Path Planning Comparison Strategies for Mobile RobotNavigation

Anderson F. de S. Lima[1*], Marcella G.S. dos Santos[1], João V.S. Mendes[1], Matheus A. da Silva[1], Marco A. dos Reis[2]
[1]Robotics and Autonomous Systems Competence Center, SENAI CIMATEC University Center; [2]Computational Modeling and Industrial Technology Program, SENAI CIMATEC University Center;Salvador, Bahia, Brazil

Autonomous navigation is an essential application because it allows the robot to perform activities without human interference. It enables the execution of tasks that pose a risk or difficulty to the human being. This material aims to present the research to evaluate the performance of different navigation algorithms. The results obtained in the first phase of the research will be highlighted, in which the A* (A Star) and Dijkstra techniques were evaluated. The robot was integrated into the Robot Operating System (ROS) framework, and the navigations were performed in a labyrinth-like environment.
Keywords: Autonomous Navigation. ROS. A*. Dijkstra.

Introduction

Mobile ground robots are increasingly common in robotics in hazardous environments. They are classified as Unmanned Ground Vehicles (UGVs), used in many areas, and may have applications in disaster rescue, nuclear inspection, planetary exploration, and military combats [1]. These robots navigate autonomously, requiring intelligence to define the most efficient routes to complete their missions. In a robotic navigation platform, there is a layer responsible for obtaining information from the environment and another layer responsible for reading and interpreting this data. Path planning algorithms are used to find a path between one point and another as efficiently as possible, such as Dijkstra and A* [2].

Robotic System

Autonomous robots are systems capable of interacting with the work environment without human action throughout their task execution. One of the most common applications in mobile robotics is autonomous navigation through unfamiliar environments, where many navigation techniques can be used to achieve an end goal.

The robotic system used in this work was the Turtlebot, a low-cost, open-source mobile robotics platform designed to be user-friendly and with the same capabilities as platforms from large robotics companies [3]. Turtlebot (Figure 1) has several devices in its structure to move, which allows the robot to recognize the environment around the asset, process this information and control its actions.

A 2D LIDAR is responsible for collecting information from the environment, using a laser beam fired by the device so that the system can identify close objects based on the reflection time [4]. A Raspberry is responsible for processing this data, which is presented as a low-cost, portable solution for integrating the sensor into the robotic system [5]. Hence, an OpenCR control board was used to control the robot actuators and distribute power to the devices [4].

Robotic systems may also be able to explore unknown spaces. For example, explorations can be dedicated to getting a robot out of a residence, a maze, getting a map of an unknown region, along with others.

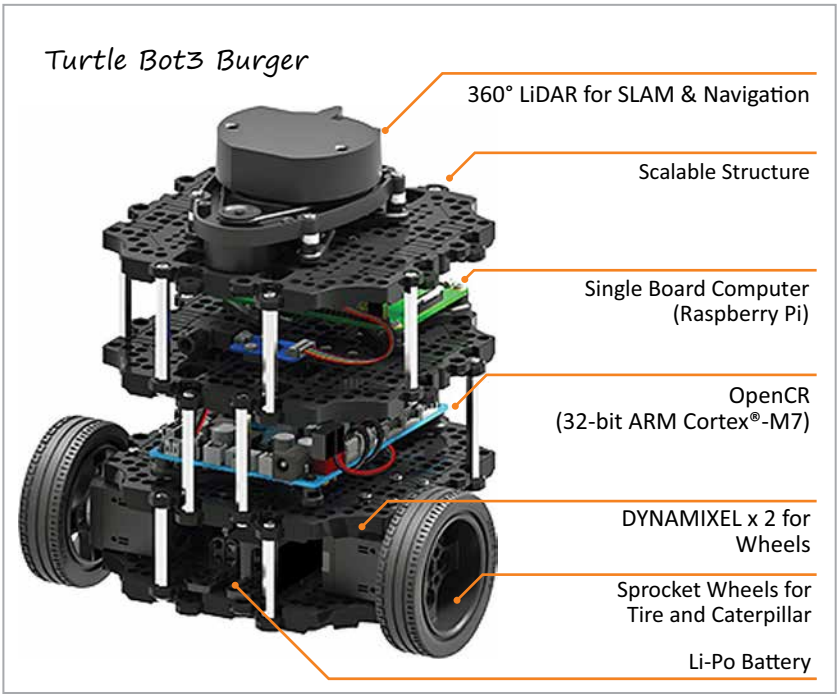
Robotics Navigation

Navigation is essential for mobile autonomous systems because mobile robots need to move in environments with little or no human intervention

Received on 12 September 2022; revised 21 February 2023.
Address for correspondence: Anderson F. de S. Lima. Anderson F. de S. Lima. Rua Senador Quintino - 1984 - Brasília. Feira de Santana, Bahia, Brazil. Zipcode: 44088-720. E-mail:eng.andersonfsl@gmail.com.

J Bioeng. Tech. Health 2023;6(1):52-57

Figure 1. TurtleBot3 - Burger Version.



in many applications. A framework is needed that supports the navigation. ROS2 contains several packages that are used in robotic applications. Among them, NAV2 seeks to find a safe way to move a robot from point A to point B. It runs in several robot navigation applications, such as the following dynamic points. It completes dynamic path planning, calculate motor speeds, avoid obstacles, and structure recovery behaviors. The package uses behavior trees to call modular servers, so it completes an action, which can be, calculating a path, controlling effort, recovery, or any other activity related to navigation [6].

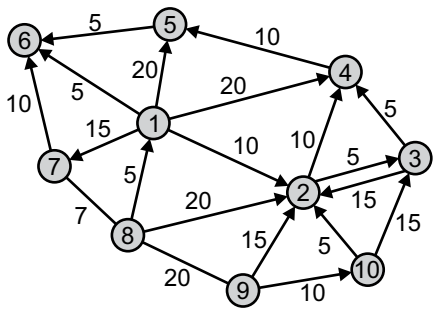
Some algorithms are used in navigation to do trajectory planning to perform the trajectory and achieve the proposed goal at a lower cost. The minimum path discovery algorithms can be classified into two forms: Uninformed search, when the algorithm does not use heuristics to find the shortest path between origin and destination, and informed search when the algorithm uses heuristics to estimate the minimum cost path [7]. Some examples of these algorithms are Dijkstra and A*, which use uninformed and informed search, respectively.

Dijkstra

Dijkstra's algorithm is a technique that is often used in differential mobile robots because it uses uniformed search capable of obtaining a trajectory between two nodes. These two nodes are points in the environment. From a specific node in space, the Dijkstra algorithm calculates from all available nodes a trajectory to the other node where the goal is. Figure 2 illustrates the possibilities of the paths to be used by the Dijkstra algorithm.

Dijkstra solves the single-origin shortest path problem on a directed or undirected graph when all edge weights are non negative. Equation 1

Figure 2. Algorithm Dijkstra.



illustrates the time cost of Dijkstra's algorithm, in which V is the number of vertices, and E is the number of edges [8].

$$O\left[E + \frac{V}{Log(V)}\right] \quad (1)$$

*A**

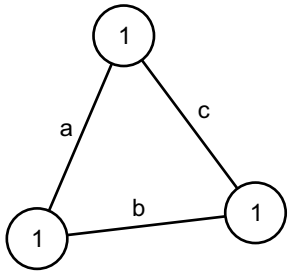
The A* search algorithm is used in various fields of computer science and can also be applied to search problems related to mobile robotics.

The A* was created as part of a general-purpose mobile robot project called Shakey [9]. One of Shakey's most notable results was using this search algorithm. The A* algorithm uses graphs as the basis of the search system (Figure 3). The initial vertex represents the starting point of the search, and the endpoint is the final goal. The algorithm is formulated using weighted graphs to find a path to the given objective with the lowest cost (shortest distance traveled, along with others). The algorithm keeps several paths originating from the starting point and expands these paths one point at a time until its search criteria are satisfied.

The search is performed through minimal paths using heuristic functions, i.e., the selection of nodes is based on the distance from the start node plus the approximate distance to the destination. This approximation estimate can be represented by the function f(n) = g(n) + h(n) [10].

According to Rachmawati and Gustin [11], the star algorithm and the most widely known form of best-choice search solution, A star evaluates nodes in graphs by combining the cost of reaching a particular node already visited and the cost of going to the destination node.

Figure 3. Graphs of A*' algorithm.

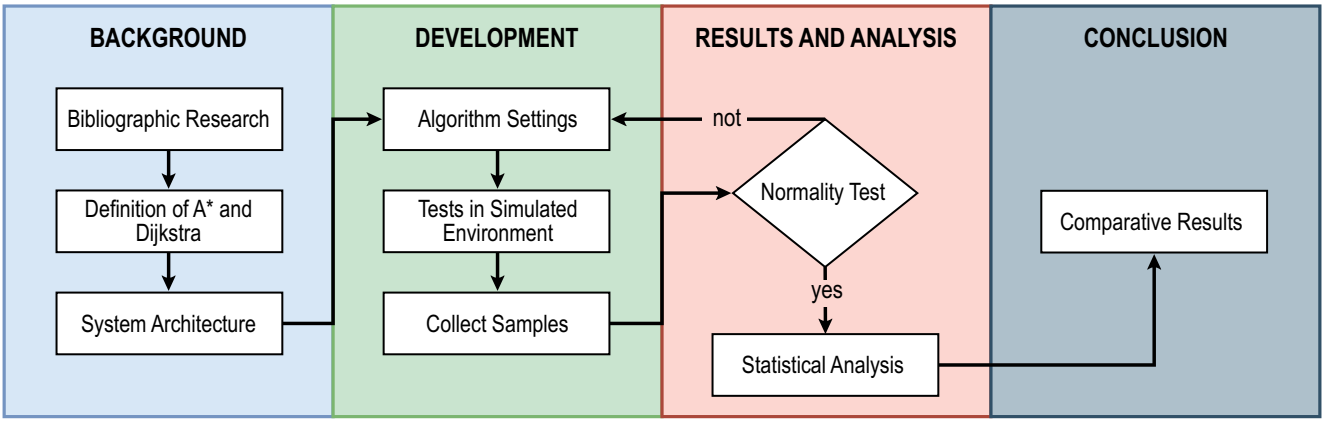


Materials and Methods

This research aims to compare the performance of trajectory planning strategies in ROS using Turtlebot3, and the scope of the paper is to present the results of the comparison made between the A* and Dijkstra techniques. Figure 4 presents the Methods used to perform the comparison between planning strategies.

In the first stage, a literature search was done to understand the concepts of the A* and Dijkstra techniques, how to use them, and thus define the system architecture. After that, the algorithms were configured on the robotic platform to be tested in a simulation environment, and the first samples of the system were collected. After collecting the samples, a normality test is done to know if the sample is

Figure 4. Methods used to compare strategies' plans.



viable for analysis. If not, a new configuration is made, and the tests are done again. However, when the samples pass the test, a statistical analysis will be done to present the comparative results between the techniques.

Reults and Discussion

The Mission

We proposed a misison to Turtlebot3 to analyze the performance of the algorithms (Figure 5). The mission was to move it from an initial point to a final point in the simulated gazebo environment.

The robot uses AMCL to locate itself and plans the path using the A* and Dijkstra, enabling observing each algorithm's time spent completing the mission and comparing them. Thirty trials were run for each to perform the task to compare the algorithms' performance (Figure 6).

Observing the navigation missions, we observed the trajectory generated by each path-planning algorithm (Figure 7). Meanwhile, Figure 8 presents the time samples for each algorithm.

For the analysis, the normality of the dataset was initially verified through the Shapiro-Wilk test. The data for the Dijkstra algorithm had a p-value of 0.404. For the A star, the p-value was 0.150 for a significance level of 5% for both cases. This value confirms that the data is usually distributed. The average execution time for the mission using the Dijkstra path planner was 20.04 seconds with a standard deviation of 0.158 seconds, and the average time using the A star path planner was 20.14 seconds with a standard deviation of 0.207 seconds. Figure 9 shows the comparison between the two algorithms. For the sample values, we conclude that there was no relevant difference between the navigation time based on the mean and standard deviation.

We used the t-test to compare the average navigation population time between the algorithms. The null hypothesis was Dijkstra, and A star algorithm has an equal average navigation time. The null hypothesis for the significance level of 5% was rejected for a p-value of 0.042, concluding that the population means time for this navigation mission using the A star and Dijkstra algorithms are not equal, with little significance.

Conclusion

We performed a comparison between A* and Dijkstra to make UGVs more efficient in the trajectory planning process, using statistical comparison methods to understand the differences.

Figure 5. Gazebo simulation.

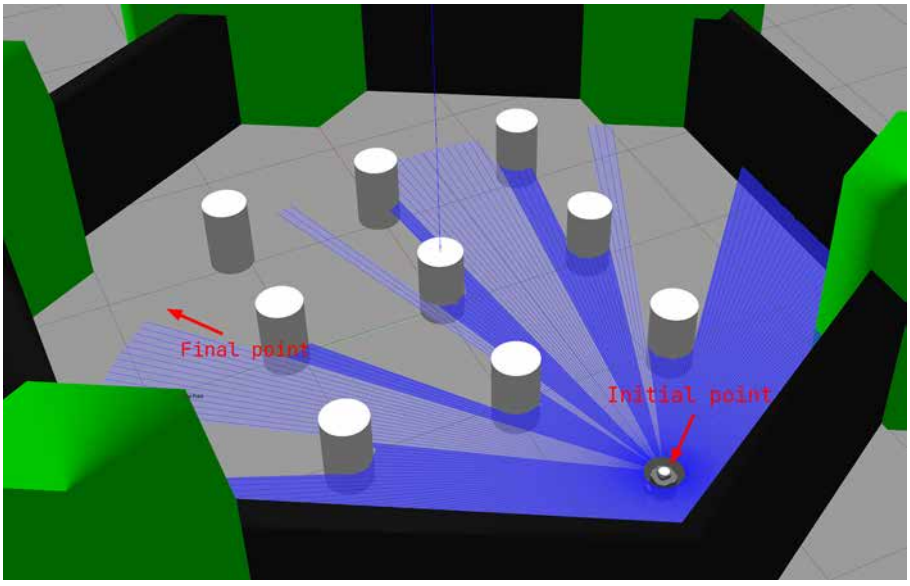


Figure 7. Path planned for each algorithm.

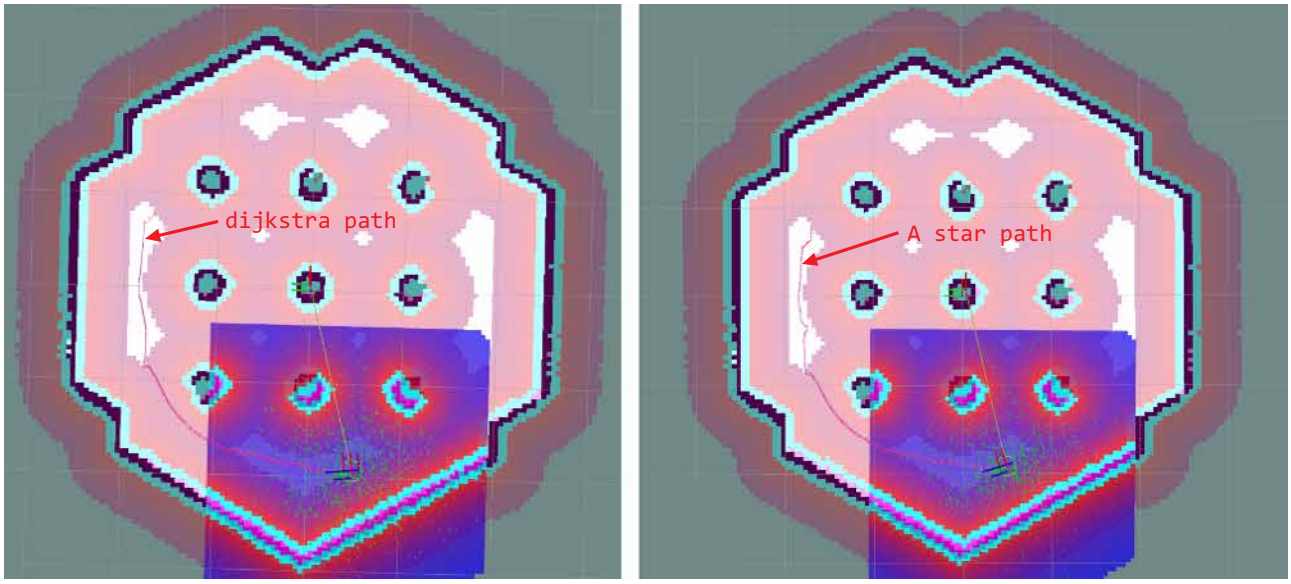
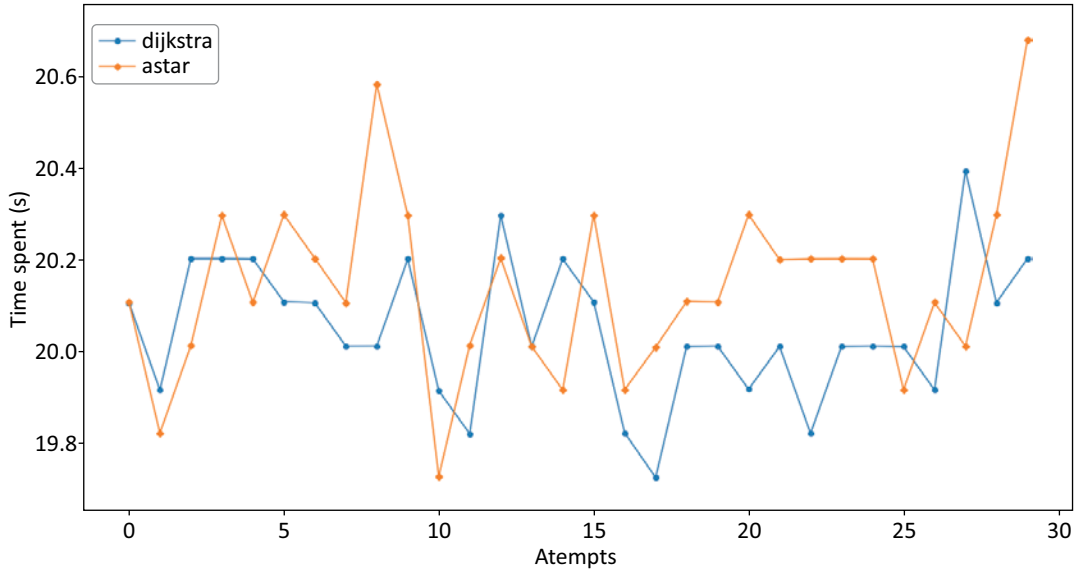


Figure 8. Time samples for each algorithm.

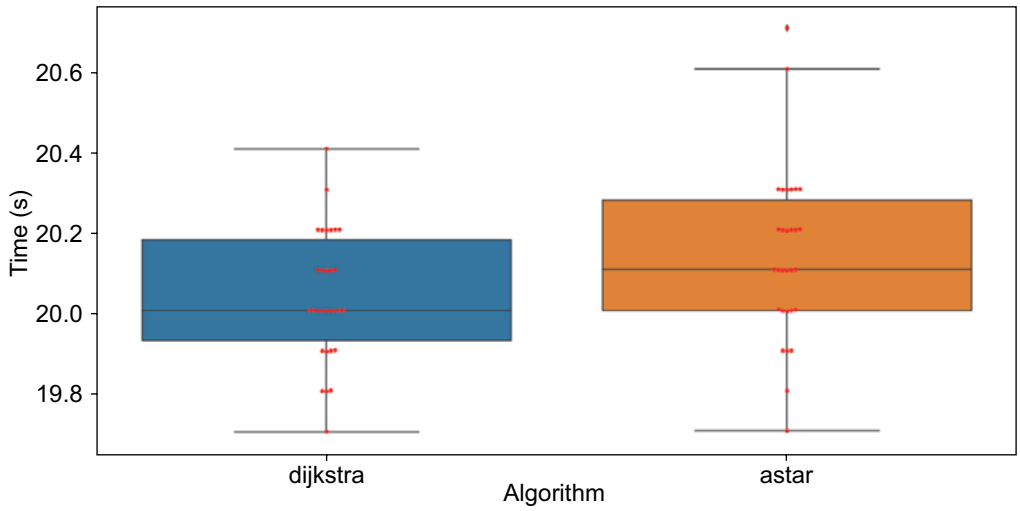


From the data obtained, we observed that the amplitude of the minimum and maximum time for the two algorithms was one second, which within the total time was too small for a definition of which method showed a better performance through graphical analysis. The t-test were used to verify the averages of time spent by each algorithm, and the value obtained to reject the hypothesis was irrelevant.

In this sense, we deduced that the Dijkstra and A* trajectory planning techniques present very close performances in a simulated environment.

At this point, we concluded that the difference between the algorithms for short trajectories becomes irrelevant even if Dijkstra is considered a greendy search algorithm leading to a longer mission time. According to Rachmawati and Gustin [11], the A* algorithm only scans toward the final

Figure 9. Algorithm time comparison.



destination. In contrast, the Dijkstra algorithm does an equally expanding scan for each point ending in exploration with a larger area before finding the final objective.

For future studies on the topic, it is recommended to consider new mission possibilities, mainly regarding the increase in trajectory distance, the presence of more obstacles, and varying scenarios, enabling the analysis of each technique's performance in more contexts.

References

1. Hua C et al. A global path planning method for unmanned ground vehicles in off-road environments based on mobility prediction. Machines MDPI 2022;10(5):375.
2. Guruji AK, Agarwal H, Parsediya D. Time-efficient A* algorithm for robot path planning. Procedia Technology Elsevier 2016;23:144–149.
3. What is TurtleBot? Disponível em: <https://www.turtlebot.com/about/>.
4. Amsters R, Slaets P. Turtlebot 3 as a robotics education platform. In: Springer. International Conference on Robotics in Education (RiE). [S.l.], 2019:170–181.
5. Zhao CW, Jegatheesan J, Loon SC. Exploring iot application using raspberry pi. International Journal of Computer Networks and Applications 2015;2(1):27–34.
6. NAV2 — Documentação da Navegação 2 1.0.0. <https://navigation.ros.org/>. Accessed on June 30, 2022.
7. Inteligência Artificial - 3ª Ed. 2013. <https://www.cin.ufpe.br/~gtsa/Periodo/PDF/4P/SI.pdf>.Accessed on June 30, 2022.
8. Russel PN. Artificial intelligence: A modern approach by Stuart. Russell and Peter Norvig contributing writers, Ernest Davis et al., 2010.
9. NilssonNJ. Shakey the robot. In: [S.l.: s.n.], 1984.
10. Zhao CW, Jegatheesan J, Loon SC. Exploring iot application using raspberry pi. International Journal of Computer Networks and Applications 2015;2(1):27–34.
11. Rachmawati D, Gustin L. Analysis of Dijkstra's algorithm and A* Algorithm in shortest path problem. In: Journal of Physics: Conference Series. IOP Publishing, 2020:012061.