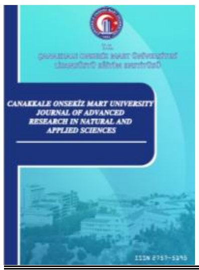
e-ISSN 2757-5195



Çanakkale Onsekiz Mart University

Journal of Advanced Research in Natural and Applied Sciences

Open Access

doi.org/10.28979/jarnas.1119957 2023, Vol. 9, Issue 1, Pages: 205-217 dergipark.org.tr/tr/pub/jarnas

A Modified Dijkstra Algorithm for ROS Based Autonomous Mobile Robots

Orkan Murat Çelik[1], Murat Köseoğlu[2,*]

[1]Havelsan A.Ş., Ankara, Türkiye
[2]Department of Electrical-Electronics Engineering, Faculty of Engineering, Inonu University, Malatya, Türkiye

Article History
Received: 23.05.2022
Accepted: 08.12.2022
Published: 05.03.2023

Research Article

Abstract − Autonomous Mobile Robots (AMRs) are frequently used in many fields of technology. In this study, an AMR was designed to execute different path planning algorithms. Firstly, working principle, system architecture and motion planning of AMR are presented. Then, a map for the current environment is produced by a Robot Operating System (ROS) powered AMR which was designed for this study. The AMR locates itself on the produced map with the aid of an integrated Light Detection and Ranging sensor (LIDAR). The locomotion of AMR to a user-defined target on the produced map is performed by an optimal path based on AMR's own navigation plan. Two different path planning algorithms, which are Dijkstra's algorithm and a modified version of Dijkstra's algorithm, are executed on a cost-effective AMR platform, which has the capability of Simultaneous Localization and Mapping (SLAM). The reason why Dijkstra algorithm is handled in this study rather than A*, D* and RRT algorithms is that this algorithm is a basic and widely used algorithm. Dijkstra's algorithm is modified, and pros and cons of the modified algorithm are analysed compared to Dijkstra algorithm. The proposed algorithm and navigation of AMR are tested both in real time in real world and as a simulation in Gazebo. Two algorithms were compared according to the results obtained from the robot locomotion both in real application and simulation environment. It is observed that the modified version of the Dijkstra's algorithm comparatively yielded a bit more satisfactory results in the aspect of path planning.

Keywords − *AMR, dijkstra algorithm, path planning, robot navigation, ROS*

1. Introduction

Different robotic technologies and production methods based on the robots, which have been increasingly involved in production in recent years, have rapidly developed and evolved to high levels, especially after the 3rd Industrial Revolution. When the robots were first used in the industry, it was aimed to perform a predefined task at a certain time and with a certain quality in the production process. Since the main purpose of most of the industrial robots produced until today is to perform a defined task, it was sufficient to simply program and use the robots in accordance with the requirements of the production line. However, as the production of different items have become more complex in several industrial fields, it has become a necessity to evolve production techniques to adapt to the resulting complexity. Depending on the technological developments and requirements, some major improvements have been made in robot designs, and the robots have been technologically updated in accordance with the requirements. In this way, it has been possible for robots to adapt to the developing industry. As a result of the complexity brought by the developing technology, it has become inefficient to use industrial robots only for a specific job. Accordingly, Autonomous Guided Vehicles (AGV), which were widely used in the industry, have led up the development of AMRs (Autonomous Mobile Robot). Since the AMRs have a considerable locomotion capability, they can perform complex tasks under

1 orkanmcelik@gmail.com
2 murat.koseoglu@inonu.edu.tr
*Corresponding Author

different working conditions and environments, such as disaster relief operations, factories and restaurants. Depending on their interacting environment, they can be classified as aquatic, terrestrial and airborne (Ben-Ari Mordechai and Mondada, 2018). Arkin and Murphy emphasized the importance of autonomous navigation and AMRs in the industrial plants and flexible production systems in future. They also gave comparative basic information on autonomous robot architecture and explained the disadvantages of AGVs (Arkin & Murphy, 1990). Efficient navigation is one of the most important capabilities of autonomous mobile robots (AMRs) to successfully perform industrial tasks, especially in irregular environments with moving obstacles (Nguyen et al., 2022).

An AMR is capable of navigating in an unpredictable environment. AMRs can sense the environment, create a model based on the environmental properties and locate itself in this model. This capability enables AMR to make a navigation plan and optimize this plan by a special planning algorithm. This is known as simultaneous localization and mapping (SLAM) (Köseoğlu et al., 2017). As the environmental flexibility increases, the AMRs have been used for different scenarios in a wide range of industrial applications. (Pagani et al., 2018).

When a task is assigned to an AMR, it plans how to perform this task, then it performs this task according to this plan. The AMR makes this plan by using several algorithms, which work on itself and include flexible steps defining how to this task will be performed. This process is called autonomy.

In this study, an AMR, which performs the movement task by using Dijkstra's Shortest Path Algorithm and Modified Dijkstra's Shortest Path Algorithm, is developed. Then, the performance of the AMR is analyzed for these algorithms, comparatively. In the locomotion process, the AMR localizes itself in the current environment by using SLAM algorithm and completes its movement from an initial point to a target point which is defined by the user.

2. Materials and Methods

In this section, some basic concepts, methods and tools, which are used in the movement and path planning of the AMRs, are presented.

2.1. Design of the AMR

The AMR used in this study was designed cost-effectively by using off-the-shelf commercial products. Details of hardware is explained below:

- A Raspberry Pi 2 Model B was used as single board computer (SBC) for the Operating System (Ubuntu 14.04) and ROS (Indigo). All of the robotic algorithms were executed on the SBC.
- An STM32F4 Discovery was used as the peripheral controller which is gathering data from the sensor and driving the motors for locomotion. The software developed for the peripheral controller uses FreeRTOS to keep all tasks running in real time.
- An RpLidar A1 that collects point cloud data for SLAM algorithms was used to create a map describing the perimeter of the AMR.
- As actuator two 12V 76 RPM gear motors were used. Each motor has a 64 CPR quadrature encoder at the motor shaft for the odometry calculation with the direction information. The motor was driven by PID signals which generated by the peripheral controller based on the velocity signal which calculated by the ROS algorithms.
- A 9 DOF RazorIMU was used for the odometry correction based on Extended Kalman Filter.

2.2. Localization

For a mobile robot, localization means that the robot positions itself on the environment in which it will move. In robotics, localization is an important problem. Since all the movement of the mobile robot is estimated based on an initial point, its position must be determined accurately for the movement planning to take place as intended. Position of an autonomous robot can be described as a vector given in Equation 2.1 in the Cartesian coordinate system as seen in Figure 1. In this figure, x is the position component on the X axis,

y is the position component on the Y axis, and θ is the heading angle. v(t) is the velocity, and w(t) is the rotation speed of the robot.

$$p = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \tag{2.1}$$

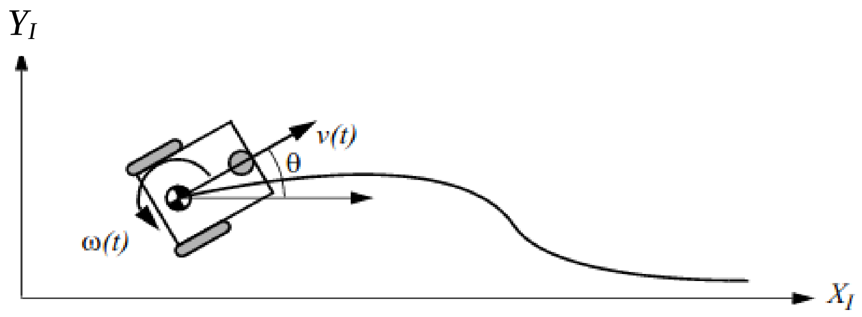


Figure 1. Position and Movement of a differential-drive robot (Siegwart et al., 2011).

2.3. Dead Reckoning

Dead reckoning means that the next position of a moving object is estimated based on its previous position. In addition to being an important position estimation method especially in aerial and nautical vehicles, it is also frequently used in mobile robotics. Making an estimation based on the latest velocity and obtained direction data is an important issue for dead reckoning (Tsai, 1998). It should be noted that the difference between the estimated location and the actual location can increase by the use of former velocity and direction data.

2.4. Mapping

2.4.1. Map Based Localization

The environment can be expressed in a Cartesian coordinate system, and the position of the robot can be shown on the coordinate plane for a robot moving on a flat surface. In addition, a map can be created with the help of SLAM algorithms and a sensor (Lidar, Stereo Camera, etc.) placed on the robot. The map which is created by the robot used in this study is given in Figure 2.

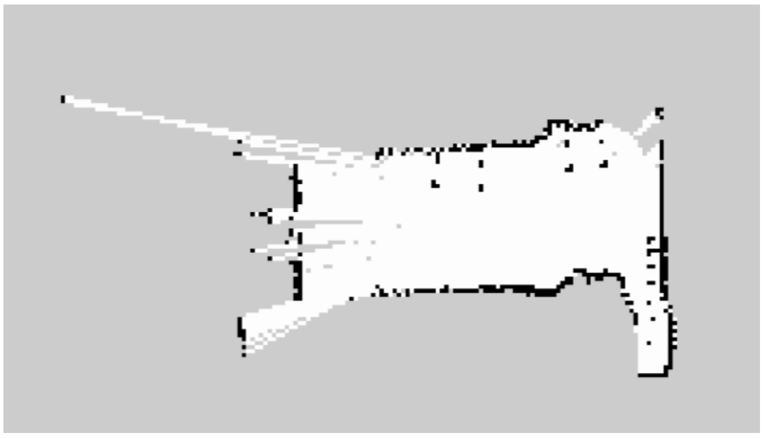


Figure 2. A map example that was created by the designed robot for this study.

2.4.2. Simultaneous Localization and Mapping (SLAM)

SLAM is the building of an environmental model of the environment in which a mobile robot is located, positioning itself in the model it creates and mapping its surroundings according to the created environmental model (Wolf & Sukhatme, 2005). Various sensors have been developed for SLAM to solve the fundamental problem of finding robot position in a global representation (Merzlyakov & Macenski, 2021).

For most of the SLAM algorithms, the robot continually updates and expands the environmental model while it is in motion (Durrant-Whyte & Bailey, 2006). In other words, the map created by SLAM is a dynamic map. In Figure 3, the position of the robot is shown on the map based on SLAM.

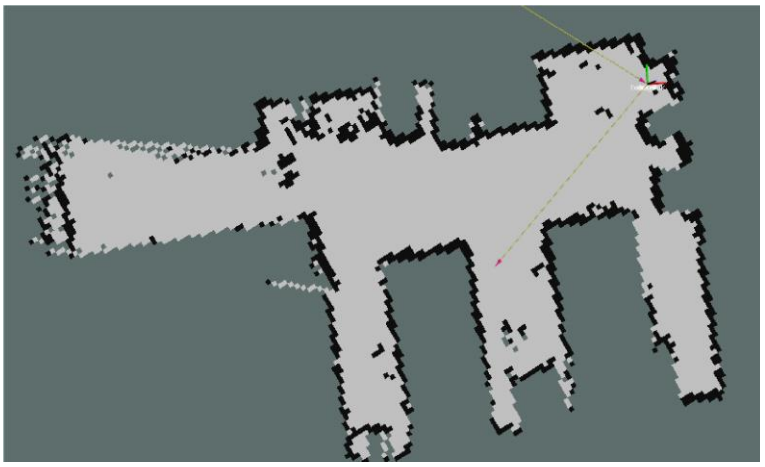


Figure 3. SLAM example which was created by the designed robot for this study.

The SLAM can be explained mathematically by following steps:

Position of the robot (X_T) at each sample time T can be expressed as

$$X_T = \{x_0, x_1, x_2, \dots, x_T\} \tag{2.2}$$

If the motion of the robot in the time interval of $T-1$ and T is expressed as U_T, and it is assumed that the motion data is obtained from encoder readings or control inputs to the motor, the time dependent motion of the robot can be written as:

$$U_T = \{u_0, u_1, u_2, \dots, u_T\} \tag{2.3}$$

Accordingly, if it is assumed that the value of m_i represents the objects around the robot, M is expressed as:

$$M_T = \{m_0, m_1, m_2, \dots, m_{n-1},\} \tag{2.4}$$

Assuming that each sensor on the robot takes only one measurement at a time, we can express the entire measurement for one sensor as follows:

$$Z_T = \{z_0, z_1, z_2, \dots, z_T,\} \tag{2.5}$$

According to the stated terminology, we can accept SLAM as a structure that improves the map model (*M*) and the path of the robot according to the data obtained from position (X_T), odometry (U_T) and observations (Z_T). There are two different algorithms as full SLAM and online SLAM that are used by AMRs. AMR can follow many different paths composed of consecutive joints on the map to reach the target point. The probability of the next joint to be followed by AMR along the full SLAM $X_{T,}$ depends on different parameters, and this relationship is represented by the following equation:

$$p\{X_T, M \vee Z_T, U_T\} \tag{2.6}$$

A difference between the full SLAM and online SLAM can be explained as follows: In online SLAM, x_t shows the next common probability along M for the next joint to be followed. As seen, full SLAM tries to estimate the entire path (X_T) of the robot, while online SLAM only shows the current position (x_t).

2.5. Navigation and Motion

Autonomous navigation is the autonomous movement of the robot from its current location to a target point, which is located on the map defined by the SLAM algorithm previously. Navigation generally works based on odometry data which is provided by the encoders. Velocity and steering commands, which are calculated by using sensor data, are transmitted to the robot's controllers. In the current study, steering commands are also transmitted to the controllers as velocity information, since the designed robot is a specific robot that moves with a differential driving system. However, there are some important concepts that should be known in the navigation process. These concepts can be explained as:

TF**:** TF is a package that lets the user keep track of multiple coordinate frames over time. TF maintains the relationship between coordinate frames in a tree structure buffered in time, and lets the user transform points, vectors, etc. between any two coordinate frames at any desired point in time (*Http://Wiki.Ros.Org/Tf,* 2021).

Odometry**:** A mobile robot uses odometry data to track its movement as it navigates a familiar environment. However, any uncertainty in the odometry data confuses the robot about its current position. (Panigrahi & Bisoy, 2021). Odometry can be defined as the relative position determination of a robot according to the surrounding objects with the help of integrated sensors such as IMU, encoder, etc. For example, if the circumference and the number of rotations of the wheels of a mobile robot are known, the path length and displacement of the robot can be calculated according to its initial position. Because odometry calculations are generally based on the result of the sensor data, the result of erroneous readings from the sensors makes the calculations increasingly inaccurate.

In this study, as primary sensor readings for odometry calculations, 2 channel quadrature encoders are used. In addition to the velocity information, the direction of the rotation of the motor can be obtained by quadrature encoders due to 90 degrees phase shift between 2 channels of encoders. However, in case of misreading from the encoder, the Navigation system uses an IMU (Inertial Measurement Unit) to make more accurate odometry calculations. For more accurate odometry calculations, sensor readings are combined by EKF (Extended Kalman filter) which is one of the sensor fusion algorithms.

EKF (Extended Kalman Filter)**:** Almost all sensors, which are used in robotics, generate some noise associated with the measured signals during the measurement process. This noise quite affects the system and navigation performance negatively. If there is a Gaussian noise, which may cause a deterioration in the system, the Kalman filter significantly reduces the average measurement error in a gradual way. Thus, the Kalman filter used in the system is a causal filter, which operates by considering the former measurements.

An EKF generally reduces a nonlinear system model to a linear system for each sample time. For a system, which includes a single variable, this process uses the current value of the variable and the derivative of that variable at each sample time. However, in a system with several variables, Jacobian matrices are used to analyze and obtain an approximate linear model of the system. The general expression of EKF can be written as below:

$$x_k = Ax_{(k-1)} + Bu_k + w_{(k-1)} \tag{2.7}$$

$$z_k = Hx_k + v_k \tag{2.8}$$

In Equations 2.7 and 2.8, k describes the measurement number (for each separate measurement), x_k is the instant signal value, $x_{(k-1)}$ is the previous instant signal value, u_k is the control signal, $w_{(k-1)}$ describes the disturbances, z_k is the sensor measurement, v_k is the sensor disturbance, A is the state matrix, B is the input matrix and H is the observation matrix. When Equations 2.7 and 2.8 are considered, it can be clearly seen that EKF cannot predict the position of the robot for the non-gaussian values of the $w_{(k-1)}$ and v_k.

EKF is implemented in 2 stages. First stage is the prediction phase and can be written as:

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_k \tag{2.9}$$

$$\hat{P}_k^- = AP_{k-1}.A^T \tag{2.10}$$

In Equations 2.9 and 2.10, k is the index term for the measurement, $\hat{x}_k^-$ is the predicted value, $\hat{x}_{k-1}$ is the previous predicted value, u_k is the control signal applied to the system, $\hat{P}_k^-$ is the error covariance and P_{k-1} is the former error covariance. When the prediction equations are inspected, these equations show that $\hat{x}_k^-$ element expresses a general prediction for the first loop of the filter due to the update process after prediction phase. The same methodology is valid for the calculation of $\hat{P}_k^-$. The main duty of the prediction phase is to stabilize the filter output with calculation of variables at the update phase.

Second stage of EKF is the update phase and can be expressed with three equations which are written below:

Calculation of the Kalman filter gain:

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1} \tag{2.11}$$

- Prediction Update by Sensor Measurements:

$$\hat{x}_k = \hat{x}_k^- + K_k(Z_k - H\hat{x}_k^-) \tag{2.12}$$

- Updating of Error Covariance:

$$P_k = (1 - K_k H)P_k^- \tag{2.13}$$

When Equations 2.11, 2.12 and 2.13 are examined, it’s clearly shown that these equations have analogies with the standard Kalman Filter equations. In each cycle of the filter, the prediction and update phases work recursively. It can be seen that each sensor measurement should be defined with an independent and different Z variable in Equation 2.12. Thus, sensor fusion must be set according to the weight of each different Z variable.

***Navigation Methodology*:** Navigation planning for an autonomous mobile robot could be basically described as motion planning from a specific initial point to a target point which is defined by the user. Basically, the success of navigation planning is measured by whether it is heuristic or not. A heuristic navigation algorithm should create the shortest path to the target point.

Based on the data acquired from the environment, there are two types of motion planning approaches, namely global path planning and local path planning. If the environment is well known, it is identified as global path planning, and the motion planning serving in unknown environment is identified as local path planning (Tang SH & Zulkifli N, 2015).

In this study Dijkstra's Algorithm was used as a global path planning algorithm. Dijkstra's algorithm is an algorithm for finding the shortest paths between the nodes in a graph (*Dijkstra's Algorithm*, 2021; Iscan & Tuncel, 2022). Dijkstra is a breadth-first-search (BFS) algorithm for finding the shortest paths from a single source vertex to all other vertices, and it is able to produce the shortest route (Fadzli et al., 2015; Kumar & Gao, 2021).

Dijkstra's algorithm is a simple but powerful method (Guo, 2013). When the robot is located on a well-defined and mapped environment, the algorithm can calculate the heuristic path swiftly. According to the algorithm, the initial point of the robot could be described as the first node, and the desired point could be described as the last node. Dijkstra's method is a method that can determine the shortest route from the boarding point to the arrival point by the smallest weight (Hartomo et al., 2019).

The algorithm used in this study is given below:

Step 1: To calculate the shortest path from an initial node to a target node, an empty array/list is created to keep vertices which generate the path tree.

Step 2: All the nodes except the initial node are defined as unvisited nodes, and the initial distances of all unvisited nodes are set as infinite. The distance of the initial node is set to zero.

Step 3: Each distance value is calculated from initial node (Current node) to neighbor nodes. The shortest distance value is appended to the shortest path list as an element. Along the path finding process, this step is applied between each current node and neighboring node. After the end of calculation step for each node, the current node is removed from the unvisited nodes list and appended to the visited nodes list. Third step is applied until the target node is found.

Step 4: When the robot arrives to the target node, the algorithm ends.

When Dijkstra's Algorithm is compared with the other weighted algorithms, it can be seen that it is one of the simplest implementable algorithms (Kumar & Gao, 2021). Also, it has a flexible structure and can be modified conveniently to find the most available path on the map. Calculation cost of the algorithm is very low since the algorithm is based on the basic mathematical calculations. To calculate the most available path, the robot can try a few of the existing modified algorithms. In this study, the original Dijkstra's algorithm and modified Dijkstra's algorithm are tested on an AMR comparatively. The evaluations are presented in the next section.

3. Results and Discussion

ROS is a widely used platform for robot's implementation (Ochiai et al., 2014; Zaman et al., 2011). ROS official packages are adequate in common robotics tasks. Furthermore, ROS provides API to build custom packages or communicate with external systems or equipment e.g. interfaces and planner (Ochiai et al., 2014).

In this study, a ROS based differential drive AMR was designed for the implementation of Dijkstra's algorithm (Figure 4). The AMR is a custom design, and it has own lidar sensor to identify environment and generates a map by Hector SLAM package, and the ROS software version preferred in this study is the ROS indigo. The nodes in the Dijkstra's algorithm were determined as the vertices of grids on the map. By this way, each corner on the map could be described as a node.

The nodes were clearly shown on the map in Figure 5. This figure shows a navigation plan which is denoted by a thin red line. The navigation plan was prepared based on Dijkstra's original algorithm. The green dot shows the initial node, and the purple one shows the target node. The thick red arrow shows the motion of the robot. Along the study, robot actuators have been driven by the Proportional-Integral-Derivative (PID) algorithm, which is designed for this study. Although there are different tuning methods such as Ziegler–Nichols and Tyreus Luyben, the PID coefficients in this study were tuned by try and error method due to the ease of application. The same PID parameters were used in all of navigation plans. The simplicity and fast computation of the PID controller made it the most popular low-level control strategy, and it is currently implemented in several mobile robots' control systems (Carlucho et al., 2019). For this reason, PID controller is selected as a motion control method. As seen in Figure 5, the robot has reached the target point by using its own navigation plan. When Figure 5 is inspected, the entire navigation plan passes through the nodes on the map until it reaches the target point. So, the robot has applied almost the same plan toward to the target point, but there is a little deviation from the plan due to physical reasons such as the features of the motors, PID algorithm tolerance and encoder counts. However, the heading angle of the robot, which is described with a thick arrow cone, is directed to the target node. For these reasons, there is a particularly small deviation according to the navigation plan. When the same algorithm is applied for a different target as shown in Figure 6, the observed deviation is more than the deviation shown in Figure 5.

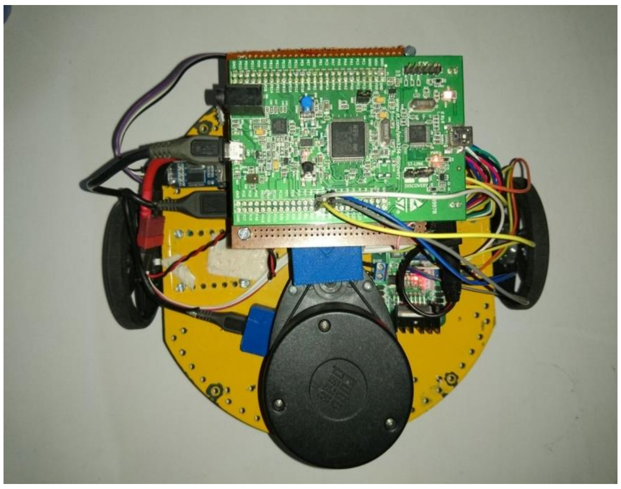


Figure 4. The designed AMR for this study

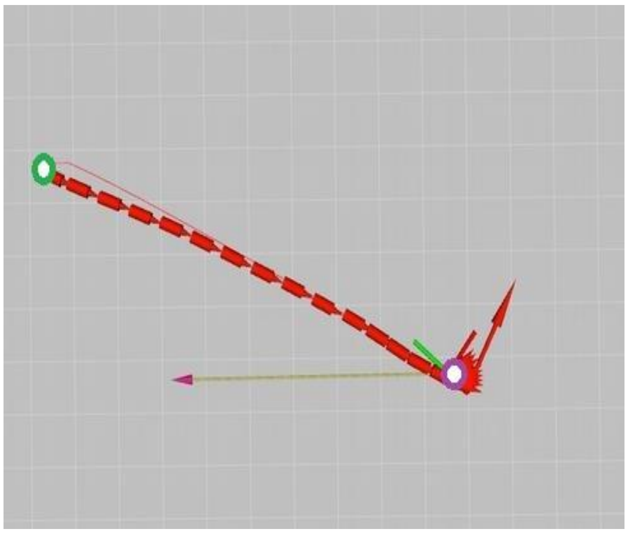


Figure 5. Nodes and navigation plan on the map

When the motion, which is shown in Figure 6, is compared with the motion shown in Figure 5, it is seen that the main difference in the beginning of the motions is the initial head angle of the robot. The target is defined almost 90 degrees on the left side of the initial heading of the robot. According to the illustration in Figure 6, the navigation plan is almost the same as the plan that is shown in Figure 5. In the implementation of the navigation plan, which is shown in Figure 6, the robot has started its motion by rotating heading angle. The linear motion of the robot does not start until the error between the navigation plan and the robot's head angle decreases. Along all motion tracks from start point to target point, a little deviation can be seen on the motion. The deviation is thought to arise from the non-ideal PID algorithm.

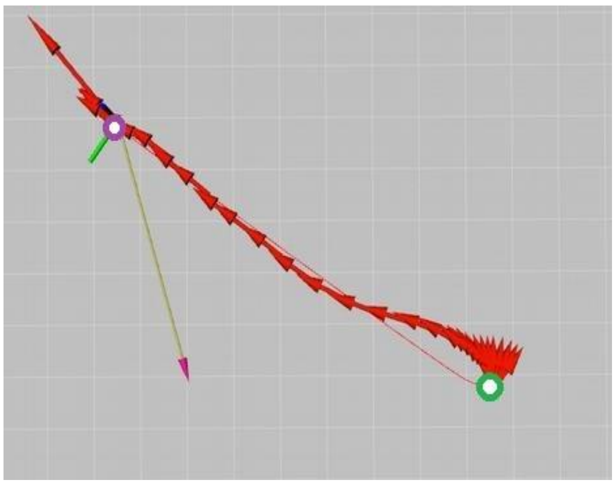


Figure 6. Dijkstra’s algorithm-based navigation plan and motion trail on the map

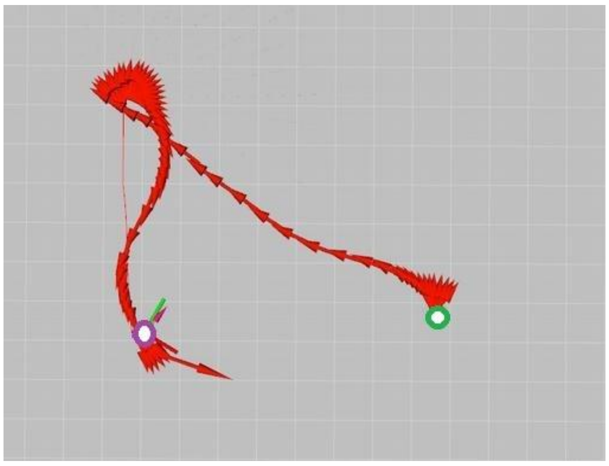


Figure 7. Modified Dijkstra’s algorithm-based navigation plan and motion trail on the map

Figure 7 shows the application of a navigation plan, which is shown by a thin red line, based on the Modified Dijkstra’s algorithm. The modified Dijkstra’s algorithm predicts the path by using the basic estimation principles of conventional Dijkstra’s algorithm except the third step of the algorithm. According to the original algorithm’s third step, the algorithm just calculates the distances between the neighbour nodes. The modified algorithm uses the same calculation methodology of the classic Dijkstra’s algorithm, but as a difference, the modified algorithm also calculates the further neighbours’ successive node distances after the existing node while the neighbour’s node distances are calculated. Main aim of the modified algorithm is to search the possible shorter navigation paths as well as possible.

When Figure 7 is inspected, the robot navigation could be described in two stages. The first stages of both two navigation algorithms are similar as seen in Figure 5 and Figure 6. In the second stage of the navigation algorithm, the target point has been defined at 135 degrees left from the robot’s heading. The motion of the robot is illustrated with a red thick curved line in Figure 7. The curve results from the calculation of the third step of the modified algorithm. According to the modified algorithm, the navigation plan is shown to be more optimistic than the conventional algorithm due to the soft direction changes which are predicted. It is thought that the unexpected slight deviations arise from the non-ideal tuning of PID algorithms, and some problems encountered in practice due to the component tolerances.

As mentioned above, at first, a real ROS Indigo based AMR was developed in the study. The proposed algorithm and the classical Dijkstra algorithm were practically executed and compared by using this robot. Then, the same algorithms were executed and compared in Gazebo simulation environment which includes obstacles. This simulation environment was constructed by using ROS Humble-based Turtlebot3, and the comparison was made by considering both two path plans in this environment. These two path plans are presented in Figure 8 and Figure 9. In both path plans, it has been observed that the modified Dijkstra algorithm produces approximately 1% to 3% shorter paths than the classical Dijkstra algorithm.

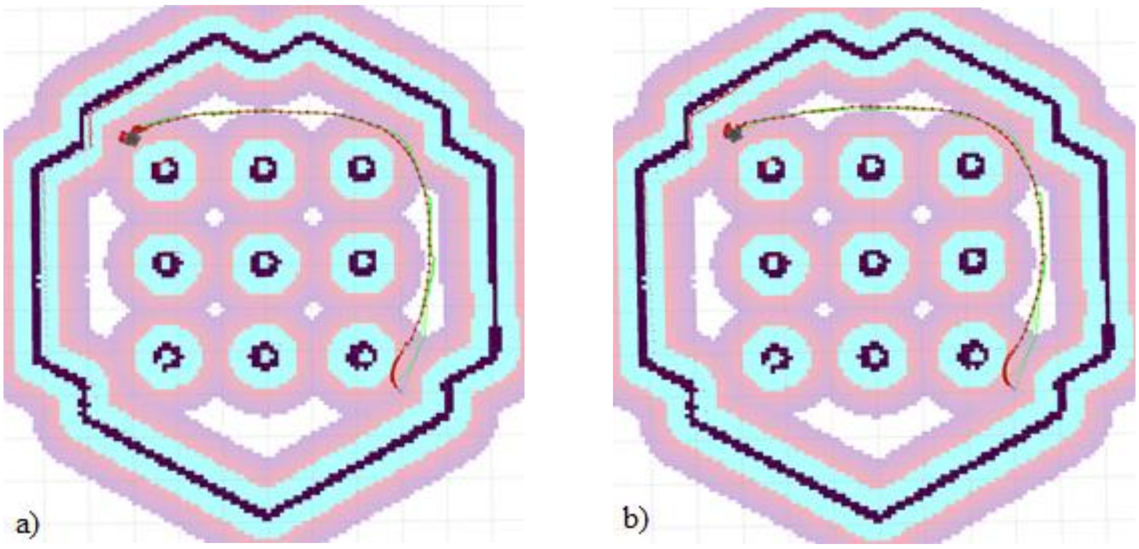


Figure 8. The movement of AMR by considering the Classical Dijkstra (a) and Modified Dijkstra (b) algorithms for path planning in Gazebo TurtleBot World environment.

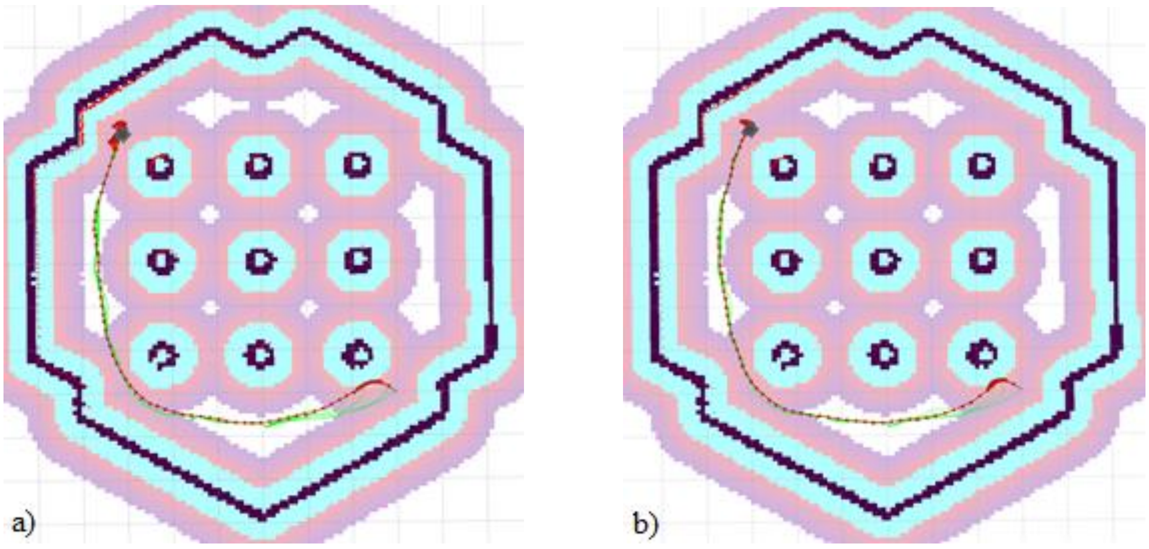


Figure 9. The movement of AMR by considering the Classical Dijkstra (a) and Modified Dijkstra (b) algorithms for path planning in Gazebo TurtleBot World environment.

In Figure 8a, the path, which is obtained by the classical Dijkstra algorithm, and the movement of the robot on this path planning are presented. This algorithm is simulated on ROS Humble using Gazebo and Rviz with a plugin written on the Navigation2 stack. The Navigation2 stack was designed for a high-degree of configurability and future expansion (Macenski et al., 2020). In Figure 8b, the path planning obtained by the modified Dijkstra algorithm and the movement of the robot, which is simulated as in Figure 8a, are shown. The red line indicates the odometry data of the robot's movement on the plan, while the green lines indicate the dynamic path plan produced by the algorithm. The navigation2 stack has its own distance measuring tool. All distance measurements of the path plans used in this study were obtained by using the Distance Remaining data on the Navigation2 panel on Rviz.

The reason why the green lines are constantly changing throughout the robot's movement is that the Navigation2 stack dynamically updates the movement plan by making a cost-based planning. As seen in Figures 8a and 8b, the path length between the starting point and final point of the movement was measured as 651 cm for classical Dijkstra algorithm, while it was measured as 631 cm for the proposed algorithm. In the same way for the other path planning, the path length between the starting point and final point of the movement was measured as 640 cm for classical Dijkstra algorithm (see Figure 9a), while it was measured as 633 cm for the proposed algorithm (see Figure 9b).

As seen in Figure 8 and Figure 9, there are cylindrical obstacles in the environment. The path planning and the simulation are realized by considering the cost values based on Navigation2 stack around these obstacles. The cost values are determined according to the risk factor around each obstacle. The risk factor is a quantitative value and expresses the crash risk of the robot in the environment during the movement.

Both the classic Dijkstra algorithm and the modified Dijkstra algorithm try to achieve small cumulative cost value for the planned route by trying to minimize the cost value for each step while planning the navigation. In this context, the magenta-colored regions on the images represent high-cost zones, the blue-colored regions represent the zones with medium-cost values, and the white-colored regions represent low-risk zones.

4. Conclusion

The Dijkstra’s Shortest Path Algorithm is one of the common path planning algorithms in mobile robotics. Due to the current pace of mobile robotics, the algorithm should be upgraded to different variations to improve the task completion performance. In this study, Dijkstra’s Shortest Path Algorithm is modified by continuously estimating the node distances for two successive steps between two successive neighbors in addition to the estimation of different neighbor node distances for only one step. Thus, to find the other optimal paths for the robot, the AMR does not only consider the next shortest node distance, but it also estimates the total shortest path for successive two steps by summing two distances between subsequent nodes. The proposed algorithm gives promising results in terms of total estimation time as well as optimal path planning. When the experimental and simulation results based on the conventional Dijkstra’s algorithm and proposed modified algorithm are compared, it is observed that the proposed Modified Dijkstra’s algorithm has exhibited a little bit better performance in terms of navigation path planning and estimation time. Also, when the tests made in Gazebo environment are considered, it was seen that the proposed algorithm has yielded 1% to 3% shorter path distances compared to Dijkstra’s Shortest Path Algorithm. It is certain that more experiments should be done, considering many more scenarios, to obtain a generalized statement about the efficiency of the proposed method. However, since the path planning capability and the velocity of decision-making process have increased, the hardware components such as sensors and actuators, which have more accurate responses in shorter reaction times, are recommended to be provided. In the future works, it is aimed to implement the fractional order control methods or model predictive control methods in order to improve the navigation path planning of the AMR for the robots which have fast decision-making processes. Also, it is planned to carry out a new study regarding the development of the proposed algorithm and its comparison with the A* and D* algorithms on a more advanced robot.

Acknowledgement

This work was supported by Inonu University, Scientific Research Projects Coordination Unit (BAP) [Grant number FDK-2022-2777].

Author Contributions

Orkan Murat Çelik: Prepared the experimental environment, conducted the experiments and simulations, analysed the received data and contributed to all stages of the article.

Murat Köseoğlu: Designed the study, performed analyses, contributed to all stages of the article as the supervisor.

Conflicts of Interest

The authors declare no conflict of interest.

References

Arkin, R. C., & Murphy, R. R. (1990). Autonomous navigation in a manufacturing environment. *IEEE Transactions on Robotics and Automation*, *6*(4), 445–454. https://doi.org/10.1109/70.59355

Ben-Ari Mordechaiand Mondada, F. (2018). Robots and Their Applications . In *Elements of Robotics* (pp. 1–20). Springer International Publishing. https://doi.org/10.1007/978-3-319-62533-1_1

Carlucho, I., De Paula, M., & Acosta, G. G. (2019). Double Q-PID algorithm for mobile robot control. *Expert Systems with Applications*, *137*, 292–307. https://doi.org/https://doi.org/10.1016/j.eswa.2019.06.066

Dijkstra's algorithm. (2021). https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm

Durrant-Whyte, H., & Bailey, T. (2006). Simultaneous localization and mapping: part I. *IEEE Robotics Automation Magazine*, *13*(2), 99–110. https://doi.org/10.1109/MRA.2006.1638022

Fadzli, S. A., Abdulkadir, S. I., Makhtar, M., & Jamal, A. A. (2015). Robotic Indoor Path Planning Using Dijkstra's Algorithm with Multi-Layer Dictionaries. *2015 2nd International Conference on Information Science and Security (ICISS)*, 1–4. https://doi.org/10.1109/ICISSEC.2015.7371031

Guo, C. (2013). A Solution to Best Itinerary Problem Based on Strategy Set under Dijkstra Algorithm. *Applied Mechanics and Materials*, *333–335*, 1442–1445. https://doi.org/10.4028/www.scientific.net/AMM.333-335.1442

Hartomo, K., Ismanto, B., Nugraha, A., Yulianto, S., & Laksono, B. (2019). Searching the shortest route to distribute disaster's logistical assistance using Dijkstra method. *Journal of Physics: Conference Series*, *1402*(7), 77014. https://doi.org/10.1088/1742-6596/1402/7/077014

http://wiki.ros.org/tf. (2021).

Iscan, H., & Tuncel, E. (2022). TurtleBot 3 İle Ros Tabanlı Yol Planlama Uygulaması. European Journal of Science and Technology. https://doi.org/10.31590/ejosat.1081097

Köseoğlu, M., Çelik, O. M., & Pektaş, Ö. (2017). Design of an autonomous mobile robot based on ROS. *2017 International Artificial Intelligence and Data Processing Symposium (IDAP)*, 1–5. https://doi.org/10.1109/IDAP.2017.8090199

Kumar, A., & Gao, N. (2021). *Optimization of Distributıon Route Using Dijkstra's Based Greedy Algorithm : Case of Retaıl Chain*. *6*(8), 186–190.

Macenski, S., Martin, F., White, R., & Clavero, J. G. (2020). The Marathon 2: A Navigation System. 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2718–2725. https://doi.org/10.1109/IROS45743.2020.9341207

Merzlyakov, A., & Macenski, S. (2021). A Comparison of Modern General-Purpose Visual SLAM Approaches. 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).

Nguyen, T. Van, Do, M. H., & Jo, J. (2022). MoDeT: a low-cost obstacle tracker for self-driving mobile robot navigation using 2D-laser scan. Industrial Robot, ahead-of-p(ahead-of-print). https://doi.org/10.1108/IR-12-2021-0289/FULL/PDF

Ochiai, Y., Takemura, K., Ikeda, A., Takamatsu, J., & Ogasawara, T. (2014). Remote control system for multiple mobile robots using touch panel interface and autonomous mobility. *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 3272–3277. https://doi.org/10.1109/IROS.2014.6943017

Pagani, P., Colling, D., & Furmans, K. (2018). A Neural Network-Based Algorithm with Genetic Training for a Combined Job and Energy Management for AGVs. Logistics Journal : Proceedings, 2018(01). https://doi.org/10.2195/lj_Proc_pagani_en_201811_01

Panigrahi, P. K., & Bisoy, S. K. (2021). Localization strategies for autonomous mobile robots: A review. *Journal of King Saud University - Computer and Information Sciences*.

https://doi.org/https://doi.org/10.1016/j.jksuci.2021.02.015

Siegwart, R., Nourbakhsh, I. R., & Scaramuzza, D. (2011). *Introduction to Autonomous Mobile Robots* (Second Edi, Vol. 5). The MIT Press.

Tang SH, K. F., & Zulkifli N, K. W. (2015). A Review on Motion Planning and Obstacle Avoidance Approaches in Dynamic Environments. Advances in Robotics & Automation, 04(02). https://doi.org/10.4172/2168-9695.1000134

Tsai, C.-C. (1998). A localization system of a mobile robot by fusing dead-reckoning and ultrasonic measurements. *IEEE Transactions on Instrumentation and Measurement*, *47*(5), 1399–1404. https://doi.org/10.1109/19.746618

Wolf, D. F., & Sukhatme, G. S. (2005). Mobile Robot Simultaneous Localization and Mapping in Dynamic Environments. *Autonomous Robots*, *19*(1), 53–65. https://doi.org/10.1007/s10514-005-0606-4

Zaman, S., Slany, W., & Steinbauer, G. (2011). ROS-based mapping, localization and autonomous navigation using a Pioneer 3-DX robot and their relevant issues. *2011 Saudi International Electronics, Communications and Photonics Conference (SIECPC)*, 1–5. https://doi.org/10.1109/SIECPC.2011.5876943