EMITTER International Journal of Engineering Technology
Vol. 9, No. 2, December 2021, pp. 222~238
DOI : 10.24003/emitter.v9i2.614

Optimized Graph Search Algorithms for Exploration with Mobile Robot

Aydın GÜLLÜ[1], Hilmi KUŞÇU[2]

[1]Electronics and Automation Department, Ipsala Vocational School, Trakya University, Edirne, Turkey
[2]Department of Mechanical Engineering, Faculty of Engineering, Trakya University, Edirne, Turkey
E-Mail: hilmi@trakya.edu.tr
Correspondence Author: aydingullu@trakya.edu.tr

Received August 8, 2021; Revised September 11, 2021; Accepted October 15, 2021

Abstract

Graph search algorithms and shortest path algorithms, designed to allow real mobile robots to search unknown environments, are typically run in a hybrid manner, which results in the fast exploration of an entire environment using the shortest path. In this study, a mobile robot explored an unknown environment using separate depth-first search (DFS) and breadth-first search (BFS) algorithms. Afterward, developed DFS + Dijkstra and BFS + Dijkstra algorithms were run for the same environment. It was observed that the newly developed hybrid algorithm performed the identification using less distance. In experimental studies with real robots, progression with DFS for the first-time discovery of an unknown environment is very efficient for detecting boundaries. After finding the last point with DFS, the shortest route was found with Dijkstra for the robot to reach the previous node. In defining a robot that works in a real environment using DFS algorithm for movement in unknown environments and Dijkstra algorithm in returning, time and path are shortened. The same situation was tested with BFS and the results were examined. However, DFS + Dijkstra was found to be the best algorithm in field scanning with real robots. With the hybrid algorithm developed, it is possible to scan the area with real autonomous robots in a shorter time. In this study, field scanning was optimized using hybrid algorithms known.

Keywords: Optimized graph search, Real environment exploration, Dijkstra, Mobile Robot

1. INTRODUCTION

Intelligent robots have been developed with sensor and software technology and are expected to operate autonomously; they should generate solutions by making their own decisions. The goal of this study was to enable

an autonomous mobile robot to identify an unknown environment. One of the parameters required in the scanning of a real field is that the robot reaches a result in a short time by traveling a small distance. Since the 1950s, researchers have been seeking a solution to the shortest path problem [1-3]. The robot begins at a starting point and moves toward the next intersection point. It identifies the environment by recording the distance travelled and the direction of the route [4]. A map of the environment can be recorded through various structures, such as graphs, geometrics, and partial maps [5, 6]. For this study, the environment was recorded in graph format. In addition, a 2D environment map was drawn with the developed computer interface.

Different algorithms are used to define environments. Wall tracing algorithms can be used for this process. Despite this being a simple concept, solutions are impossible in complex environments [7]. Wall tracing algorithms have been used along with other algorithms to find solutions [8-10]. Graph search algorithms provide more efficient results for exploration of an environment. In a study by [11], environment identification was performed using a depth-first search (DFS) algorithm. This work was developed for the identification of a simple graph-type structure. Many studies [12-14] have comparatively examined DFS and BFS search algorithms. In a study by [15], graph search and Dijkstra algorithms were used in combination to find the shortest and quickest path. In a study by [16] to explore the trajectory planning of autonomous robots, algorithm operation time was taken into account. [17] considered the importance of the travel distances of real robots.

In the present study, a Dijkstra algorithm was run concurrently with graph search algorithms for environment exploration by real mobile robots. This enabled the robot to identify the environment in a shorter time and optimized the results of the graph search algorithms.

Section 2 provides the definitions of the algorithms used. In Section 3, the problem is explained. The hardware and software developed for the real test environment are described in Section 4. The proposed method is practically tested and the results are provided in Section 5. Section 6 presents the study evaluation, and suggestions for future work are discussed.

2. ALGORITHM OVERVIEW

BFS and DFS algorithms that were adapted to allow a real mobile robot to move autonomously were used in this study.

In the breadth-first search (BFS) algorithm, nodes are progressively scanned from top to bottom according to a graph structure. Transition to the next stage is not possible until all nodes in one stage are complete [18]. When a BFS algorithm is used for a real mobile robot, the robot passes the same nodes more than once during the search.

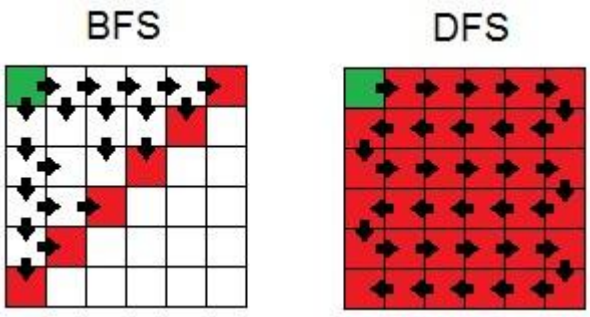


Figure 1. Processing with Graphical Search Algorithms

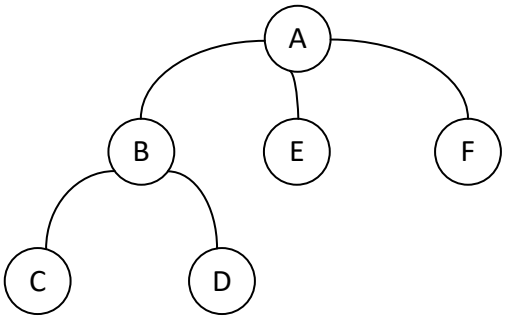


Figure 2. Sample Graph-Type Data Structure

In the DFS algorithm, the robot proceeds from the starting point through to the last node. When the robot reaches a node, it continues searching by returning to the previous node [19]. Using these two algorithms, unknown nodes are recorded. The operation of DFS and BFS algorithms is illustrated in Figure 1. In the figure, BFS is scanning horizontally. DFS is a vertical scanning method. With DFS, it progresses to the last point and then returns to the previous node. With BFS, all nodes are visited horizontally. When the BFS algorithm is operated, as shown in Figure 2, the results are A, B, E, F, C, D. When both algorithms are run, the results defined are A, B, C, D, E, F. The robot will separately work with these two search algorithms to explore the entire unknown environment.

In cases in which the identified area has a simple tree structure, no optimization is required for the final identification. In complex graphs, nodes are connected with each other and there are multiple ways to go from one node to another. During calculation on a computer, there is no path cost. It is possible to pass from one node to another simply by changing the address information. However, in a real environment, there must find path cost to travel from one node to another. For this reason, in this study, the Dijkstra shortest path algorithm is used to go a short way between known nodes. This algorithm, developed by Edsger W. Dijkstra in 1956, provides the shortest way to go from one node to another node in a graph environment [20]. When the Dijkstra algorithm is applied to the data structure created on the software side,

each found node V (vertex) is formulated according to the distance between the node's E (edge) with a mathematical "Big O Notation" as follows:

$$O(V^2) \quad (1)$$

It is executed as

$$O(V+E) \quad (2)$$

to find the shortest path by searching in cycles; the operation is executed in interwoven cycles and the number of operations decreases with time.

Thus, it becomes

$$O(\log V) . \quad (3)$$

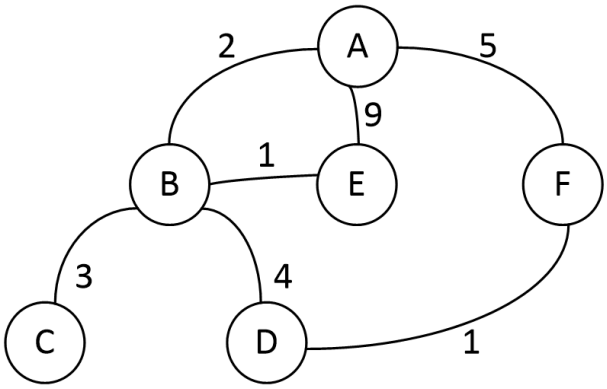
All these transformations together are shown as

$$O((V+E)\log V)=O(E \log V). \quad (4)$$

During the problem, the start and end points of the shortest path change dynamically. In equation 4, this is shown as a time complexity.

3. PROBLEM DESCRIPTION

Various search algorithms are used to identify unknown environments. Mobile robots travel around, exploring the environment. BFS and DFS algorithms are used for exploration. Each new intersection point is recorded with the address of that location. This intersection point is called a node; the distance from one node to another is called an edge. All nodes and edges are saved as a graph structure and later used for search activities.



BFS Path: A ▸ B ▸ E ▸ F ▸ C ▸ D

Figure 3. Sample Graph Data for an Optimized BFS

In the identification of an environment such as the type of structure shown in Figure 3, when the BFS algorithm is executed, there is a move from

the start node A to node B. Afterward, nodes A, E, A, and F are traversed sequentially and the first width level is completed. For the second level, there is a move from node F to node B, after which, nodes B, C, B, D, B, and E are traversed sequentially. Despite the completion of the second level connected to node B, it is necessary to scan the second level of the width connected to node F. For this reason, the robot must go from the E-node to the F-node. Nodes and path costs are given in the form of a graph. The robot can select route B, A, F or route A, F to reach node F. It must select the shortest path between the two choices. The algorithm developed in this study selects the shortest route among those detected. The Dijkstra algorithm is used for this purpose. The map in the graph structure is transformed into a neighbourhood matrix, the lowest cost route is determined, and the robot is provided with this route. In this structure, although the robot travels more nodes, route B, A, F is the shortest distance. With this hybrid work, the distance and time required for exploration have been reduced.

Another graph structure example is given in Figure 4. If exploration is made with this DFS algorithm to map a graph structure, the robot moved from start node A to the final node as A, B, C. After node C, it must return to the last node B and follow nodes D, E, D, F, H, and A. In the same way, after node A, it must return to node H. A, H is farther than A, B, D, F, H. The determination of this route was calculated with the Dijkstra algorithm and the route information transferred to the robot.

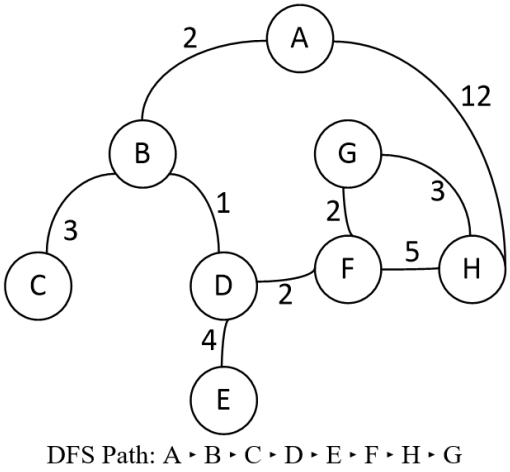


Figure 4. Sample Graph Data for an Optimized DFS

4. PRACTICAL APPLICATION

We developed mobile robot hardware, mobile robot software, and computer software for practical applications. The software was developed to work seamlessly with the entire system. Also a line maze was chosen as its environment, and an interface with C # software was developed to run the robot algorithms and draw a 2D map of the environment.

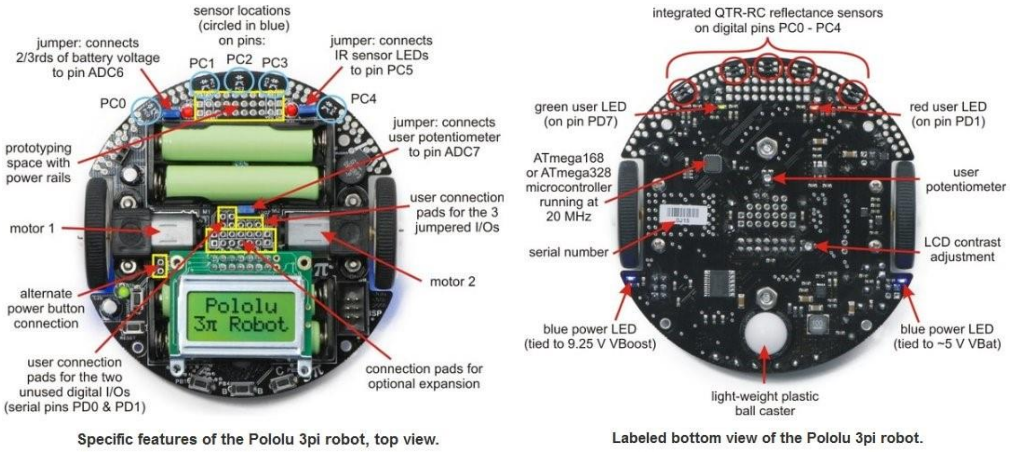


Figure 5. The Pololu 3Pi Mobile Robot

4.1 MOBILE ROBOT

The Pololu 3pi robot was used to test the algorithms used to define the environment. This robot has a small structure (9.5 cm in diameter) and carries five line detection sensors, which allows the robot to successfully track the line [21,22]. There are serial communication ports and additional digital inputs and outputs on the robot. An electronic card is built into the robot to communicate with the sub-circuit using the communication ports. This electronic card, as well as additional sensors, allows the robot to communicate with the computer; information from the robot can be transferred instantaneously to the computer. The robot used in Figure 5 and the developed electronic card are shown in Figure 6. While the bottom layer of the robot performs the line tracking functions, the upper layer performs the decision and communication functions.

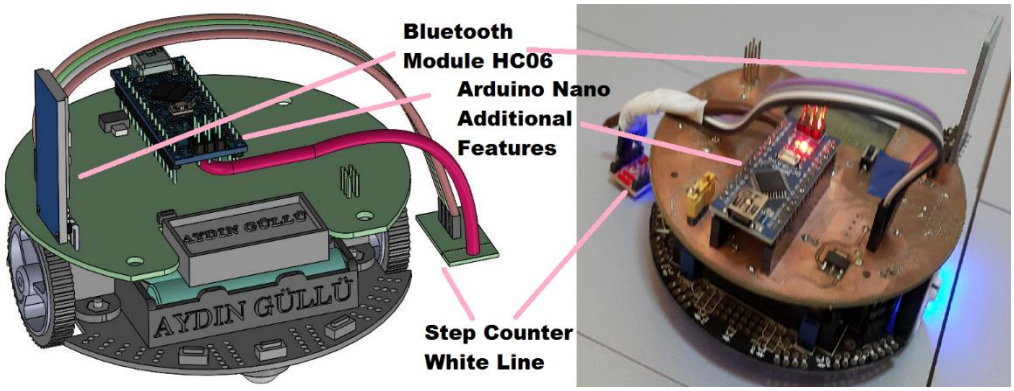


Figure 6. Pololu 3Pi and Additional Printed Circuit Board (PCB)

In order for the robot to detect the environment, sensors compatible with the environment should be used. As the environment in this study is a line maze, it will be sufficient for the robot to only detect the line. If there was a corridor maze or an obstacle environment, there might be a need for other

sensors that report depth, distance, and direction. In the present case, the robot obtains two groups of information from the environment. With the first group of information, the robot moves by itself. There is no choice of direction or decision.

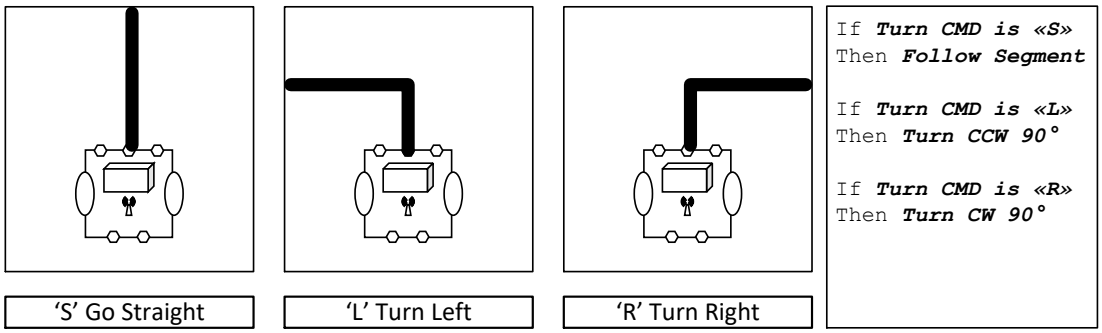


Figure 7. First Group Environmental Information and Pseudocode

In the first group, the information received from the medium is single commanded, such as left (L), right (R), or straight (S). These commands will be recorded step by step and operated. When the first group of information arrives, the code shown in Figure is run.

The second group is the information that enables the robot to choose its direction and make decisions. The place where the second group of information is located is also a node for software. The distance between the two node points is determined by the data perceived and recorded in the first group. The first group of data contain the edge information; the second group of data is perceived as a combination of several commands. For example, it can be in the form of L-R, L-S, R-S, L,S,R, or B (the last node, indicating return). A selection of turning direction is required between the nodes except "B".

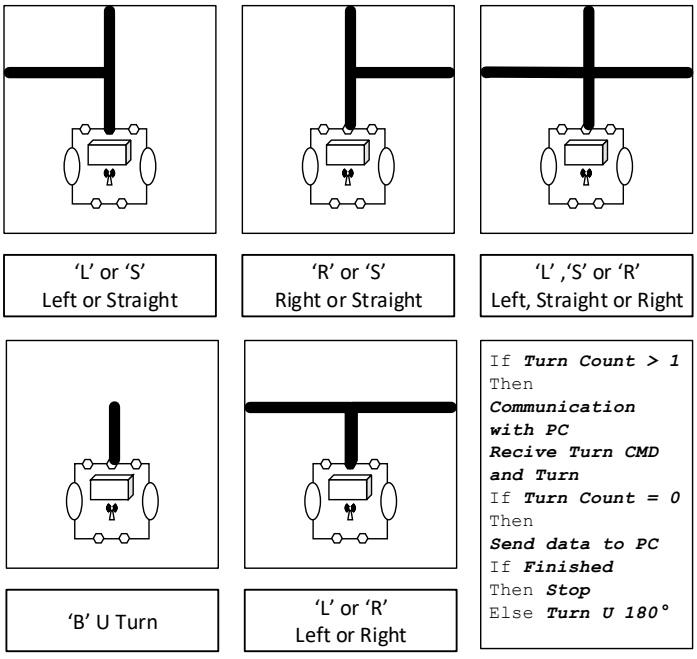


Figure 8. Second Group Environmental Information and Pseudocode

When the above-mentioned second group information is read by the mobile robot sensors, the code shown in Figure will operate. At this stage, the robot travels until it encounters any node and the distance is recorded.

When the nodes are detected, the address of the node is sent to the computer with the number of directions in the node, as is the distance from the previous node to the current node. According to the direction command received from the computer, the robot must rotate 90° or 180° around its own axis. There are two active wheels connected to the DC motor for robot rotation. An independent wheel is used for balance. If differential driving is applied to the kinematic model of a robot with this structure, the turning motion is obtained from the following equations 5-11.

In the differential drive mechanism, the robot can move in two planes—x and y. It can also rotate between these coordinates. This indicates that the mobile robot has three angles of freedom, which are expressed by the matrix in equation 4.1.

$$q = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \tag{5}$$

In this structure, the x and y planes indicate a θ rotation angle. The motors must move at different speeds in order to rotate. In addition to the rotation speeds of the motors, the wheel diameters are also important. Equations 6 and 7 are indicate the relationship of the motor speeds.

$$v_R = r\ \omega_R \tag{6}$$

$$v_L = r\ \omega_L \tag{7}$$

Here, r is the radius of the wheel and ω is the angular velocity of the respective motor. The motion and angular motion of the robot from independent wheel movements are shown in equations 8 and 9. In the angular velocity formula, r_c is the radius of the mobile robot, which is half the distance between the two wheels.

$$v(t) = \frac{v_R(t) + v_L(t)}{2} \tag{8}$$

$$\omega(t) = \frac{v_R(t) - v_L(t)}{r_c} \tag{9}$$

The change in the x, y coordinates according to the resultant linear velocity of the mobile robot and the resulting angle is shown in equation 10-11.

$$\dot{x} = v \cos\theta \tag{10}$$

$$\dot{y} = v\ \sin\theta \tag{11}$$

Here, $\dot{q}$ is the state variable that is a derivative of the vector. It consists of linear and angular velocity changes in the x and y axes [23-25].

$$\begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{\theta}(t) \end{bmatrix} = \dot{q} = \begin{bmatrix} cos\,\theta & 0 \\ sin\,\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v(t) \\ \omega(t) \end{bmatrix} = j(\theta)v \qquad (12)$$

Differential driving is only used to track the line in turns. The proportional–integral–derivative (PID) controller is applied the robot to follow the line properly and effectively in the physical environment. The error information for the PID controller is a read analogue data from the line sensors. The block diagram of the system applied to the mobile robot are as follows fig 9. In this way, the robot is able to steadily follow the line and turn around. Speed change is applied to the motors by pulse width modulation so that the speeds of the motors are controlled independently.

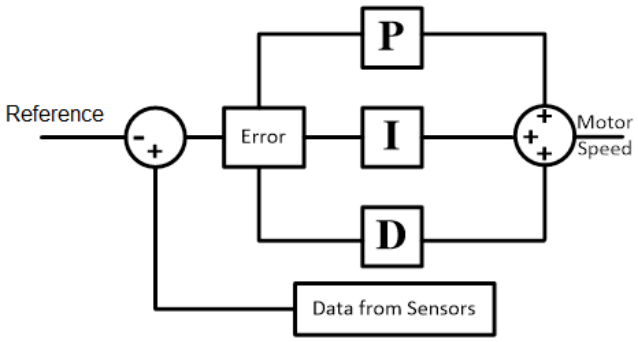


Figure 9. PID Control Block Diagram

4.2 PHYSICAL TESTING ENVIRONMENT

As a physical test medium, 8 mm thick and 25 x 12 mm size chips were used. These chips were laid on the floor. One chip represents two cells. The environment was drawn on the chip in the desired size and structure with an electric band. The sensors on the mobile robot allow it to move following the line. For this reason, a line labyrinth was chosen for the test environment. The physical testing environment is depicted in Figure.

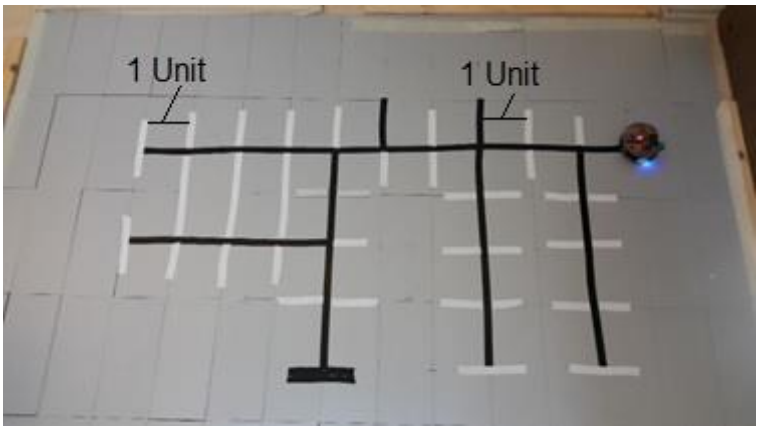


Figure 10. Physical Testing Environment

4.3 COMPUTER SOFTWARE

Software was developed to communicate with the mobile robot that receives the information from the environment and sends it computer. The software development environment used was C #. The robot will start to define the environment from a starting point. When it comes to a new node, it sends the route it recorded during the move and the properties of the new node it found to the computer. Computer communication was provided via Bluetooth. The computer evaluates the information that is transmitted wirelessly and guides the robot according to the selected algorithm. If a node has already been added, it is not saved. In the same way, path information between nodes is recorded with information, such as route information, from one node to another. All these data are stored in a graph structure. The saved graph structure is given in Figure.

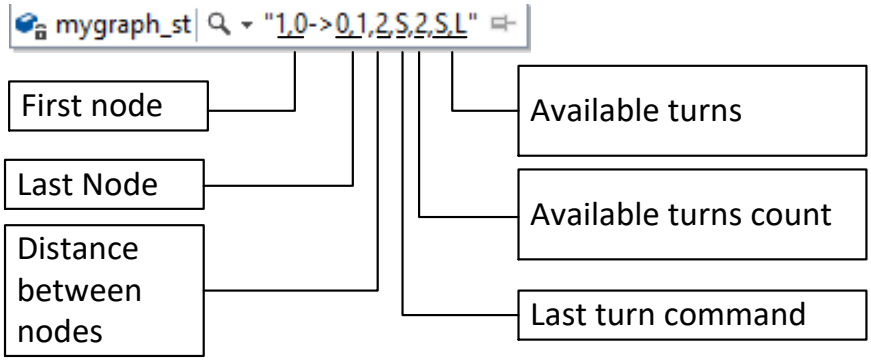
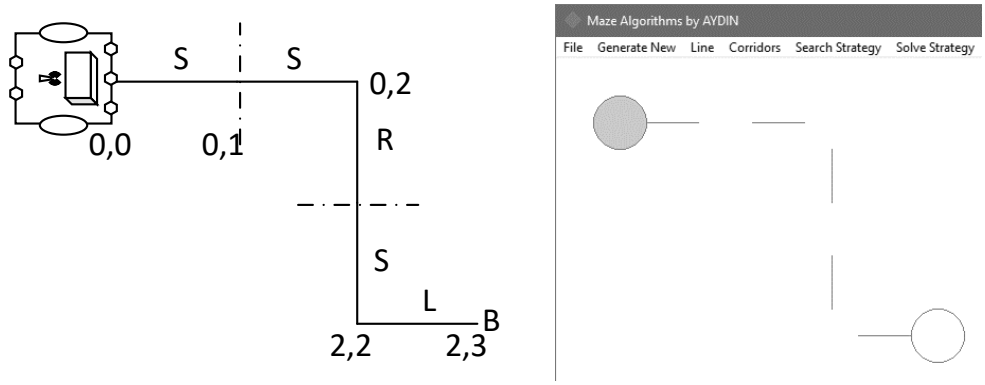


Figure 11. Generated Graph Structure

If there is movement between previously known nodes, Dijkstra algorithm is run. In this way, the shortest path is found and quick exploration is ensured. The software developed on the computer can also draw a map of the environment. The developed software interface is given in Figureb. The robot receives cell information from the field. When it starts to move from a starting point, the S command indicates straight, the L command indicates left, the R command indicates right, and the B command indicates return. When the robot moves from one node to another, the first group information is recorded. If this information is S→S →R→S→L, the route information in Figurea is drawn on the computer. At the same time, a path cost is incurred. In this example, the path is 5 units, which means that a 5 x 12 = 60 cm path was recorded.



(a) (b)

Figure 12. Simple Robot Motion and Plotting on the Computer

This distance is plotted as in Figureb with the specified scale in the computer. Likewise, the node points are indicated as circular. If this is a node point and we assume the address starting point as 0,0, the address of the destination node is 2.3.

5 EXPERIMENTAL RESULTS

Optimized graph scanning algorithms were tested with a real mobile robot in four different environments, as indicated in Figure, 15 and 16. For comparative purposes, three different environment DFS and BFS algorithms were defined separately. Test mazes are small in size because of the boundary of the physical environment. But the search processes in the environment seem to be shortened for both algorithms. If testing is done in larger and complex environments, optimization will work better. Maze 3 and 4, the largest test environment in the tests performed, shows the success of the optimization results better.

In the figures shown (figures 13 to 16), the environment transferred to the computer. On the computerized map, the blue dot shows the robot's position, and the hollow points show the nodes. The places where the robot starts and ends (return) are also considered as nodes. Nodes are decision points for which direction to choose in directions.

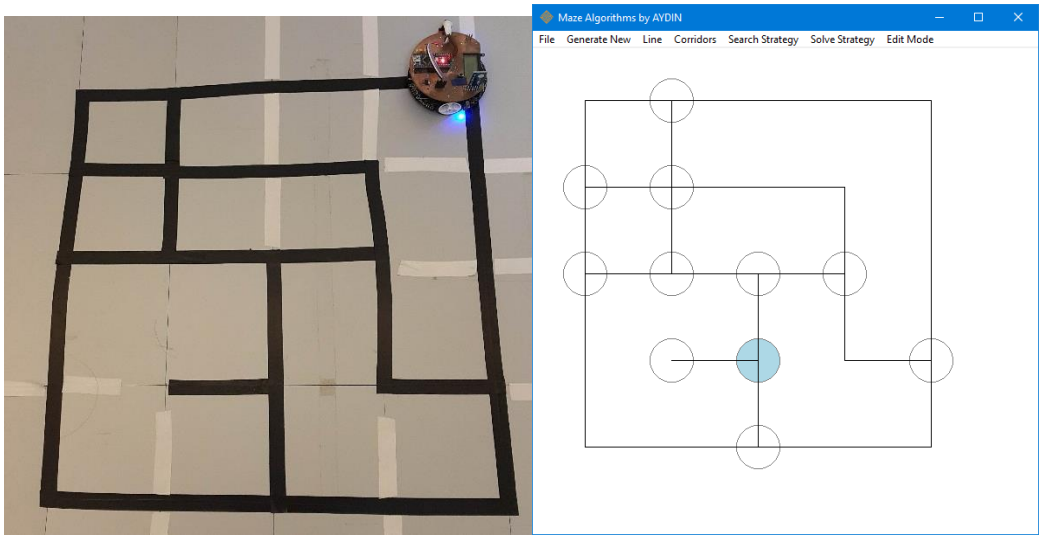


Figure 13. Test Maze 1 and Its Transfer to Computers

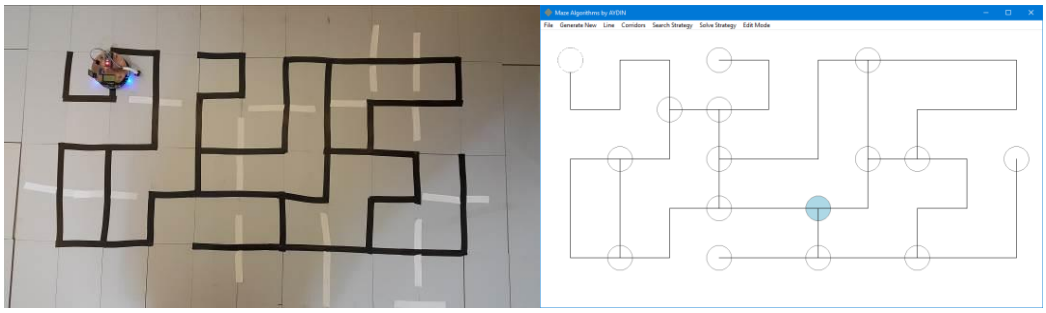


Figure 14. Test Maze 2 and Its Transfer to Computers

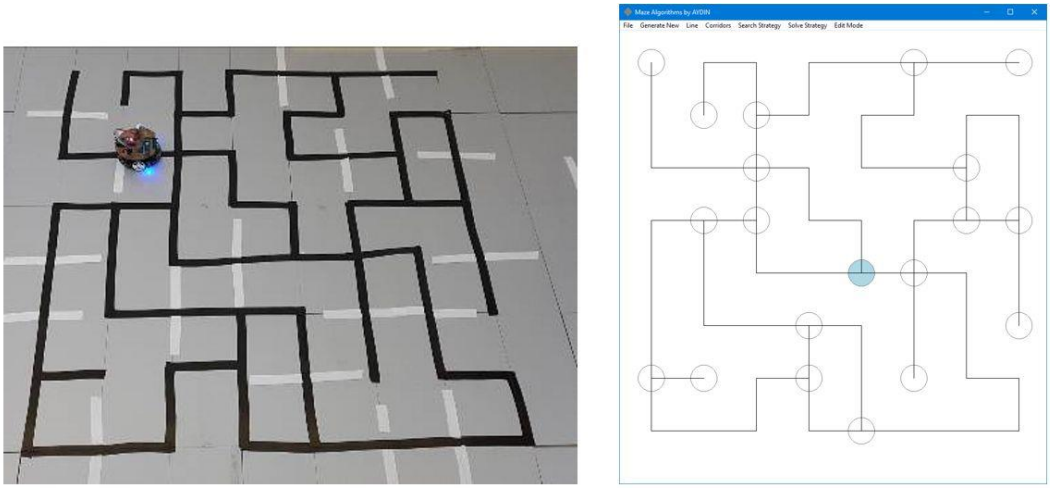


Figure 15. Test Maze 3 and Its Transfer to Computers

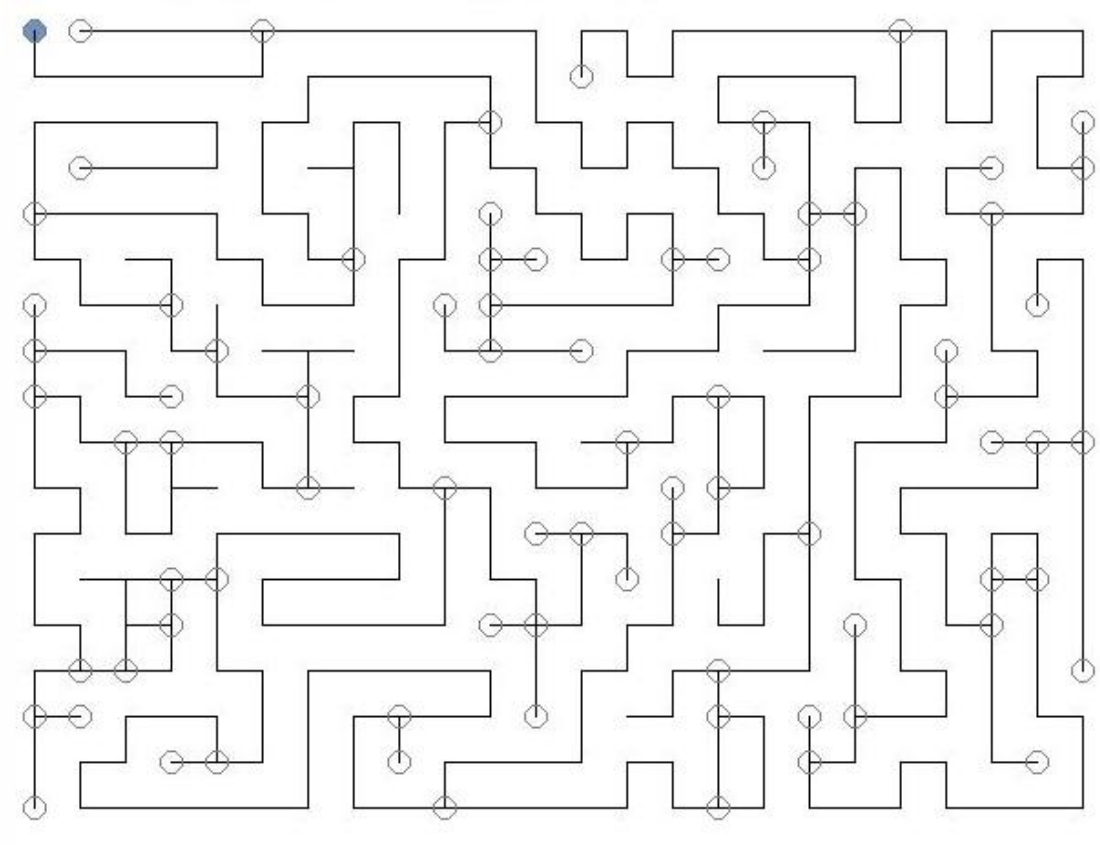


Figure 16. Test Maze 4's Transfer to Computers

DFS + Dijkstra and BFS + Dijkstra algorithms were operated and the results examined. The path costs of the robot for scanning each entire environment were recorded. The results are shown in Table 1. In the experimental study, each unit of distance was taken at the same time. For these reasons, unit step can be evaluated as unit time at the same time. The least number of steps gives the shortest distance.

Table 1. Algorithm Results as Total Unit Step Distance

	BFS	**BFS + DIJKSTRA**	**DFS**	**DFS + DIJKSTRA**
MAZE 1	90 Step	81 Step	39 Step	37 Step
MAZE 2	169 Step	137 Step	76 Step	72 Step
MAZE 3	292 Step	233 Step	112 Step	94 step
MAZE 4	958 Step	785 Step	654 Step	129 step

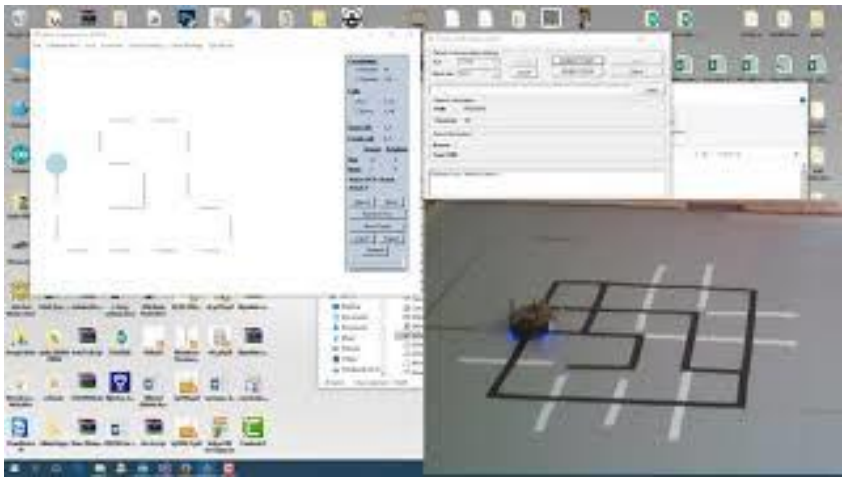
6.CONCLUSION AND FUTURE WORK

In this study, a mobile robot was used to identify real environments. The robot used graph search algorithms to navigate and map unknown environments. When identifying a real environment, every nodes in environment, the directions from each node and the distance between the nodes are appropriately obtained. By searching according to an algorithm, the robot travelled between nodes and the newly discovered nodes were recorded. The movement between the known nodes was provided by the route determined by the Dijkstra algorithm. With the code that we developed to operate with DFS and Dijkstra algorithms, integrated exploration was ensured with less distance travelled than with the DFS algorithm alone. Similar results were obtained with the BFS algorithm; BFS + Dijkstra gave better results because the width search was done in the BFS algorithm. Because BFS does a horizontal scan, it takes a lot more distance and a lot of time to explore a real environment. Instead of scanning the entire environment with BFS, slightly better results were obtained with new node detection with BFS and returning with Dijkstra. However, as shown in table 1, the best result was obtained with the DFS + Dijkstra algorithm. The increase in the number of nodes caused the Dijkstra algorithm to work better. Maze four is an example of this.

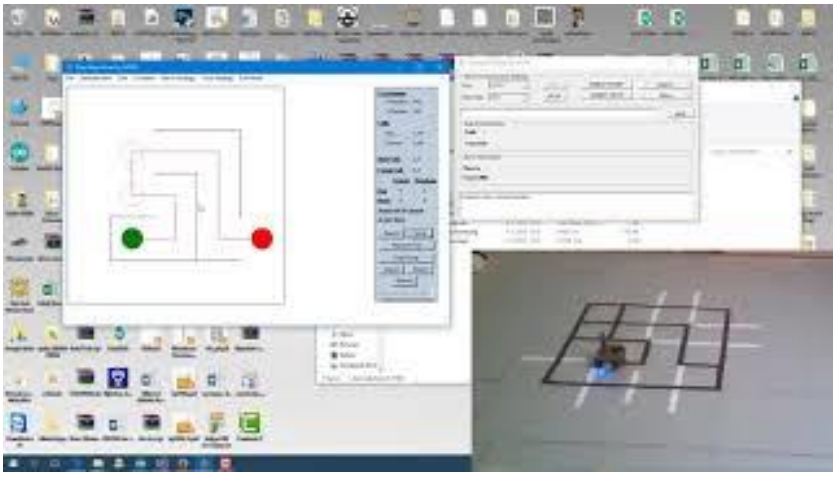
When searching in the real world, the robot must travel a certain distance to arrive at one node from another. The shorter this path, the shorter the total time and distance required for exploring the real environment. The developed hybrid algorithm was successfully tested in a real test environment. Among the tested algorithms, the algorithm that performs a search with the real robot in the shortest time and at the shortest distance is the DFS + Dijkstra application. In this study, horizontal scanning algorithms have covered much more time and distance in real environments. During the discovery of new paths with DFS, Dijikstra was preferred in choosing the shortest path for the running algorithm to go to known nodes. In this way, the discovery was made in a hybrid way. DFS will be able to scan the entire environment, eliminating the disadvantages of wall tracking algorithms used in real robots, such as looping. In more complex environments, more efficient results could be obtained with perception algorithms such as A * or GBF instead of Dijkstra. In this way, scanning can be done by going to the node point in a shorter way and in a shorter time. In daily life, our proposed algorithm can be used in the field survey of autonomous vacuum cleaners, cleaning robots, and lawn mowers. In the future, it is planned to apply these studies to a mobile robot working in 3D real environment. For this purpose, it is planned to change the mechanical and sensor structure of the robot. The discovery of the real environment of 3D environment is planned to be provided by these developed algorithms. It is also aimed to use these algorithms in multi-robot discovery applications.

7. SUPPLEMENT DATA

In this study, algorithms are obtained by testing with a real robot for a real environment. An interface was coded for communication with the robot and wireless data was collected via Bluetooth. Mobile robot and environment have been selected as limited since only algorithm tests have been conducted. It does not contain a robot encoder. It only detects the line with optical sensors. In real tests, the line is maze-based. However, it has been observed that the complex tests carried out in computer environment by simulations are compatible with the tests performed with limited environment. In the tests, it enabled a real mobile robot to travel the search time and the shortest distance in the search process. The developed test platform is shown in the video link below.



https://youtu.be/ZlcunRXh5BE



https://youtu.be/5_kUUGt0Blk

REFERENCES

[1] Brass, P., I. Vigan, and N. Xu, **Shortest path planning for a tethered robot.** Computational Geometry, 2015. 48(9): p. 732-742.

[2] Moore, E.F., ***The shortest path through a maze***. 1959*: Bell Telephone System.*

[3] Zhan, F.B. and C.E. Noon, **Shortest path algorithms: an evaluation using real road networks.** Transportation Science, 1998. 32(1): p. 65-73.

[4] Zelinsky, A., **A Mobile Robot Exploration Algorithm.** Ieee Transactions on Robotics and Automation, 1992. 8(6): p. 707-717.

[5] Adorni, G., et al., **Vision-based localization for mobile robots.** Robotics and Autonomous Systems, 2001. 36(2): p. 103-119.

[6] Nüchter, A., et al., **6D SLAM—3D mapping outdoor environments.** Journal of Field Robotics, 2007. 24(8-9): p. 699-722.

[7] del Rosario, J.R.B., et al. ***Modelling and Characterization of a Maze-Solving Mobile Robot Using Wall Follower Algorithm*.** in *Applied Mechanics and Materials*. 2014. Trans Tech Publ.

[8] Cai, Z., L. Ye, and A. Yang. ***FloodFill Maze Solving with Expected Toll of Penetrating Unknown Walls for Micromouse*.** in *High Performance Computing and Communication & 2012 IEEE 9th International Conference on Embedded Software and Systems (HPCC-ICESS), 2012 IEEE 14th International Conference on*. 2012. IEEE.

[9] Dain, R.A., **Developing mobile robot wall-following algorithms using genetic programming.** Applied Intelligence, 1998. 8(1): p. 33-41.

[10] Hanafi, D., Y.M. Abueejela, and M.F. Zakaria, **Wall follower autonomous robot development applying fuzzy incremental controller.** Intelligent Control and Automation, 2013. 4(01): p. 18.

[11] Kwek, S. ***On a simple depth-first search strategy for exploring unknown graphs*.** in *Workshop on Algorithms and Data Structures*. 1997. Springer.

[12] Everitt, T. and M. Hutter, ***Analytical Results on the BFS vs. DFS Algorithm Selection Problem. Part I: Tree Search*, in *AI 2015: Advances in Artificial Intelligence***. 2015, *Springer*. p. 157-165.

[13] Everitt, T. and M. Hutter, ***Analytical Results on the BFS vs. DFS Algorithm Selection Problem: Part II: Graph Search*, in *AI 2015: Advances in Artificial Intelligence***. 2015, *Springer*. p. 166-178.

[14] Everitt, T. and M. Hutter, **A topological approach to meta-heuristics: analytical results on the BFS vs. DFS algorithm selection problem.** arXiv preprint arXiv:1509.02709, 2015.

[15] Potamias, M., et al. ***Fast shortest path distance estimation in large networks*.** in *Proceedings of the 18th ACM conference on Information and knowledge management*. 2009. ACM.

[16] Bihlmaier, A., L. Winkler, and H. Worn. ***Automated planning as a new approach for the self-reconfiguration of mobile modular robots*.** in *Robot Motion and Control (RoMoCo), 2013 9th Workshop on*. 2013. IEEE.
[17] Ryu, H. and W.K. Chung, **Local map-based exploration using a breadth-first search algorithm for mobile robots.** International Journal of Precision Engineering and Manufacturing, 2015. 16(10): p. 2073-2080.
[18] Subramanian, M.B., K. Sudhagar, and G. RajaRajeswari, **Optimal Path Forecasting of an Autonomous Mobile Robot Agent Using Breadth First Search Algorithm.** 2014.
[19] Tarjan, R., **Depth-first search and linear graph algorithms.** SIAM journal on computing, 1972. 1(2): p. 146-160.
[20] Dijkstra, E.W., **A note on two problems in connexion with graphs.** Numerische mathematik, 1959. 1(1): p. 269-271.
[21] Electronics, P.R.a. **Pololu 3pi Robot.** 2016 [cited 2016 02.01.16]; Available from: https://www.pololu.com/product/975.
[22] Costa, H., et al. ***A mixed reality game using 3Pi robots—"PiTanks".*** in *Information Systems and Technologies (CISTI), 2015 10th Iberian Conference on*. 2015. IEEE.
[23] Ribeiro, M.I. and P. Lima, **Kinematics models of mobile robots.** Instituto de Sistemas e Robotica, 2002: p. 1000-1049.
[24] Mireles Jr, J., **Kinematic models of mobile robots.** Automation and Robotics Research Institute, University of Texas at Austin, 2004.
[25] Siegwart, R., I.R. Nourbakhsh, and D. Scaramuzza, ***Introduction to autonomous mobile robots***. 2011*: MIT press*.