Exploring the Efficiency of ROS2 Path Planners in a Dynamic Maze Environment

Syed Muhammad Hussain
sh06892@st.habib.edu.pk
Habib University
Karachi, Pakistan

Syed Muhammad Daniyal
sz06880@st.habib.edu.pk
Habib University
Karachi, Pakistan

Ahmed Atif
ma06413@st.habib.edu.pk
Habib University
Karachi, Pakistan

Laiba Ahmed
la06855@st.habib.edu.pk
Habib University
Karachi, Pakistan

Dr.Qasim Pasta
qasim.pasta@sse.habib.edu.pk
Habib University
Karachi, Pakistan

Dr.Basit Memon
basit.memon@sse.habib.edu.pk
Habib University
Karachi, Pakistan

ABSTRACT

A critical task in the quickly developing field of robotics is efficient path planning in dynamic environments. Within the framework of ROS2 (Robot Operating System 2), this paper gives a thorough comparative analysis of several autonomous mobile robot path planning algorithms, including A*, Dijkstra, Regulated Pure Pursuit (RPP), and Dynamic Window Approach (DWA). This research investigated the effectiveness of merging local and global path planners in a dynamic maze environment using an experimental platform built on top of ROS2. Metrics include path length, deviation from the planned trajectory, and execution time. Obtained results demonstrate the performance of these algorithms highlighting their strengths and limitations while focusing on how some combinations perform better than others in terms of accuracy, planning time, and reliability.

KEYWORDS

Global and Local Path Planning, Autonomous Mobile Robot, ROS2, Comparative Analysis, A*, Dijkstra, Regulated Pure Pursuit, Dynamic-Window Approach

ACM Reference Format:
Syed Muhammad Hussain, Syed Muhammad Daniyal, Ahmed Atif, Laiba Ahmed, Dr.Qasim Pasta, and Dr.Basit Memon. 2024. Exploring the Efficiency of ROS2 Path Planners in a Dynamic Maze Environment. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

In the evolving landscape of automation, the involvement of robots in human-centered environments is critically increasing. These autonomous machines are not just improving the way tasks are carried out in sectors such as healthcare, warehouses, etc but also signifying a paradigm shift in how tasks are approached and executed. With the current advancements in mobile robots, autonomous path planning, and navigation in structured environments that contain dynamic obstacles remain a challenge.

For the smooth movement of mobile robots from one point to their destination - path planning involves two distinct yet complementary strategies: local and global path planning. The purpose of global path planners is to construct a route in a known environment that connects the beginning point with the destination. They use an extensive environmental map to plot a path that is usually the most efficient in terms of distance traveled or energy used. Global planners, like the Dijkstra and A* algorithms, are skilled at charting a path in a static environment with fixed barriers and a predetermined topology. Their incapacity to instantly adjust to unforeseen circumstances such as dynamic obstacles that weren't on the original map, however, is their drawback. On the other hand, Local Path Planners are meant to make judgments in real time and concentrate on the immediate environment. They react to unforeseen changes in the robot's immediate environment and dynamic impediments. Local planning strategies include things like the Dynamic Window Approach (DWA) algorithm. Based on sensor inputs, these planners continuously modify the robot's path to ensure safe navigation by averting collisions with objects in motion or unexpected obstacles.

In this context, this paper presents a comprehensive comparative analysis of the combination of Local and Global path planners i.e A*, Dijkstra, Regulated Pure Pursuit, and Dynamic-Window approach utilizing an experimental platform for autonomous mobile robots built on the ROS2 (Robot Operating System) These algorithms have been chosen based on their widely used nature and impact in the autonomous navigation domain. The comparative analysis provides a comprehensive review of the efficiency of the combination of these algorithms based on Path Length, Deviation from Planned Trajectory, and execution time.

The remainder of this paper is organized as follows. Section 2 provides a literature review on existing path planning techniques. Section 3 discusses the Key Concept of a state of the art path planning algorithms. Section 4 outlines the methodology and environment setup. Section 5 presents the results and discussion. Section 6 concludes the paper and Section 7 gives the potential future direction of this research.

Conference'17, July 2017, Washington, DC, USA
© 2024 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 LITERATURE REVIEW

The second generation of the Robot Operating System also known as ROS2, provides a stack of Global and Local path planning algorithms in its NAV2 framework, whose aim is to navigate the robot efficiently in different environments [14]. This paper aims to navigate the robot in a maze-like environment. To figure out which algorithm works best in this situation, the research will quantify the efficiency of Global and Local path-planning algorithms in ROS2 in the existence of static and dynamic obstacles. The literature review revolves around four distinct algorithms namely A-star and Dijktra's algorithm which are categorized into Global planners while DWA and RRT for Local planners.

Global path planners are streamlined algorithms that are responsible for establishing a detailed route for the robot from the current position to the goal. To explore the related work, Wu et al. in [1] explored a ROS-based logistics system with A-star as the Global path planning Algorithm for 20 robots to observe the multitasking path planning, efficiency, response time, and transportation speed. The results show that the algorithm works better than conventional methods, especially in managing how things are moved around.

Auh et al. in [2] utilized the A-star algorithm for efficient sequence planning in cluttered logistics container unloading, where they yield minimum system movement resulting in low cost and reduced distances. Since automation in logistics systems requires efficient use of path planning algorithms that is why most of the literature revolves around this domain. Moreover, Zhao et al. in [3] proposes an efficient Complete Coverage Path Planning (CCPP) scheme for ROS-based robots while utilizing a sub-area division and a Bidirectional A-star Algorithm for path planning which resulted in a 98% improvement in coverage ratio. Moving on to Dijkstra's algorithm, its efficiency can be well observed in the experiment conducted by Khadr et al. in [4]. He observed efficient path planning using Dijkstra in ROS simulation to design a small indoor robot for rescue and military operations tasks.

Similarly, Pereira et al. in [5] compared Dijkstra and A-star global planners in symmetric/asymmetric environments in Gazebo and with the real robot. Both algorithms guided the robot collision-free while differing mainly in processing power. In [6], Tüfekçi et al. evaluated NavfnROS planning algorithms which include a pack of several local and Global planners (Dijkstra and A-Star, DWA, etc) for mobile robot trajectory planning in dynamic maze environments. He observed the successful navigation of the Robot from point A to point B in all analyzed algorithms. These studies signify the importance of choosing efficient Global planners as per the dynamics of the environment.

Local Planners are responsible for aiding the robot in achieving the milestones while following the global path. To explore the related work, Dai et al. in [7] improved the conventional RRT algorithm's efficiency with a 5.15% shorter path and significant time savings (78.34% in planning, 21.67% in navigation) by incorporating the DWA algorithm in RRT. Similarly, Shen and Soh in [8] employed enhanced DWA to optimize omnidirectional robot navigation in confined spaces by employing a targeted sampling strategy. This modification to the DWA enhanced efficiency, ensuring successful navigation trials in space-constrained environments. In [9], Dien et al. present a ROS-powered mobile robot with effective obstacle avoidance using the Dynamic Window Approach, demonstrating precise navigation and target reaching in flat environments. Another experiment has been performed by Rajendran et al. in [10] where he introduced a RobMAP, an efficient hybrid path-planning approach for autonomous robots in unstructured environments. It utilizes the RRT algorithm for collision-free path generation in ROS. Experimental results demonstrated RobMAP's superior navigation efficiency, precision, and coverage compared to existing strategies, achieving 92.89% accuracy.

Wang in [11] has demonstrated a more in-depth analysis of the Algorithm's efficiencies. He tested four distinct Algorithms (A-star, Dijkstra, RRT, PRM) using the first generation of ROS. The results showed that PRM was good at planning accurately and reliably, Dijkstra was effective but might not always take the best path, A* was accurate but had some limitations, and RRT had trouble with unpredictable situations. This existing work signifies that choosing an efficient algorithm is necessary as they are like white canes for a robot that aids them in making rational and quick decisions efficiently at every step of their journey.

To address a research gap, the upcoming sections of this paper will quantitatively assess the efficiencies of ROS2 Navigation stack path planning algorithms. Specifically, the research aims to conduct a thorough comparative analysis of A-star, Dijkstra's, DWA, and RPP Algorithms within dynamic obstacle environments. Evaluation metrics, including path length, deviation of the robot from the planned trajectory, and execution time, will be employed to provide a comprehensive understanding of each algorithm's performance.

3 KEY CONCEPTS

This section explains several concepts that are essential to understand this paper. It discusses the A-star and Dijkstra Global Planners and DWA and RPP local planners in subsequent sections. By understanding these concepts, the methodology in section 4 can be understood easily.

3.1 Global Planners

(1) **A* Algorithm**
The A* method is a well-known pathfinding technique that is widely used in mobile robot navigation to discover the best route across a two-dimensional grid or map from a starting point to an objective. Combining the advantages of heuristic techniques and Dijkstra's algorithm, A* effectively explores the search space while taking the projected cost from the beginning to the end. It is beneficial for exploring complicated surroundings because it uses a heuristic function, usually the Manhattan or Euclidean distance, to direct the search towards the objective. The A* algorithm reduces the number of cells visited by using a heuristic function that indicates the direction to the goal cell[2]. A* algorithm manages the open set of nodes using a priority queue. This queue prioritizes nodes based on their F score - which is the sum of the actual path cost from the start node to the current node (G score) and the heuristic estimated cost from the current node to the goal (H score).

(2) **Dijkstra Algorithm**
A widely used technique in computer science and mobile

Algorithm 1 A* Pathfinding Algorithm

```
Require: Start node, End node, Graph
Ensure: Shortest path from Start node to End node
 1: Initialize open list with Start node
 2: Initialize closed list as empty
 3: while open list is not empty do
 4:     current_node ← node from open list with the lowest F score

 5:     Remove current_node from open list
 6:     Add current_node to closed list
 7:     if current_node is End node then
 8:         return Construct path from Start node to End node
 9:     end if
10:     for each neighbor of current_node do
11:         if neighbor is in closed list then
12:             Continue to next neighbor
13:         end if
14:         Calculate G, H, and F scores for neighbor
15:         if neighbor is not in open list then
16:             Add neighbor to open list
17:         end if
18:     end for
19: end while=0
```

Algorithm 2 Dijkstra's Shortest Path Algorithm

```
Require: Graph with vertices $V$ and edges $E$, Source vertex $src$
Ensure: Shortest path distances from $src$ to all other vertices in $V$
 1: Initialize priority queue $Q$
 2: for all vertex $v \in V$ do
 3:     $dist[v] \leftarrow \infty$
 4:     Add $v$ to $Q$
 5: end for
 6: $dist[src] \leftarrow 0$
 7: while $Q$ is not empty do
 8:     $u \leftarrow$ vertex in $Q$ with minimum $dist[u]$
 9:     Remove $u$ from $Q$
10:     for all neighbor $v$ of $u$ do
11:         $alt \leftarrow dist[u] + \text{weight}(u, v)$
12:         if $alt < dist[v]$ then
13:             $dist[v] \leftarrow alt$
14:         end if
15:     end for
16: end while
17:
18: return $dist[\,]$ =0
```

robotics for figuring out the shortest path between nodes in a grid network is called Dijkstra's algorithm, after the Dutch computer scientist Edsger Dijkstra. The goal of the technique is to discover the most efficient path from a defined source node to every other node in the network, which works on graphs having weighted edges that reflect costs or distances. Initially, it establishes approximate distances, utilizes a priority queue to investigate nodes progressively more apart, and keeps adjusting distance estimations till the best routes are found. Because it can find the shortest pathways in a range of situations, Dijkstra's method is essential in applications like network routing and optimization. While Dijkstra's method can be effective in a complicated environment, it is less effective when the path to the destination cell is straightforward, such as a straight line.[1]

3.2 Local Planners

(1) **Regulated Pure Pursuit Algorithm (RPP)**
Regulated Pure Pursuit is a mobile robot navigation algorithm that improves performance by adding further regulatory mechanisms to the Pure Pursuit method's simplicity. In Pure Pursuit, the robot is guided along a predetermined course, usually based on a lookahead distance, towards a target location. Refinements are introduced in Regulated Pure Pursuit to alleviate oscillations and overshooting. To provide smoother control, it combines lookahead adaptation with speed regulation. This method modifies the lookahead distance dependent on the robot's velocity. Regulated Pure Pursuit, which combines regulatory features with Pure Pursuit, is ideally suited for mobile robot navigation applications [15]. It alleviates some of the drawbacks of standard Pure Pursuit while offering a balance between simplicity and adaptability in a variety of environments.

(2) **Dynamic Window Approach (DWA)**
In mobile robotics, the Dynamic Window Approach (DWA) is a local navigation method that allows for real-time path planning and obstacle avoidance. Using its kinematic restrictions and sensor data, this technique dynamically computes a "window" of possible velocities and rotational speeds for the robot. The program generates a collection of possible routes by taking into account the robot's dynamic capabilities, its present condition, and the world around it. Subsequently, DWA assesses these paths by employing a cost function that considers the distance to barriers, objective alignment, and motion smoothness [14]. The robot is guided through its immediate surroundings by the necessary control instructions once the trajectory with the lowest cost within the dynamically possible window is selected. The algorithm is as follows:
Step 1: Pruning of Search space:

The search space of the possible velocities is reduced in three steps:

(a) Circular trajectories: The dynamic window approach considers only circular trajectories (curvatures) uniquely determined by pairs (v, ω) of translational and rotational velocities. This results in a two-dimensional velocity search space.
(b) Admissible Velocites: The restriction to admissible velocities ensures that only safe trajectories are considered. A pair (v, ω) is considered admissible if the robot can stop

before it reaches the closest obstacle on the corresponding curvature.

(c) Dynamic window: The dynamic window restricts the admissible velocities to those that can be reached within a short time interval given the limited accelerations of the robot.

Step 2: Optimize (maximally) the objective function:

$$G(v,\omega) = \sigma(\alpha \cdot heading(v,\omega) + \beta \cdot dist(v,\omega) + \gamma \cdot vel(v,\gamma))$$

where $heading(v,\omega)$ is the measure of progress towards the goal. The $dist(v,\omega)$ is the distance to the nearest obstacle. And $vel(v,\omega)$ is the linear forward velocity of the turtlebot.

4 METHODOLOGY

In this methodology section, we detail the key components that form the foundation of our research on path-planning algorithms within dynamic environments. Our approach is centered around the utilization of ROS2 Navigation Stack, Gazebo simulator, Turtle Bot3, and carefully designed maze environments with dynamic obstacles. Each element contributes uniquely to the study's objective, ensuring a comprehensive evaluation of the adaptability and efficiency of NAV2 path planners.

4.1 ROS2

This research employs Robot Operating System 2 (ROS2) as a foundational framework, capitalizing on its flexible and modular design to enable seamless communication among robotic components. A specific version of ROS2 Humble is chosen for consistency, accompanied by necessary packages and dependencies. ROS2 serves as the core platform for developing and implementing path-planning algorithms using its Navigation stack (NAV2), ensuring a standardized and extensible environment. The system architecture is composed of these NAV2 integral components that collectively form the foundation for the robotic navigation system that is depicted in figure 1.

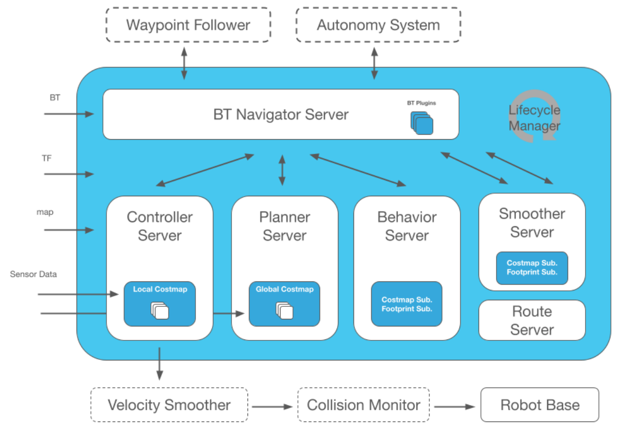

Figure 1: ROS2 Navigation Stack [14]

4.2 Environment Setup

In configuring the environment for this research, careful consideration has been given to key components that form the foundation for near-realistic experimentation and a comprehensive evaluation of ROS2 path-planning algorithms. The following sections provide detailed information about the setup of the simulation platform, robotic platform, maze environment, dynamic obstacles, and mapping procedures.

4.2.1 Gazebo. Serving as the simulation platform, Gazebo provides a dynamic and interactive 3D environment for realistic experimentation. Specific parameters, physics settings, and Gazebo configuration are meticulously specified to create controlled conditions for evaluating path-planning algorithms. The chosen Gazebo setup is pivotal in ensuring the reproducibility and validity of experimental results.

4.2.2 Turtle Bot 3. The Turtle Bot3 Waffle edition is chosen as the robotic platform for simulation due to its Lidar sensor, actuators, and kinematic properties. Widely used in the ROS ecosystem, this choice ensures an accurate representation of real-world scenarios, adding practical relevance to the study's outcomes.

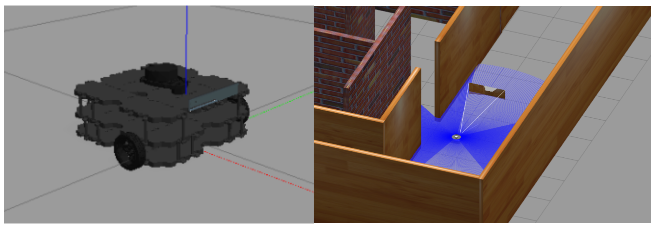

Figure 2: Turtle Bot3 Waffle

4.2.3 Maze. Designed to challenge NAV2 path planners with dynamic obstacles, the maze environment's specifics, including layout, dimensions, and obstacle characteristics, are meticulously defined. This intentional design introduces dynamic elements, allowing for the evaluation of ROS2 path planning algorithms' adaptability to changing environmental conditions as shown in figure 3.

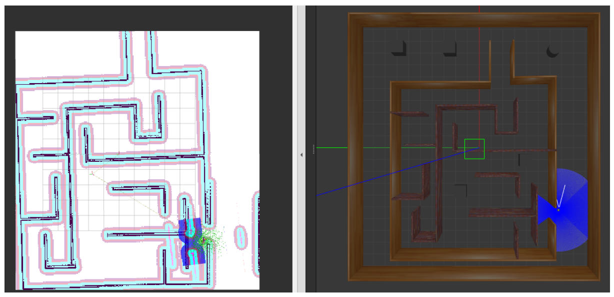

Figure 3: Top View of the Maze

4.2.4 Dynamic Obstacles. Four cubes are strategically placed within the maze, moving randomly with fixed velocity and kinematics constraints. Cube placement is designed to create a 50% chance of encountering a moving obstacle, with cube velocity set below the turtlebot's minimum for safe navigation.

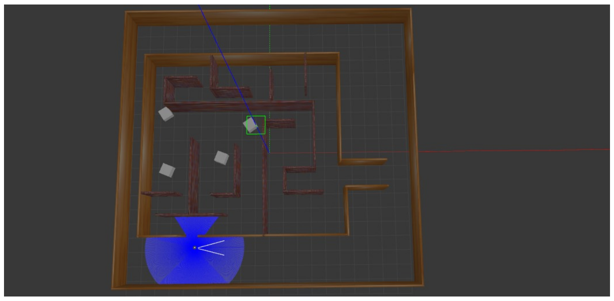

Figure 4: Dynamic Obstacles

4.2.5 Mapping. For generating a Lidar-based SLAM of the environment, Nav2's Slam-Toolbox is employed. This mapping process enhances the study's ability to capture and analyze environmental features, contributing to a more comprehensive evaluation of the entire navigation system.

In the upcoming section, we will analyze prominent path planning algorithms from the NAV2 stack and present the results based on defined metrics.

5 RESULTS AND ANALYSIS

In this section, a thorough analysis is conducted to gauge the performance of the planners. The evaluation is based on three key performance metrics:

1. **Path Length**
2. **Deviation of Robot from Planned Trajectory**
3. **Execution Time**

These metrics are employed to assess the performance of global planners, local planners, and the combined influence of both, respectively. The subsequent subsections provide detailed definitions and considerations for each of these performance metrics.

5.1 Path length

The path length simply calculates the length magnitude of the non-linear path taken by the robot and of the path generated by the global planner. The path length is calculated using the accumulative Euclidean distance formula:

$$dist_t = dist_{t-1} + \sqrt{(x_t - x_{t-1})^2 + (y_t - y_{t-1})^2}$$

5.2 Deviation of robot from planned trajectory

The calculates the mean difference of distance between robot trajectory and planned path. The equation is as follows:

$$dev = \frac{1}{T}\sum_{t}^{T}\sqrt{(x_{robot}(t) - x_{plan}(t))^2 + (y_{robot}(t) - y_{t-1}(t))^2}$$

5.3 Execution Time

The calculates the mean difference of distance between robot trajectory and planned path. The equation is as follows:

$$time\_taken = T$$

The analysis methodology involves conducting four repeated experiments, each representing a specific combination of a global planner and a local planner. Each experiment was repeated 10 times to achieve a standardized result. The objective is to generate analytical data, including plots, for all these planner combinations. Subsequently, a comparative analysis is performed to deduce the most effective combination of planners and to identify the scenarios in which they perform optimally. The presented plots depict both the path taken by the robot odometry (in blue) and the path generated by the global planner (in red). Note that these plots are from one of the iteration in each experiment.

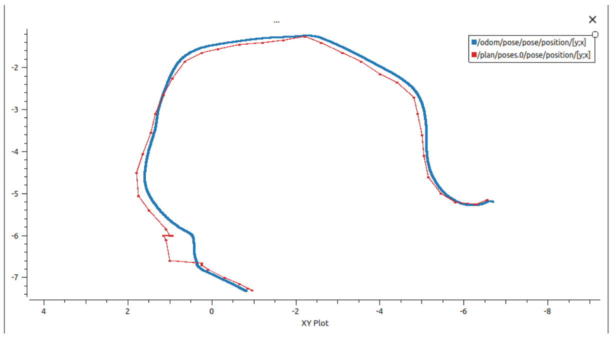

Figure 5: Dijkstra and RPP

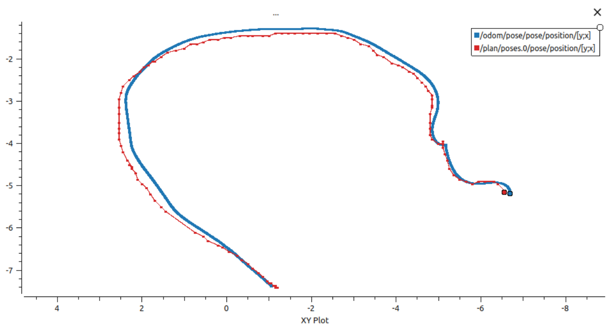

Figure 6: Dijkstra and DWB

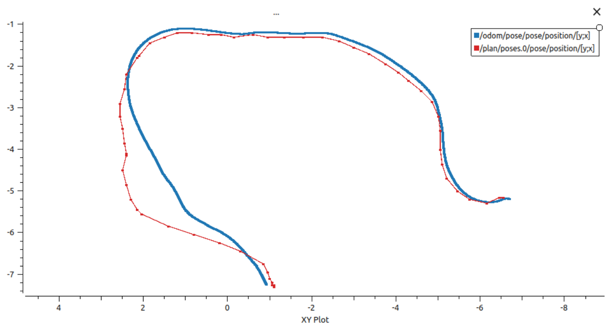

Figure 7: A* and RPP

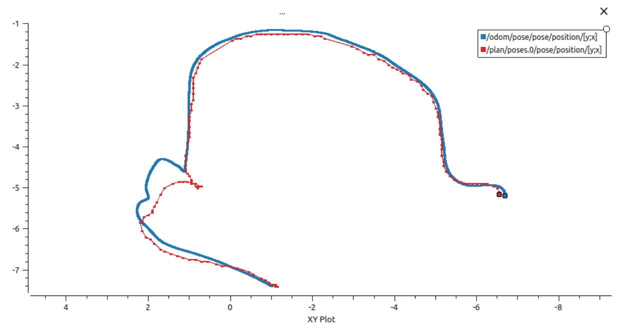

Figure 8: A* and DWB

In all the plot, what we can observe that on average most deviation happens when the turtle bot encounters a dynamic obstacle in front of it. There is a case like figure 6 where no significant deviation from the path is seen at all. Whereas, in Figure 8 we see the opposite version of figure 6 where the turtle bot spends a significant time being deviated from the plan.

Table 1: Average Performances of all 4 experiments

Planner	Distance	Deviation	Time
Dijkstra/RPP	25.354	0.217	153
Dijkstra/DWB	26.797	0.165	109
A*/RPP	26.780	0.270	62
A*/DWB	27.349	0.222	140

6 CONCLUSIONS

From the table 1, we can observe that on average Dijkstra takes the shortest path, while A* takes the longest path. We can also observe that the DWB performs the least deviation from the plan. Generally, the combination of A* and RPP has the shortest execution time, while Dijkstra and RPP have the highest execution time. Implying the RPP does not perform well with Dijkstra.

In conclusion, there is no one holistic setting in which a particular global and local planner combination is the best. The comparative analysis discussed in this paper rather rules out specific cases where one is better than the other. The purpose of the paper is to let the readers in academia and practice know to use which combination of planners depending on what their objective performance goal is.

7 FUTURE WORKS

In future work, we plan to deploy and evaluate these algorithms on the physical hardware robot currently in development. Building the whole system with Nvidia Jetson Nano as shown in figure 9. This initiative aims to bridge the gap between simulation and reality, providing valuable insights into how these algorithms perform in real-world scenarios.

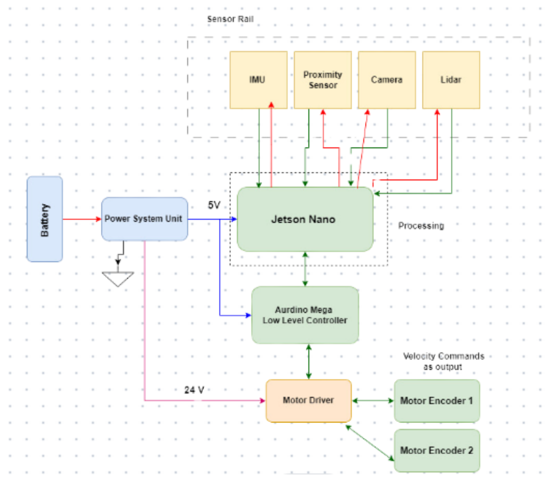

Figure 9: System Level Diagram

REFERENCES

[1] Ben-Ari, M., Mondada, F. (2018). Mapping-Based Navigation. In: Elements of Robotics. Springer, Cham. https://doi.org/10.1007/978-3-319-62533-1_10

[2] Xiang Liu and Daoxiong Gong, "A comparative study of A-star algorithms for search and rescue in perfect maze," 2011 International Conference on Electric Information and Control Engineering, Wuhan, 2011, pp. 24-27, doi: 10.1109/ICE-ICE.2011.5777723.

[3] Wu, R. (2023). Optimization Path and Design of Intelligent Logistics Management System Based on ROS Robot. Journal of Robotics, 2023.

[4] Auh, E., Kim, J., Joo, Y., Park, J., Lee, G., Oh, I., ... & Moon, H. (2024). Unloading sequence planning for autonomous robotic container-unloading system using A-star search algorithm. Engineering Science and Technology, an International Journal, 50, 101610.

[5] Zhao, S., & Hwang, S. H. (2023). Complete coverage path planning scheme for autonomous navigation ROS-based robots. ICT Express.

[6] Khadr Sr, M. S., Beaber Sr, S. I., Said, E., Elmayyah, W. M., & Hussein, W. M. (2021, April). Indoor path-planning for a tracked mobile robot using Dijkstra's algorithm and Ros. In Unmanned Systems Technology XXIII (Vol. 11758, pp. 211-220). SPIE.

[7] Ugalde Pereira, F., Medeiros de Assis Brasil, P., de Souza Leite Cuadros, M.A., Cukla, A.R., Drews Junior, P. and Tello Gamarra, D.F. 2021. Analysis of Local Trajectory Planners for Mobile Robot with Robot Operating System. IEEE Latin America Transactions. 20, 1 (Aug. 2021), 92–99.

[8] Tüfekçi, Z., & Erdemir, G. (2023, June). Experimental Comparison of Global Planners for Trajectory Planning of Mobile Robots in an Unknown Environment with Dynamic Obstacles. In 2023 5th International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA) (pp. 1-6). IEEE.

[9] Dai, J., Li, D., Zhao, J., & Li, Y. (2022). Autonomous navigation of robots based on the improved informed-RRT algorithm and DWA. Journal of Robotics, 2022.

[10] Shen, C., & Soh, G. S. (2023, August). Targeted Sampling DWA: A Path-Aware DWA Sampling Strategy for Omni-Directional Robots. In International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (Vol. 87363, p. V008T08A061). American Society of Mechanical Engineers.

[11] Dien, N. D., Van Hoa, R., Van Minh, P., Van Huy, P., Van Anh, L., & Duc, T. (2022). Research, Design and Control Mobile Robots for Intelligent Navigation Based on ROS Programming.

[12] Rajendran, G., Uma, V., & O'Brien, B. (2022). Unified robot task and motion planning with extended planner using ROS simulator. Journal of King Saud University-Computer and Information Sciences, 34(9), 7468-7481.

[13] Wang, S. (2024, January). Comparative research on path planning algorithms for autonomous mobile robots based on ROS. In International Conference on Algorithm, Imaging Processing, and Machine Vision (AIPMV 2023) (Vol. 12969, pp. 219-224). SPIE.

[14] Steven Macenski, Francisco Martin, Ruffin White, Jonatan Ginés Clavero, *The Marathon 2: A Navigation System*, in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[15] S. Macenski, T. Moore, DV Lu, A. Merzlyakov, M. Ferguson, *From the desks of ROS maintainers: A survey of modern & capable mobile robotics algorithms in the robot operating system 2*, *Robotics and Autonomous Systems*, 2023.