius of curvature. As empirical evidence suggests, the pursuit of simplicity confers distinct advantages in certain contexts. The time taken by the NavFN global planner was 3.4 seconds.

2) SMAC Planner (Global Planner – Nav2 Stack)

Within the Nav2Stack, the SMAC planner stands as one of the two planning servers included, the other being NavFn. Leveraging the Hybrid State A* algorithm, the SMAC planner facilitates continuous state transitions within discrete navigation cells, offering the additional capability to select various motion models such as DUBIN or 2D Moore. Furthermore, numerous parameters concerning optimization, down-sampling, and cost multipliers contribute to the planner's adaptability, enabling the modification of its behavior to suit specific application requirements. The SMAC Planner took 1.5 seconds to generate the path plan.

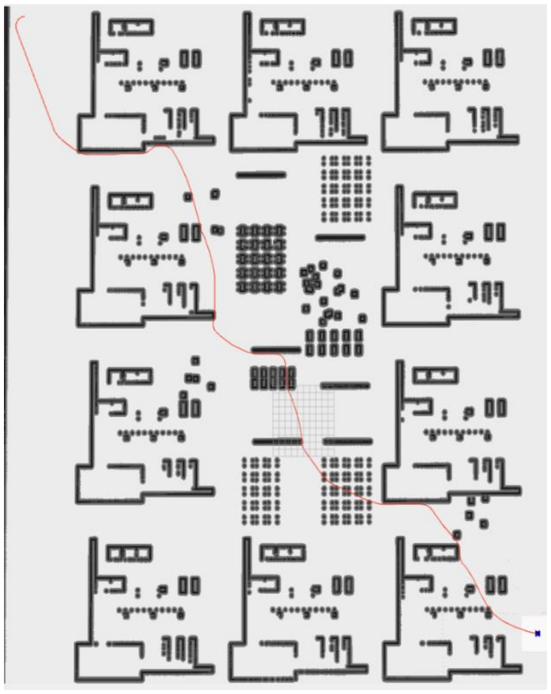
Fig 6. SMAC Global Plan

3) Trajectory Rollout Planner (Local Planner – NavStack)

The ROS NavStack incorporates the trajectory rollout planner and the Dynamic Window Approach (DWA) as integral components. Both planners operate on the principle of discretely sampling the robot's control inputs within user-specified constraints, followed by the evaluation of resulting trajectories based on relevant costs, culminating in the selection of the most optimal inputs. Conducting multiple simulations without parameter tuning for the trajectory rollout planner, albeit with the appropriate consideration of the robot's footprint, yielded results that were regrettably unsatisfactory. Drawing from previous experiences in hardware deployment, albeit not without imperfections, superior outcomes have been observed with the DWA and TEB local planners.

4) DWB Planner (Local Planner – Nav2 Stack)

Nav2 incorporates the DWB planner, representing an evolutionary iteration of the DWA planner. While the core algorithm remains largely unchanged, software enhancements aimed at bug fixes and code customization have been implemented, as indicated by available documentation. In an effort to assess the comparative performance of NavStack and Nav2, both systems were subjected to analogous, if not identical, test conditions. The objective was to evaluate their navigational efficacy in scenarios characterized by static yet undisclosed obstacles and congested dynamic environments. In the majority of instances, the DWB planner facilitated successful navigation, enabling the robot to navigate around obstacles. However, a singular occurrence was observed where the robot collided with an obstacle, representing an anomalous outcome.

V. FUTURE SCOPE

The ROS NavStack serves as a widely utilized platform among students, hobbyists, and various industrial sectors. Its popularity is attributed to the abundance of comprehensive documentation and robust community support, facilitating a seamless process for configuring robotic systems with NavStack. In contrast, our experience indicates that ROS2 documentation may be lacking, particularly in areas such as the setup of launch files in Python and the comprehension of novel concepts such as lifecycle management and Quality of Service (QoS) settings. These factors may contribute to a sense of overwhelm among users.

In terms of algorithmic advancements, it appears that significant bug fixes have been implemented, particularly within the realm of local planning. While this study did not explore several parameters associated with the SLAM toolbox, SMAC planner, and DWB planner, we intend to investigate these aspects in future research endeavors.

VI. CONCLUSION

This research conducted a comprehensive benchmarking study to evaluate and compare the simultaneous localization and mapping (SLAM), global path planning, and local planning capabilities of the ROS Navigation Stack and the ROS2 Navigation2 (Nav2) Stack for autonomous mobile robot navigation. An expansive Gazebo simulation environment resembling a large factory site was developed to rigorously test these navigation frameworks in a challenging, dynamic, and large-scale setting akin to real-world industrial deployments. The gmapping package in ROS and SLAM Toolbox in Nav2 were evaluated for SLAM performance, with gmapping exhibiting seamless map generation without parameter tuning. For global planning, Nav2's SMAC planner demonstrated superior performance compared to ROS's NavFN planner in terms of computational time. As for local planning, both the ROS Navigation Stack's trajectory rollout planner and Nav2's DWB planner facilitated successful obstacle avoidance in most instances, though an isolated collision occurred with the DWB planner. Overall, the study highlighted Nav2's strengths in global planning efficiency and competitive local planning capabilities, offering valuable insights to guide the selection and optimization of navigation solutions across diverse robotic applications.

VII. ACKNOWLEDGEMENT

We extend our sincere gratitude to Prof. Rupali Gavaraskar for her invaluable contributions to our project. Her expertise in writing and formatting the paper helped us write and present our research paper. We would also like to express our appreciation for her guidance in presentations, suggestions for improvements in design, in-depth research of the problem statements, and for providing us with resources for study.

REFERENCES

[1] S. Macenski, F. Martĭn, R. White and J. G. Clavero, "The Marathon 2: A Navigation System," 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 2020, pp. 2718-2725

[2] Ke Jin, Peilin Liu, Rongdi Sun, Zhenqi Wei and Zunquan Zhou, "Real-time plane segmentation in a ROS-based navigation system for the visually impaired," 2016 Fourth International Conference on Ubiquitous Positioning, Indoor Navigation and Location Based Services (UPINLBS), Shanghai, China, 2016, pp. 170-175

[3] L. Zhi and M. Xuesong, "Navigation and Control System of Mobile Robot Based on ROS," 2018 IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), Chongqing, China, 2018, pp. 368-372

[4] C. Zeng, Q. Zhang and X. Wei, "Robotic Global Path-Planning Based Modified Genetic Algorithm and A* Algorithm," 2011 Third International Conference on Measuring Technology and Mechatronics Automation, Shanghai, China, 2011, pp. 167-170

[5] Z. Liu, "Implementation of SLAM and path planning for mobile robots under ROS framework," 2021 6th International Conference on Intelligent Computing and Signal Processing (ICSP), Xi'an, China, 2021, pp. 1096-1100

[6] I. Z. Ibragimov and I. M. Afanasyev, "Comparison of ROS-based visual SLAM methods in homogeneous indoor environment," 2017 14th Workshop on Positioning, Navigation and Communications (WPNC), Bremen, Germany, 2017, pp. 1-6

[7] A. Spournias, T. Skandamis, E. Pappas, C. Antonopoulos and N. Voros, "Enchancing SLAM method for mapping and tracking using a low cost laser scanner," 2019 10th International Conference on Information, Intelligence, Systems and Applications (IISA), Patras, Greece, 2019, pp. 1-4

[8] A. L. Ibáñez, R. Qiu and D. Li, "An implementation of SLAM using ROS and Arduino," 2017 IEEE International Conference on Manipulation, Manufacturing and Measurement on the Nanoscale (3M-NANO), Shanghai, China, 2017, pp. 1-6

[9] M. Li, B. Xu, M. Lu, P. Zhu and J. Shi, "Ant system based LMB filter for SLAM implementation in ROS platform," 2017 International Conference on Control, Automation and Information Sciences (ICCAIS), Chiang Mai, Thailand, 2017, pp. 209-214

[10] M. Li, X. Yi, Y. Wang, Z. Cai and Y. Zhang, "Subsumption model implemented on ROS for mobile robots," 2016 Annual IEEE Systems Conference (SysCon), Orlando, FL, USA, 2016, pp. 1-6

[11] W. -J. Tang and Z. -T. Liu, "A convenient method for tracking color-based object in living video based on ROS and MATLAB/Simulink," 2017 2nd International Conference on Advanced Robotics and Mechatronics (ICARM), Hefei and Tai'an, China, 2017, pp. 724-727

[12] S. Gatesichapakorn, J. Takamatsu and M. Ruchanurucks, "ROS based Autonomous Mobile Robot Navigation using 2D LiDAR and RGB-D Camera," 2019 First International Symposium on Instrumentation, Control, Artificial Intelligence,

and Robotics (ICA-SYMP), Bangkok, Thailand, 2019, pp. 151-154

[13] L. Zhi and M. Xuesong, "Navigation and Control System of Mobile Robot Based on ROS," 2018 IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), Chongqing, China, 2018, pp. 368-372

[14] S. Park and G. Lee, "Mapping and localization of cooperative robots by ROS and SLAM in unknown working area," 2017 56th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE), Kanazawa, Japan, 2017, pp. 858-861

[15] R. K. Megalingam, A. Rajendraprasad and S. K. Manoharan, "Comparison of Planned Path and Travelled Path Using ROS Navigation Stack," 2020 International Conference for Emerging Technology (INCET), Belgaum, India, 2020, pp. 1-6