Path Planning Application using Dijkstra Algorithm on Service Robot

Ryan Satria Wijaya[1] Arya Mahendra[2], Senanjung Prayoga[3], and Anugerah Wibisana[4]

[1,2,3,4] Politeknik Negeri Batam, Batam Kepulauan Riau 29461, Indonesia
[1]ryan@polibatam.ac.id
[2]aryamahendra060799@gmail.com
[3]senanjung@polibatam.ac.id
[4]wibisana@polibatam.ac.id

Abstract. The advancement of robot technology in mobile robots is quickly evolving and being utilized in various sectors such as industries, military, medicine, and public services. Challenges consist of perception, localization, motion control, and path planning. The aim of Dijkstra's algorithm, which is a greedy algorithm, is to optimize path planning to improve movement efficiency. Dijkstra's algorithm is a useful method in graph theory to find the shortest path between two nodes in a weighted graph, utilizing an iterative approach to compute the distance. The algorithm being suggested speeds up the initial process by simultaneously determining the shortest route from the starting point to all other points, utilizing various paths or continuing on the same path to reach additional nodes. Nonetheless, it begins at the central node, utilizing data that is not influenced by the route taken. The author experimented with Dijkstra's algorithm using a Service Robot, successfully navigating past three obstacles without any collisions. The robot demonstrated success in finding the shortest and fastest path by maintaining an average speed of 0.23 m/s and average errors of 0.021 meters on the x-axis and 0.017 meters on the y-axis.

Keywords: Path Planning, Robotics, Shortest Path Algorithm.

1 Introduction

The advancement of robotic technology is currently accelerating, particularly in the domain of mobile robots, which are extensively utilised across diverse sectors including industry, military, healthcare, and public services [1]. This shows that the robot has to navigate its environment effectively and avoid any obstacles in its path [2]. The development of dependable, secure, and efficient mobile robots is an important research area in robotics, yet it presents significant challenges [3]. The typical issues include perception, localization, controlling motion, and planning paths [4]. Out of all these domains, path planning is possibly the most crucial[5]. Effective and efficient path planning algorithms are crucial for mobile robots to navigate optimally in dynamic and complex environments [6].

L. Lumombo et al. (eds.), *Proceedings of the 7th International Conference on Applied Engineering (ICAE 2024)*,
Advances in Engineering Research 251,
https://doi.org/10.2991/978-94-6463-620-8_20

Dijkstra's algorithm is a greedy algorithm that chooses the smallest overall cost along the path by considering various distances and offering multiple ways to find the shortest route [7]. This particular algorithm is famous for its capability to discover the most efficient route between two nodes in a weighted graph with no negative weights [8]. Implementing Dijkstra's algorithm on mobile robots is intended to offer the best path planning solutions, enabling the robots to navigate effectively while steering clear of any obstacles in their surroundings [9].

2 Research Method

This research employs techniques such as reviewing literature and gathering data. Research was carried out in the field of literature to gather required information from sources. Data collection was conducted to gather and examine data on path planning [10]. The aim of this research is to optimize, expedite, and find the shortest route planning [11] and can avoid obstacles so that the robot can navigate with the control system using ROS [12] and ESP32 microcontroller [13] with a communication system using ROSserial [14]. In this regard, the result of this research is to move the robot from the starting point to the destination along the path created based on the input data.

2.1 Overview

The robot used in this research is the Service Robot. The robot has 4 mechanical wheels designed to move in any direction, the wheels consist of a central wheel surrounded by a number of rollers that move freely to form a 45° angle with the circumference of the wheel[15]. A3 Lidar sensor [16] used for mapping and obstacle detection. The Lidar sensor is interfaced with Jetson Nano [17] as the map and obstacle access system, which is then integrated with ROS and ESP 32 microcontroller as the robot controller [18].

2.2 Robot Design

In this study the writer designed hardware for robots using 2- level aluminium panels, where the 1st level serves to place hardware such as ESP32, motor drivers, DC motors and voltage reducers on the robot's aluminium plate. On the second level were placed Jetson Nano and Lidar sensor. The dimension of robot is 300 mm long, 270 mm wide, and 250 mm high. The Service Robot prototype can be seen in **Fig. 1**.

Fig. 1. The Service Robot Prototype

2.3 Path Planning

Path planning is a method of finding a route to get a mobile robot to its goals [19]. This is a simple algorithm for the single-source shortest path problem that can efficiently determine the shortest path to any goals [20]. Because the optimal path is not always the shortest, path planning can also be used to determine the shortest or optimal path between two points [21]. There are things to consider, such as the number of rotations and stops required for the robot to operate [22]. The algorithm that will be used in this research is Dijkstra's algorithm.

Dijkstra's algorithm may be an exceptionally successful strategy to decide the shortest way to a goal point [23]. This algorithm is utilized in graph theory to discover the shortest path between two nodes in a weighted graph, accepting that the weight of each path is positive [24]. This algorithm works in an iterative method, calculating the shortest distance from the starting point to all other points in the graph. The flowchart of this algorithm can be seen in **Fig. 2**

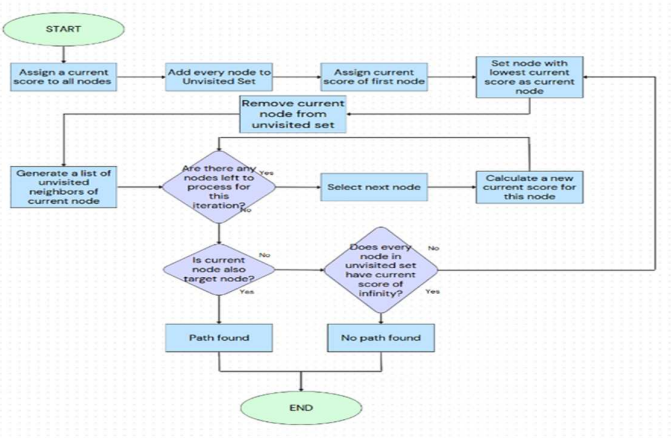

Fig. 2. Dijkstra's Algorithm Flowchart

The proposed algorithm parallelizes the initial process: it starts by finding the shortest paths from the first node to all other nodes, allowing them to use different paths in the best case and the same path to additional nodes in the worst case. The best case for this may be less than 1, but it is still applicable in certain situations. The drawback is that the algorithm presented for comparison starts the calculation directly from the central node of all other nodes, and therefore uses information independent of the path travelled [25].

```
for each node v in G:
    d(v) = infinity;

D = {s};
U = {G-s};

for each node u neighboring with s:
    d(u) = w(u, s);

while U is not empty:
    Let v be the node from U with minimum d(v);

    U = U - v;
    D = D union v;

    for each node u neighboring with v:
        if d(u) > w(u, v) + d(v)
        then d(u) = w(u, v) + d(v);
```

Fig. 3. Dijkstra's Algorithm Pseudocode

On **Fig. 3** describe about pseudocode of Dijkstra algorithm for determining, given a directed network with non-negative weights, the shortest path between a source node and every other node. Here is a breakdown of every part of the pseudocode:

1. Distance Initialization: For all node v in graph G, initialize the distance from the source node s to v as infinity. This indicates that initially all nodes are considered unreachable from the source as we can see in equation (1)

$$\forall v \in G : d(v) = \infty \tag{1}$$

2. Set Initialization: In equation (2), set D is initialized to contain only the source node s. And, in equation (3), set U is initialized with all other nodes in graph G except the source.

$$D = \{s\} \tag{2}$$

$$U = G \setminus \{s\} \tag{3}$$

3. Distance Initialization for neighbor Source: For all node u that is a direct neighbor of source s, initialize the distance d(u) with the edge weight w(u,s), at equation (4).

$$\forall u \in adj(s) : d(u) = w(u,s) \tag{4}$$

4. Main Process of Algorithm: Select node v from U with minimum distance d(v). Remove v from U. Add v to set D. For all nodes u neighboring v, check if the current distance to u(d(u)is greater than the distance through v (w(u,v)+d(v)w(u,v)+d(v)). If

yes, update d(u) with the new value. Do the iteration as long as the set U is not empty. The equation of process of algorithm can be seen at equation (5).

$$\forall u \in adj(v): if\ d(u) > d(v) + w(v,u)\ then\ d(u) = d(v) + w(v,u) \quad (5)$$

To ensure that every node is processed once with the current minimal distance, the algorithm updates the distance based on the edge weights between the nodes and updates the distance from the source to each node. The final result is the shortest path between each node in graph G and the source.

3 Result and Discussion

Testing the Dijkstra algorithm in determining the shortest path and the number of points to be travelled is tested 3 times with same initial position at $x = 0$ and $y = 0$. The obstacles for testing the robot were varied to generate different data for each different environment. This is done in order to get data that varies according to the environment tested on the robot.

In the first test, the robot will move from point A to point B through randomly placed obstacles. The target is that the robot can move to avoid obstacles with the shortest distance and fastest time possible. Then the path planning will be projected on the Rviz visualisation. The first test on the robot can be seen in Fig. 3.

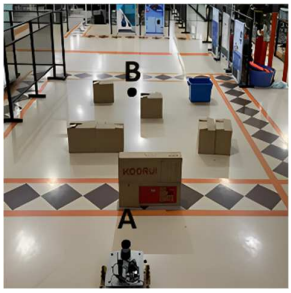

Fig. 4. First Test. Real world obstacle

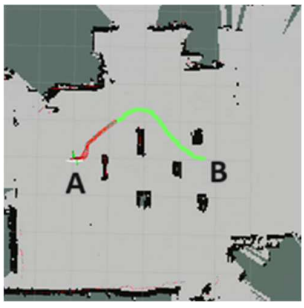

Fig. 5. First Test. Path Planning Visualization in Rviz

In the first test, the robot can move from point A to point B, avoiding obstacles that have been placed in a particular way with a distance of 4.958 meter in 17.18 seconds. in this test the robot can go through obstacles without experiencing collisions.

In the second test, the robot will move from point A to point B through randomly placed obstacles. But in the second test, the obstacles are placed differently from the first test. Then the path planning will be projected on the Rviz visualisation. The first test on the robot can be seen in **Fig. 4** and **Fig. 5**

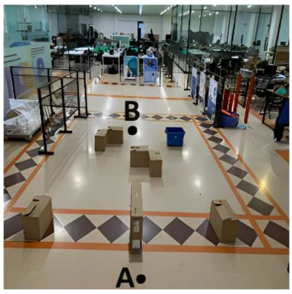

Fig. 6. Second Test. Real-world Obstacle

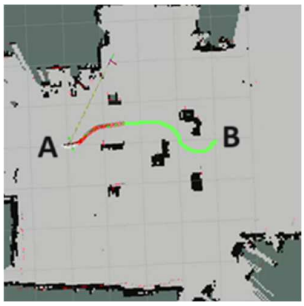

Fig. 7. Second Test. Planning Visualization in Rviz

In the second test, the robot can move from point A to point B, avoiding obstacles that have been placed in a particular way with a distance of 4.642 meters in 20.73 seconds. in this test the robot can go through obstacles without experiencing collisions

In the third test, the robot will move from point A to point B through randomly placed obstacles. But in the third test, the obstacles are placed differently from the previous test. Then the path planning will be projected on the Rviz visualisation. The first test on the robot can be seen in **Fig. 4** and **Fig. 5**

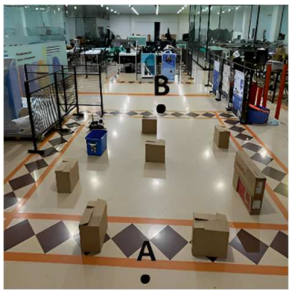

Fig. 8. Third Test. Real world obstacle.

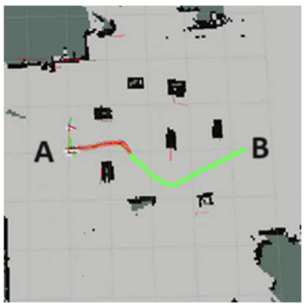

Fig. 9. Third Test. Path Planning Visualization in Rviz.

In the third test, the robot can move from point A to point B, avoiding obstacles that have been placed in a particular way with a distance of 4.867 meters in 18.91 seconds. in this test the robot can go through obstacles without experiencing collisions.

After testing the path planning carried out previously, the data results include the target position, the actual robot position, margin error, the distance travelled by the robot, the time taken for the robot to travel to the target, and the speed of the robot. The following are the test results shown in Table 1.

Table 1. Test Result

Test No.	Target Pos. (m)		Actual Pos (m)		Error Margin (m)		Distance (m)	Time (s)	Velocity (m/s)
	x	y	x	y	x	y			
1	4.027	0.007	3.997	0.021	0.030	0.014	4.958	17.78	0.27
2	4.004	0.013	3.989	0.011	0.015	0.025	4.642	20.73	0.21
3	4.352	0.021	4.332	0.033	0.020	0.012	4.867	18.91	0.22

In this table presents data from three tests involving measurements of the target and actual positions on the x and y axis. The error measured as the difference between the target and actual positions. In addition, the average distance, time, and speed are also calculated and presented for each test. For the average distance travelled, the robot travelled an average distance of 4.822 meters, with an average robot speed 0.23 m/s. For

the error margin, the data obtained for the x-axis 0.021 meters, and for the y-axis 0.017 meters.

4 Conclusion

In this journal, the writer has tested Dijkstra's algorithm as a path planning with Service Robot as the device. In this study, the writer tested with three different obstacles. In this test, the robot can pass through obstacles without experiencing collisions in each different obstacle scenario. The robot moves at an average speed of 0.23 m/s with an average error of the x-axis = 0.021 meters and an average error of the y-axis = 0.017 meters. The writer concludes that testing the robot using Dijkstra's algorithm succeeds in getting the shortest and fastest path on different obstacles without hitting them. This can be proven by doing three test scenarios with different obstacles.

Acknowledgments. The writer would like to thank all those who have provided support and assistance during this research process. The writer would also like to thank the lecturers for their valuable advice and guidance during the research process. Their knowledge and experience were very helpful in shaping and perfecting this research. Finally, the writer would like to thank their colleagues for their co-operation and support. This research would not to be happen without their support. The writer hopes that the results of this research can make a positive contribution to the scientific community and the entire society.[1]

References

1. Raj, R., & Kos, A.: A comprehensive study of mobile robot: history, developments, applications, and future research perspectives. Applied Sciences 12(14), 6951 (2022).
2. Alatise, M. B., & Hancke, G. P.: A review on challenges of autonomous mobile robot and sensor fusion methods. IEEE Access 8, 39830–39846 (2020).
3. Hercik, R., Byrtus, R., Jaros, R., & Koziorek, J.: Implementation of autonomous mobile robot in smart factory. Applied Sciences 12(17), 8912 (2022).
4. Garaffa, L. C., Basso, M., Konzen, A. A., & de Freitas, E. P.: Reinforcement learning for mobile robotics exploration: A survey. IEEE Transactions on Neural Networks and Learning Systems 34(8), 3796–3810 (2021).
5. Wang, H., Qi, X., Lou, S., Jing, J., He, H., & Liu, W.: An efficient and robust improved A* algorithm for path planning. Symmetry 13(11), 2213 (2021).
6. Amin, J., Bokovic, J., & Mehra, R.: A fast and efficient approach to path planning for unmanned vehicles. In: AIAA Guidance, Navigation, and Control Conference and Exhibit, p. 6103 (2006).
7. Al Hakim, R. R., Satria, M. H., Arief, Y. Z., Pangestu, A., Jaenul, A., Hertin, R. D., & Nugraha, D.: Aplikasi Algoritma Dijkstra dalam Penyelesaian Berbagai Masalah. Expert 11(1), 345994 (2021).

[1] For future research, the writer recommended to paying attention to the limitations that exist in this study, and can develop more effective and safety path planning methods.

8. Noto, M., & Labomtories, F. P.: A Method for the Shortest Path Search by Extended Dijkstra Algorithm. In: Systems, Man, and Cybernetics, IEEE International Conference, pp. 2316–2320 (2000).
9. Yang, J., Qu, Z., Wang, J., & Conrad, K.: Comparison of optimal solutions to real-time path planning for a mobile vehicle. IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans 40(4), 721–731 (2010).
10. Pratiwi, H.: Application of The Dijkstra Algorithm To Determine The Shortest Route From City Center Surabaya To Historical Places. Jurnal Teknologi Dan Sistem Informasi Bisnis 4(1), 213–223 (2022).
11. Raja, P., & Pugazhenthi, S.: Optimal path planning of mobile robots: A review. International Journal of Physical Sciences 7(9), 1314–1320 (2012).
12. Köseoğlu, M., Çelik, O. M., & Pektaş, Ö.: Design of an autonomous mobile robot based on ROS. In: 2017 International Artificial Intelligence and Data Processing Symposium (IDAP), pp. 1–5. IEEE (2017, September).
13. Hamici, N.: Design and Implementation of an Autonomous Mobile Robot Based on ESP-32 (2023).
14. Newman, A., Yang, G., Wang, B., Arnold, D., & Saniie, J.: Embedded mobile ROS platform for SLAM application with RGB-D cameras. In: 2020 IEEE International Conference on Electro Information Technology (EIT), pp. 449–453. IEEE (2020, July).
15. McInerney, I., & Team, F. R. C.: Simplistic Control of Mecanum Drive. FRC Team 2022 (IL Mathematics and Science Academy) (2022).
16. Molder, C., Toma, D., & Țigău, A.: Navigation algorithms with LIDAR for mobile robots. Journal of Military Technology, 2(1) (2019).
17. Oneil, E., & Sugiarto, I. (2019). Omni-Directional Mobile Robot Control using Raspberry Pi and Jetson Nano. International Journal of Industrial Research and Applied Engineering, 4(2), 57–62.
18. Díaz-Téllez, J., García-Ramírez, R. S., Pérez-Pérez, J., Estevez-Carreón, J., & Carreón-Rosales, M. A. (2023). ROS-based Controller for a Two-Wheeled Self-Balancing Robot. Journal of Robotics and Control (JRC), 4(4), 491–499.
19. Li, X. (2021, November). Path planning of intelligent mobile robot based on Dijkstra algorithm. In Journal of Physics: Conference Series (Vol. 2083, No. 4, p. 042034). IOP Publishing.
20. Luo, M., Hou, X., & Yang, J. (2020). Surface optimal path planning using an extended Dijkstra algorithm. IEEE Access, 8, 147827–147838.
21. Amirullah, R., Rusdina, A., & Darlis, D. (2021). Implementasi Sistem Path Planning Dan Routing Untuk Mobile Robot Berbasis Visible Light Communication. eProceedings of Engineering, 8(5).
22. Zhang, T. W., Xu, G. H., Zhan, X. S., & Han, T. (2022). A new hybrid algorithm for path planning of mobile robot. The Journal of Supercomputing, 78(3), 4158–4181.
23. Alyasin, A., Abbas, E. I., & Hasan, S. D. (2019). An Efficient Optimal Path Finding for Mobile Robot Based on Dijkstra Method. 4th Scientific International Conference – Najaf – IRAQ (4th SICN-2019).
24. Ubaidillah, A., & Sukri, H. (2023). Application of Odometry and Dijkstra Algorithm as Navigation and Shortest Path Determination System of Warehouse Mobile Robot. Journal of Robotics and Control (JRC), 4(3), 413–423.
25. Bogdan, P. (2015, May). Dijkstra algorithm in parallel-case study. In Proceedings of the 2015 16th International Carpathian Control Conference (ICCC) (pp. 50–53). IEEE.