Performance Comparison Robot Path Finding Uses Flood Fill - Wall Follower Algorithm and Flood Fill - Pledge Algorithm

Semuil Tjiharjadi
Maranatha Christian University, Bandung, Indonesia
Email: semuiltj@gmail.com

***Abstract*—As a path-finding robot in the maze, the robot must have the ability to decide the direction taken at the intersection inside the maze. Robot will map route and try to reach the destination in the fastest time and shortest distance. Robot will use two algorithms for the pathfinding process, the Wall Follower algorithm, and the Pledge algorithm. Both algorithms can determine the direction in the process of achieving the expected target location. After the robot reach the destination, the robot will return to its starting position. Robot can easily reach its goal by using the Flood Fill method to decide the fastest and shortest route to reach that position now. This research is an analysis of the combination of the Flood Fill method with the Wall Follower algorithm compared to the Flood Fill method with the Pledge algorithm, based on a series of experiments conducted on various maze patterns in the maze. The experimental results show that robots can explore the maze and map it using the Wall Follower algorithm, Pledge algorithm, and a combination of both with the Flood Fill algorithm. Based on the analysis, it was found that the use of the Flood Fill algorithm that works in synergy with the Wall Follower algorithm and the Pledge algorithm, can dramatically increase the effectiveness of target point searches.**

***Index Terms*—pathfinding, flood fill, wall follower, pledge**

I. Introduction

Robot Maze is a robot that is a search robot that can find directions in the maze. Its ability to determine direction independently is the advantage of this robot. The way the robot automatically determines the direction, performs a route mapping, and finally finds the shortest and fastest distance is the goal of applying the search algorithm to the maze robot [1]. Several algorithms have been developed for this purpose, and each algorithm has its advantages and disadvantages [2].

As part of its autonomous ability, the Path Finding Robot uses structured algorithms to control the autonomous navigation it has [3]. In this study, two combinations of algorithms were used to achieve the shortest and fastest target. The two combination algorithms are the Flood Fill algorithm - Wall Follower algorithm as the first combination, while the second combination is the Flood Fill algorithm - Pledge algorithm. Both combinations of algorithms are compared to get the best method and are expected to find new proposals for the development of better search techniques. It is hoped that this comparison will get the best method for autonomous robots to explore the maze. The main task is to find a path to complete the maze in the shortest possible time and use the shortest way. The robot must start navigation from the corner of the maze to the target as quickly as possible [4].

The information that the robot has is the location of the search and target. The initial task is to collect all information about obstacles to reach the target location. In this study, the maze was designed consisting of 25 square cells, with the size of each cell about 18 cm x 18 cm. The cells are designed to form a maze of 5 rows x 5 columns. The initial search position is set in one cell from its angle, and the target location is in the middle of the maze. The search terms are only one cell that is opened to pass. The design of the maze wall size and supporting platforms use the IEEE standard.

II. Literature Review

A. Breadth-First Search

Breadth-First Search is a search algorithm that tries all the possibilities available. Starting from the root node, Breadth-First Search explores all neighboring nodes to find the target node. Breadth-First Search tests all available nodes, so it requires large memory space to store node information and routes that have been made. This algorithm can find a few solutions for the route so that the shortest route can be found. This algorithm is using First In First Out queue, and it will work poorly and consume a lot of memory when finding a target that has a long path.

Although Zhou has shown Breadth-First Search modifications when using the divide-and-conquer solution reconstruction, it can reduce search memory needs. The result is Breadth-First Search to be more efficient than Best-First Search because it requires less memory to prevent regeneration of closed nodes [5].

B. Depth First Search

The Depth First Search is an algorithm for searching

Manuscript received May 22, 2019; revised May 14, 2020.
corresponding author: Semuil Tjiharjadi, semuiltj@gmail.com

doi: 10.18178/ijmerr.9.6.857-864

based on tree data structures that use the Last In First Out queue method. This algorithm is easy to implement. It starts from the root node and tries each path to the end, and then backtracks until it finds an unexplored path, and then re-explores the new path until it finds a target. The search principle that uses this depth requires high computing power. A small increase in a path can result in a runtime increasing exponentially [6].

C. Heuristic Function

Heuristic Function plays a vital role in the optimization problem. It is a function that uses all mapping information to help the search process in the right direction to achieve goals effectively [5].

D. Genetic Algorithm

Genetic algorithm is a machine-learning technique loosely based on the principles of genetic variation and inspired by natural evolution to find approximate optimal solution. Advantages of Genetic algorithm are it solves problem with multiple solutions. But it needs huge input and data. Problems of Genetic algorithm are certain optimization cases cannot be solved due to poorly known fitness function. It is not able to assure constant optimization response times because the entire population is improving [7].

E. A* algorithm

A* is one of the most popular methods for finding the shortest path in the maze area. It develops a combination heuristic approach. This approach is also used by the Best-First-Search (BFS) algorithm and the Dijkstra algorithm. Algorithm A* calculates the costs associated with each used node. Such as the application of BFS, A* will follow its path with the lowest heuristic cost. Both of them require large memory to store information, because all nodes that have been tested must be stored [8].

The A* algorithms can, during searching, judge the movement of the target point by referring heuristic information, it does not need to thumb through the map, so that the calculating complexity is relative simple and effective fast searching can be achieved [9].

F. Flood Fill Algorithm

Flood Fill algorithm that also known as the seed fill algorithm, is an algorithm that determines the area connected to a given node in a multi-dimensional array. This algorithm needs all information of maze and proper planning. It is used widely for robot maze problem [10].

The Flood Fill algorithm offers values to every node that represents the distance of the node from the center. It floods the maze when it reaches a new cell or node. This algorithm requires continue update [11].

G. Wall Follower Algorithm

Wall Follower algorithm is one of the best known and one of the simplest mazes solving algorithms. It starts following passages, and whenever it reaches a junction, always uses the righthand rule or the left-hand rule. It will turn right or left at every junction base on the right- or left-hand rule. Wall Follower is a fast algorithm and uses no extra memory. But this method will not necessarily find the shortest solution, and this algorithm has weakness when the maze is not connected, it can back at the start point of the maze [12].

H. Pledge Algorithm

The Pledge algorithm is designed to solve Wall Follower weakness. It can avoid obstacles and requires an arbitrarily chosen direction to go forward. At the beginning of the algorithm, the Pledge algorithm sets up direction and follows this direction [13]. When an obstacle is met, one hand rule is kept along the obstacle while the angles turned are counted. When the object is facing the original direction again, the solver leaves the obstacle and continues moving in its first direction [14].

III. Hardware Design

This research is tested using a mobile robot. It has robot base construction by miniQ 2WD robot chassis. It is shown in Fig. 1. It has a 122 mm diameter robot chassis, two wheels, a ball caster, and two Direct Current (DC) motors which have gearbox and DC motor bracket.

Fig. 2 is shown a couple of pieces of rotary encoder attached to the DC motor to calculate the rotation of the wheels.

Figure 1. 12WD miniQ robot chassis.

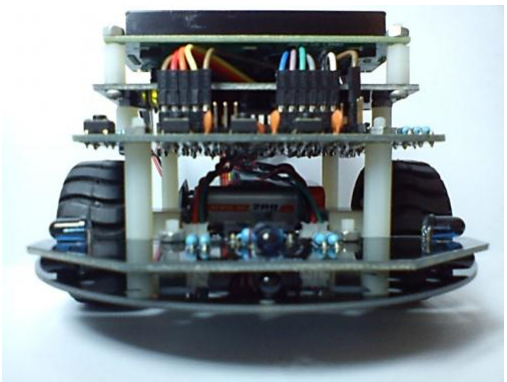

Figure 2. Mobile Robot from side view.

The robot has three infrared sensors to detect the front, right, and left positions of the maze wall. It uses the L293D driver to control the speed and rotation of a DC motor, a rotary encoder that has the task of calculating the rotation of both wheels, and a button to start the robot.

The robotic system will drive a DC motor to drive the wheel. It will control the robot to move forward, turn left or right, and turn backward. AT Mega 324 microcontroller is used to respond to input signals and run actuators based on processing algorithms. All statuses and information are displayed on Liquid Crystal Display (LCD) 16 x 2 in Fig. 3.

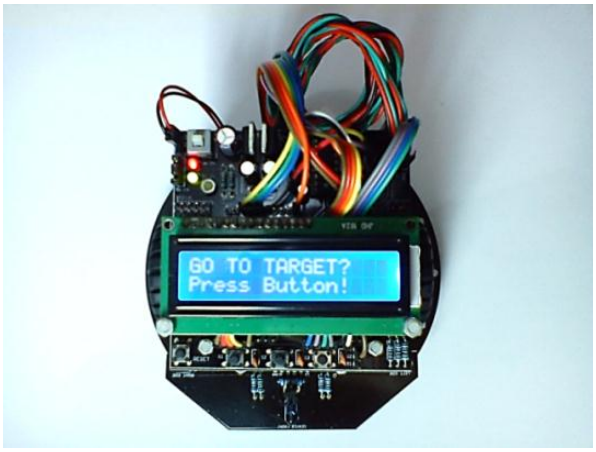

Figure 3. Display of Mobile Robot from the above.

The block diagram of the design of the whole hardware system and the flowchart of the main program can be seen in Fig. 4 and Fig. 5 [15].

The maze designed to be solved by robots is 5 × 5 cells, as shown in Fig. 6. The actual maze that was built, as shown in Fig. 7, has a physical size of about 1.32 m2. The maze is designed so that it will have two paths to complete. A path can be longer than the other, and the robot must decide which path is shorter and complete the maze through that path [16].

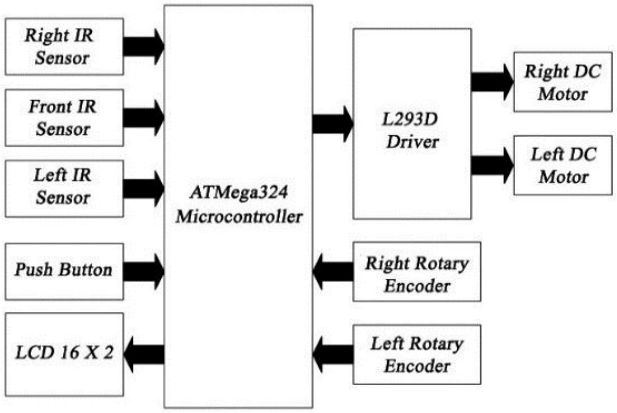

Figure 4. Maze robot's block diagram.

IV. Algorithm

Three types of algorithms were used in this paper. Wall Follower algorithm, Pledge algorithm, and Flood Fill algorithm. The results obtained from the Wall Follower algorithm, Pledge algorithm, Wall Follower combination method - Flood Fill, and Pledge - Flood Fill will then be compared.

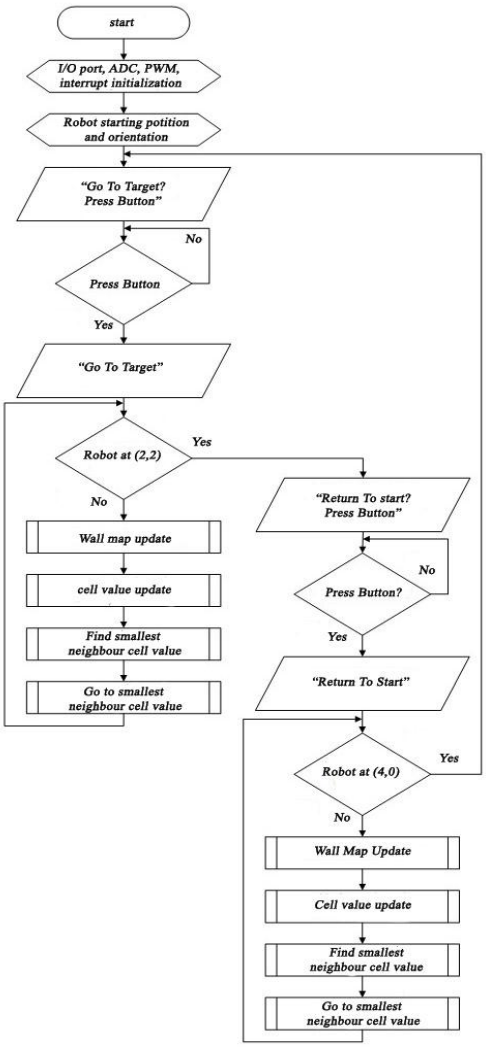

Figure 5. Flowchart of the main program.

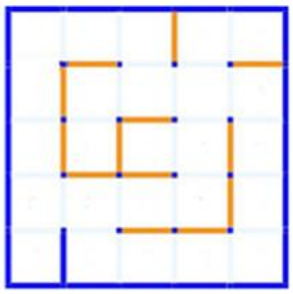

Figure 6. The layout of the maze.

Figure 7. The maze arena.

Together with the Flood Fill algorithm, they are used to find the fastest way to achieve the objectives. Results of Wall Follower algorithm and Pledge algorithm were compared when determining the priority of directions taken when the robot finds the same value of priority based on the Flood Fill algorithm [11]. The Wall Follower algorithm will use the right- or left-hand method in determining the direction to be taken at each intersection. While the Pledge algorithm will assign +1 value to the 'Play' variable every time the robot turns right and the value -1 every time the robot turns left, the goal is to achieve the target by prioritizing the smallest possible 'Turn' variable value. When the Pledge algorithm finds an intersection, the turn decision taken is to reduce the value of the 'Play' variable from rotation. The Wall Follower algorithm and the Pledge algorithm are used to help the Flood Fill algorithm so that collaboration will produce smarter decisions [17].

The Artificial Intelligence program has a two-dimensional array of memory to map the 5x5 maze arena. Memory arrays are used to store information on each maze cell wall and every cell value information. The position of the robot in the program is expressed by coordinates (rows, columns). The robot moves in the array to the location of the robot, as shown in Fig. 8.

The line coordinates will increase 1 when the robot moves one cell to the South. On the other hand, it will decrease by 1 when the robot moves north. The column will reduce by 1 when the robot moves to the West, and it will increase by 1 when the robot moves to the East. Robots already have information about the initial orientation, initial position, maze size, and location of the outer wall of the maze [18].

There are four main steps in Flood Fill algorithm: the first is updating wall data, the second is updating cell values, the third is calculating the smallest neighbor cell, and the last is moving to the smallest neighbor cell.

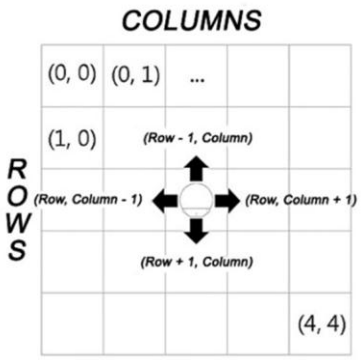

Figure 8. Robot's Array Movement

A. Updating Wall Data

Robot will test its environment, any partitions in its three directions: right, left, and forward instructions. The robot will additionally observe the distance of any obstacles of its three courses. Anything exceed 20 cm is updated as "wall" on its respective side. Flowchart in Fig. 9 describes the wall data update mechanism.

The robot will check the environment, each wall in three directions: right, left, and front. Any obstacles detected exceeding 20 cm will be updated as "walls" on each side. The flow chart in Fig. 9 explains the mechanism for updating wall data.

The maze robot always needs to know the way its faces, so it knows where to go. Table I details the relationship between robot orientation and detection of wall sensors. When it begins at the start, the robot has an initial orientation and will continue to track changes in direction. The robot orientation also determines the left, front, and right positions of the robot, as described in Table I.

TABLE I. ROBOT ORIENTATION AND WALL DETECTION

Robot Orientation	Wall Sensor Detection		
	Right	Front	Left
South	West wall	South wall	East wall
West	North wall	West wall	South wall
North	East wall	North wall	West wall
East	South wall	East wall	North wall

B. Cell Value Update

The update value of cell wall is stored in a 2-dimensional array of 5x5 memory cells. Renewing cell values is done using a Flood Filling algorithm. At the current level array, the cell will be updated. In the next level array, the neighboring cells will be calculated. When the value filling process is complete, the cells in the next level array will be copied to the current level array to do the future value. The update process will be completed if the next level array cell is empty.

C. The Smallest Neighbor Cell Calculation

Searching for the smallest neighbor cell is done by priority, so if there are two or more neighboring cell that has the smallest value, then that cell is chosen based on priority.

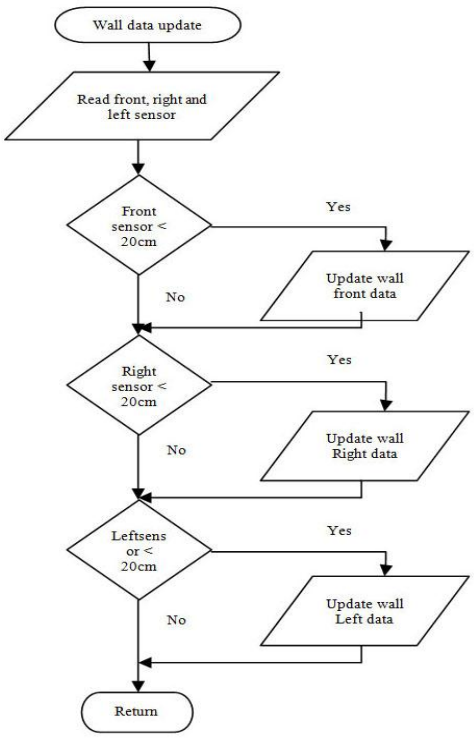

Figure 9. Wall location update flowchart.

Priority is set based on the movement of robots that move forward one cell has the highest priority, move one cell to the right is the second, while move one cell to the left is the third priority, and the fourth or final priority is to move one cell back. For example, if the robot faces the East, then the East cell has priority, the two South have priority cells, the cell has the third priority North, and the fourth priority is at the west, as in Fig. 10. When the robot faces the East, the East cell has priority, the South cell has priority second, North has the third priority cell, and the West cell has a fourth priority.

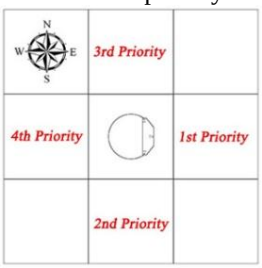

Figure 10. Priority of Neighbour cell

D. *Moving to the Smallest Neighbour Cell.*

Robot move to the smallest neighboring cells, and then the robot will move to the cell by observing orientation. When the smallest cell in the south cell and the robot is facing west, then it will move to the position of the cell, the robot must turn left, then move forward, as in Fig. 11.

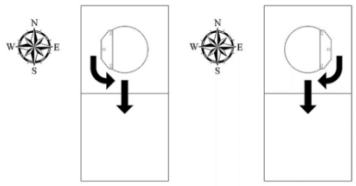

Figure 11. Moving to the smallest neighbor cell.

V. Results And Discussion

In this experiment, the Robot will learn to find the shortest path from the first cell (row 4, column 0) to the destination cell (row 2, column 2) and then return to the first cell. The robot's initial orientation faces North. The robot will learn to find the shortest path from the first cell at (row 4, column 0) to the destination cell at (row 2, column 2) and then return to the first cell [1].

The maze program aims to facilitate observations about how the Flood Filling algorithm is. Fig. 12 is a maze display simulator program. The maze blue wall is a wall whose position is known by robots, whereas the wall of the maze is colored in an orange wall where the robot is unknown.

A. *First Experiment*

In the first experiment, the Robot will look for the first cell line (4.0) to the destination cell (2, 2). The results of the Wall Follower algorithm and Pledge algorithm are shown in Tables Ii and III. The results of the combination method of Wall Follower - Flood Fill algorithm when cell line search (4, 0) to cell (2, 2) is shown in table 4, and the simulation results of the Pledge - Flood Fill algorithm is shown in Table V.

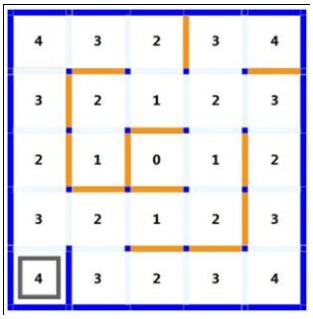

Figure 12. Simulation search route to cell (2,2), Turn = 0

TABLE II. First Experiment Result Using Wall Follower

	Routes	Steps
First run	(4,0) →(3,0) → (2,0) → (1,0) → (0,0) → (0,1) → (0,2) → (1,2) → (1,1) → (2,1) → (1,1) →(1,2) → (1,3) → (1,4) → (2,4) → (3,4) → (4,4) → (4,3) → (4,2) → (4,1) → (3,1) →(3,2) → (3,3) → (2,3) → (2,2)	24
Return home	(2,2) → (2,3) → (3,3) → (3,2) → (3,1) → (3,0) → (4,0)	6
Second run	(4,0) →(3,0) → (2,0) → (1,0) → (0,0) → (0,1) → (0,2) → (1,2) → (1,1) → (2,1) → (1,1) →(1,2) → (1,3) → (1,4) → (2,4) → (3,4) → (4,4) → (4,3) → (4,2) → (4,1) → (3,1) →(3,2) → (3,3) → (2,3) → (2,2)	24

TABLE III. FIRST EXPERIMENT RESULT USING PLEDGE

	Routes	Steps
First run	(4,0) →(3,0) → (2,0) → (1,0) → (0,0) → (0,1) → (0,2) → (1,2) → (1,3) → (0,3) → (0,4) →(0,3) → (1,3) → (2,3) → (2,2)	14
Return home	(2,2) → (2,3) → (3,3) → (3,2) → (3,1) → (3,0) → (4,0)	6
Second run	(4,0) →(3,0) → (2,0) → (1,0) → (0,0) → (0,1) → (0,2) → (1,2) → (1,3) → (0,3) → (0,4) →(0,3) → (1,3) → (2,3) → (2,2)	14

TABLE IV. FIRST EXPERIMENT RESULT USING WALL FOLLOWER – FLOOD FILL ALGORITHM

	Routes	Steps
First run	(4,0) →(3,0) → (2,0) → (1,0) → (2,0) → (3,0) → (3,1) → (3,2) → (3,3) → (2,3) → (2,2)	10
Return home	(2,2) → (2,3) → (3,3) → (3,2) → (3,1) → (3,0) → (4,0)	6
Second run	(4,0) →(3,0) → (3,1) → (3,2) → (3,3) → (2,3) → (2,2)	6

TABLE V. FIRST EXPERIMENT RESULT USING PLEDGE – FLOOD FILL ALGORITHM

	Routes	Steps
First run	(4,0) →(3,0) → (2,0) → (1,0) → (2,0) → (3,0) → (3,1) → (3,2) → (3,3) → (2,3) → (2,2)	10
Return home	(2,2) → (2,3) → (3,3) → (3,2) → (3,1) → (3,0) → (4,0)	6
Second run	(4,0) →(3,0) → (3,1) → (3,2) → (3,3) → (2,3) → (2,2)	6

The first run in the first experiment shows us that Pledge algorithm has better steps than the Wall Follower

algorithm to achieve the target point. But it also indicates that the synergistic Wall Follower – Flood Fill algorithm or Pledge – Flood Fill algorithm has better results than search applied only by using a Wall Follower algorithm or just a Pledge algorithm.

This experiment also shows that in the second run, the method that uses a combination of Wall Follower - Flood Fill Algorithm or a combination of Pledge - Flood Fill Algorithm has fewer steps than their first run. While the second run of the Wall Follower algorithm or second run of the Pledge algorithm still has the same steps as the first run because they do not record their experience in the first run.

In the first experiment, the robot updates the wall data while searching on the first run and go back home in the second run, the robot that using combination algorithm, has enough data to choose the fastest path to the destination in the cell (2,2). That's the reason why the trip back to the starting point and the second run has the same number of steps for both combination algorithm.

B. *Second Experiment*

The second experiment was carried out using a new maze, which can be seen in Figs. 13 and 14. The results of this second experiment can be seen in tables 6 to 9.

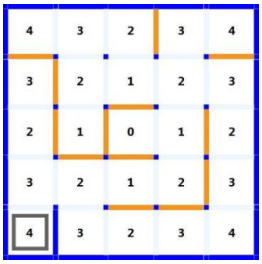

Figure 13. Simulation search path to cell (2,2) for the second experiment

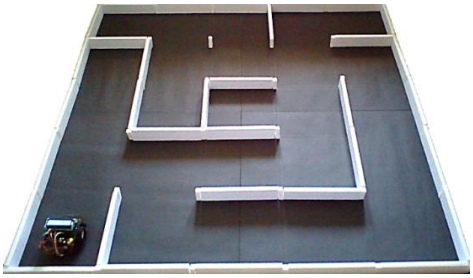

Figure 14. The maze for the second experiment.

TABLE VI. SECOND EXPERIMENT RESULT USING WALL FOLLOWER ALGORITHM

	Routes	Steps
First run	(4,0) →(3,0) → (2,0) → (1,0) → (2,0) → (3,0) → (3,1) → (3,2) → (3,3) → (2,3) → (2,2)	10
Return home	(2,2) → (2,3) → (3,3) → (3,2) → (3,1) → (3,0) → (4,0)	6
Second run	(4,0) →(3,0) → (2,0) → (1,0) → (2,0) → (3,0) → (3,1) → (3,2) → (3,3) → (2,3) → (2,2)	10

TABLE VII. SECOND EXPERIMENT RESULT USING PLEDGE ALGORITHM

	Routes	Steps
First run	(4,0) →(3,0) → (2,0) → (1,0) → (2,0) → (3,0) → (3,1) → (3,2) → (3,3) → (2,3) → (2,2)	10
Return home	(2,2) → (2,3) → (3,3) → (3,2) → (3,1) → (4,1) → (3,1) → (3,0) → (4,0)	8
Second run	(4,0) →(3,0) → (2,0) → (1,0) → (2,0) → (3,0) → (3,1) → (3,2) → (3,3) → (2,3) → (2,2)	10

TABLE VIII. SECOND EXPERIMENT RESULT USING WALL FOLLOWER – FLOOD FILL ALGORITHM

	Routes	Steps
First run	(4,0) →(3,0) → (2,0) → (1,0) → (2,0) → (3,0) → (3,1) → (3,2) → (3,3) → (2,3) → (2,2)	10
Return home	(2,2) → (2,3) → (3,3) → (3,2) → (3,1) → (3,0) → (4,0)	6
Second run	(4,0) →(3,0) → (3,1) → (3,2) → (3,3) → (2,3) → (2,2)	6

TABLE IX. SECOND EXPERIMENT RESULT USING PLEDGE – FLOOD FILL ALGORITHM

	Routes	Steps
First run	(4,0) →(3,0) → (2,0) → (1,0) → (2,0) → (3,0) → (3,1) → (3,2) → (3,3) → (2,3) → (2,2)	10
Return home	(2,2) → (2,3) → (3,3) → (3,2) → (3,1) → (4,1) → (3,1) → (3,0) → (4,0)	8
Second run	(4,0) →(3,0) → (3,1) → (3,2) → (3,3) → (2,3) → (2,2)	6

The results of the second experiment for all tests have the same results. But for the second run, all tests of the combination methods still have better results than the Wall Follower algorithm or the Pledge algorithm.

C. *Third Experiment*

The third experiment was carried out using a new maze, which can be seen in Figs. 15 and 16. The results of this second experiment can be seen in tables 10 to 13.

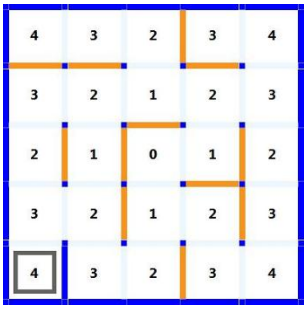

Figure 15. Simulation search path to cell (2,2) for the third experiment

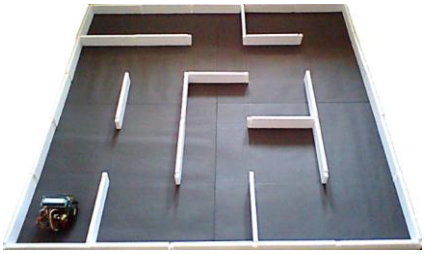

Figure 16. The maze for the third experiment.

TABLE X. THIRD EXPERIMENT RESULT USING WALL FOLLOWER ALGORITHM

	Routes	Steps
First run	(4,0) → (3,0) → (2,0) → (1,0) → (1,1) → (1,2) → (1,3) → (1,4) → (2,4) → (3,4) → (4,4) → (4,3) → (3,3) → (3,2) → (2,2)	14
Return home	(2,2) → (3,2) → (4,2) → (4,1) → (3,1) → (3,0) → (4,0)	6
Second run	(4,0) → (3,0) → (2,0) → (1,0) → (1,1) → (1,2) → (1,3) → (1,4) → (2,4) → (3,4) → (4,4) → (4,3) → (3,3) → (3,2) → (2,2)	14

TABLE XI. THIRD EXPERIMENT RESULT USING PLEDGE ALGORITHM

	Routes	Steps
First run	(4,0) → (3,0) → (2,0) → (1,0) → (1,1) → (1,2) → (0,2) → (0,1) → (0,0) → (0,1) → (0,2) → (1,2) → (1,3) → (2,3) → (2,2)	14
Return home	(2,2) → (3,2) → (4,2) → (4,1) → (3,1) → (3,0) → (4,0)	6
Second run	(4,0) → (3,0) → (2,0) → (1,0) → (1,1) → (1,2) → (0,2) → (0,1) → (0,0) → (0,1) → (0,2) → (1,2) → (1,3) → (2,3) → (2,2)	14

TABLE XII. THIRD EXPERIMENT RESULT USING WALL FOLLOWER – FLOOD FILL ALGORITHM

	Routes	Steps
First run	(4,0) → (3,0) → (2,0) → (1,0) → (1,1) → (1,2) → (1,3) → (2,3) → (2,2)	8
Return home	(2,2) → (3,2) → (4,2) → (4,1) → (3,1) → (3,0) → (4,0)	6
Second run	(4,0) →(3,0) → (3,1) → (4,1) → (4,2) → (3,2) → (2,2)	6

TABLE XIII. THIRD EXPERIMENT RESULT USING PLEDGE – FLOOD FILL ALGORITHM

	Routes	Steps
First run	(4,0) →(3,0) → (2,0) → (1,0) → (2,0) → (3,0) → (3,1) → (3,2) → (3,3) → (2,3) → (2,2)	10
Return home	(2,2) → (3,2) → (4,2) → (4,1) → (3,1) → (3,0) → (4,0)	6
Second run	(4,0) →(3,0) → (3,1) → (4,1) → (4,2) → (3,2) → (2,2)	6

In the first run of the third experiment, it was found that the Wall Follower - Flood Fill algorithm turned out to have better results than the Pledge - Flood Fill algorithm, with a difference of 2 steps faster while the return trip and second run have the same results.

The results of the Wall Follower combination method test - Flood Fill algorithm and Pledge - Flood Fill algorithm still have better results than the Wall Follower algorithm or the Pledge algorithm only.

In all experiments, wall map data will be updated when the robot enters a cell that has never been visited before. The Flood Fill algorithm will update cell values based on the position of the wall that the robot has mapped.

Robots always move to neighboring cells that have the smallest value. When there is more than one neighboring cell that has the smallest amount, then cell selection will be based on priority. Go forward has the priority, turn right has the second priority, turn left has the third priority, and move backward has the fourth priority.

This value is changed according to the position of the wall that has been mapped by the robot. The cell value represents the distance of the cell to the destination cell.

VI. Conclusion

The testing of mobile robots is done with the ability to learn how to navigate in unknown environments based on their own decisions. Algorithm Flood Fill is an effective algorithm as a combination of Wall Follower and Pledge algorithms for the completion of a medium-sized maze.

This mobile robot has managed to map the maze at first, return home, and run the second. In the second run, it reaches the target cell through the shortest route that was planned in the first run before and returns home.

Based on three experiments that have been conducted, it was found that the use of the Flood Fill algorithm can increase the effectiveness of the Wall Follower algorithm or the Pledge algorithm only. The results of the Wall Follower - Flood Fill combination algorithm and the Pledge - Flood Fill combination algorithm get almost the same results for these two algorithm combinations.

In order to develop a method of searching the maze that is more effective and faster, it is necessary to research various combinations of existing maze methods. Future works might include developing 3D maze research and also the robot's ability to compete in a bigger and more complex maze [19].

Conflict of Interest

The author declare no conflict of interest.

Author Contributions

This paper is a continuation of a series of studies on mazes that have been conducted previously. The first study using the Flood Fill algorithm and Wall Followers was a joint study between Semuil Tjiharjadi and Erwin Setiawan, where Semuil Tjiharjadi wrote the paper and led of the research, while Erwin Setiawan made the Robot and maze fields [11]. In the second study in 2017, the investigation continued with trying to optimize using A* and Flood Fill conducted by Semuil Tjiharjadi as a paper writer, research leader, including experiment and design; Marvin Chandra Wijaya compiled a research proposal and presentation; while the robot still uses a design made by Erwin Setiawan [15]. In the third study in 2019 [13], Semuil Tjiharjadi continued his research to implement the Flood Fill and Pledge methods in the robot maze, both of which have now been tested for their performance by combining the Flood Fill-Wall Follower algorithm with the Pledge-Wall Follower algorithm.

Acknowledgment

The author would like to thank the Computer Engineering Department of Maranatha Christian University for providing both financial assistance and the opportunity to research so that the series of research in the maze field can continue.

REFERENCES

[1] K. Collins and K. Borowski, "Experimental game interactions in a cave automatic virtual environment," in *2018 IEEE Games, Entertainment, Media Conference (GEM)*, 2018.

[2] G. Dudek, M. Jenkin, E. Milios, D. Wilkes, "Robotic exploration as graph construction," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 6, pp. 859-865, December 1991.

[3] E. &. C. K. Kivelevitch, "Multi-agent maze exploration," *Journal of Aerospace Computing, Information, and Communication*, vol. 7, no. 12, pp. 391-405, 2010.

[4] S. Vignesh and et al., "Cave exploration of mobile robots using soft computing algorithms," *International Journal of Computer Application*, vol. 71, no. 22, pp. 14-18, 2013.

[5] R. &. H. E. Zhou, "Breadth-first heuristic search," *Journal Artificial Intelligence*, vol. 170, no. 4-5, pp. 385-408, April 2006.

[6] S. &. M. S. Khan, "Depth first search in the semi-streaming model," *The Computing Research Repository (CoRR)*, January 2019.

[7] S. &. M. M. Forrest, "What makes a problem hard for a genetic algorithm? Some anomalous results and their explanation," *Machine Learning*, vol. 13, no. 2-3, pp. 285-319, November 1993.

[8] A. &. R. K. Kumaravel, "Algorithm for automaton specification for exploring dynamic mazes," *Indian Journal of Science and Technology*, vol. 6, no. 5, pp. 4554-4559, 2013.

[9] G. D. X. Liu, "A comparative study of a-star algorithms for search and rescue in perfect maze," in *International Conference on Electric Information and Control Engineering*, 2011.

[10] I. Elshamarka, A. B. S. Saman, "Design and Implementation of a robot for maze-solving using flood-fill algorithm," *International Journal of Computer Application*, vol. 56, no. 5, pp. 8-13, October 2012.

[11] S. Tjiharjadi, S. Setiawan, "Design and implementation of path finding robot using flood fill algorithm," *International Journal of Mechanical Engineering and Robotic Research*, vol. 5, no. 3, pp. 180-185, July 2016.

[12] J. R. B. D. Rosario and et al., "Modelling and characterization of a maze-solving mobile robot using wall follower algorithm," *Applied Mechanics and Materials*, vols. 446-447, pp. 1245-1249, 2014.

[13] S. Tjiharjadi, "Design and implementation of flood fill and pledge algorithm for maze robot," *International Journal of Mechanical Engineering and Robotics Research*, vol. 8, no. 4, pp. 632-638, July 2019.

[14] M. Babula, "Simulated maze solving algorithms through unknown mazes," in *XVIIIth Concurrency, Specification and Programming (CS&P) Workshop*, Krakow-Przegorzaly, 2009.

[15] S. Tjiharjadi, M. C. Wijaya, and E. Wijaya, "Optimization maze robot using A* and flood fill algorithm," *International Journal of Mechanical Engineering and Robotics Research*, vol. 6, no. 5, pp. 366-372, September 2017.

[16] N. K. S. S. W. I. S. Rao, "Robot navigation in unknown terrains: Introductory survey of non-heuristic algorithms," Oak Ridge National Laboratory, Oak Ridge, 1993.

[17] A. B. S. Saman and I. Abdramane, "Solving a reconfigurable maze using hybrid wall follower algorithm," *International Journal of Computer Application*, vol. 82, no. 3, pp. 22-26, 2013.

[18] Z. Cai, L. Ye, and A. Yang, "FloodFill maze solving with expected toll of penetrating unknown walls," in *2012 IEEE 14th International Conference on High Performance Computing and Communication*, 2012.

[19] H. K. Wazir and F. Annaz, "Using unity for 3D object orientation in a virtual environment," in *5th Brunei International Conference on Engineering and Technology*, 2014.

Semuil Tjiharjadi currently serves as vice-rector of capital human management, assets, and development. He is also Lectures in Computer Engineering Department. His primary research on Robotics, Computer automation, control, and security. He has written several books, To Be a Great Effective Leader (Jogjakarta, Indonesia: Andi Offset, 2012), Multimedia Programming by SMIL (Jogjakarta, Indonesia: Andi Offset, 2008), Computer Business Application (Bandung, Indonesia: Informatics, 2006) and so on. The various academic bodies on which he contributed as Head of Computer Engineering Department, Member: Senate of University, Member: APTIKOM, Member: MSDN Connection, Member: AAJI, Member: Fischertechnik Fan Club, Member: Asia Society of Researchers.