UKTI KORESPONDENSI

ARTIKEL JURNAL INTERNASIONAL BEREPUTASI

Judul artikel : Performance Comparison Robot Path Finding uses Flood Fill – Wall Follower Algorithm and Flood Fill – Pledge Algorithm

Jurnal : International Journal of Mechanical Engineering and Robotics Research, Vol. 9, No. 6, pp. 857-864.

Penulis : Semuil Tjiharjadi

No.	Perihal	Tanggal
1.	Bukti konfirmasi submit artikel dan artikel yang disubmit	22 Mei 2019
2.	Bukti konfirmasi lolos hasil preliminary review	29 Mei 2019
3.	Bukti konfirmasi review dan hasil review kedua	14 Juni 2019
4.	Bukti konfirmasi submit revisi kedua, respon kepada reviewer, dan artikel yang disubmit ulang	19 – 25 Juni 2019
5.	Bukti konfirmasi permintaan revisi tambahan	22 Mei2019
6.	Bukti konfirmasi submit revisi tambahan, dan artikel yang disubmit ulang	26 Juni 2019
7.	Bukti korespondensi artikel akan segera diterbitkan dan butuh konfirmasi terakhir, submit artikel terakhir dan perbaikannya.	7 Nopember 2019
8.	Bukti konfirmasi artikel published online	19 Mei 2020

1. Bukti Konfirmasi Submit Artikel dan Artikel yang Disubmit (22 Mei 2019)

ICAME2019-paper received-E009

4 messages

icameconf <icameconf@zhconf.ac.cn> Wed, May 22, 2019 at 3:12 PM
To: semuiltj <semuiltj@gmail.com>

Dear Semuil Tjiharjadi,

Greentings from ICAME 2019.

Thank you so much for your submission and support to ICAME 2019. Your paper "Performance Comparison Robot Path Finding uses Flood Fill - Wall Follower Algorithm and Flood Fill - Pledge Algorithm" has submitted successfully. Then we will send it for first review and infom you soon.

By the way, your paper is given the paper ID: E009. please keep in mind.

Please fill in the Authors' background and send the form back by May 24, 2019.

Position can be chosen from: **Prof. / Assoc. Prof. / Asst. Prof. / Lect. / Dr. / Ph. D Candidate / Postgraduate / others**					
Paper ID +organization	Full Name	Email address	Position	Research Interests	Personal website (if any)

Please feel free to contact me.
Have a wonderful day!

Thanks & Best Regards

Ms. Rachel Cao | Conference Secretary

E-mail: icameconf@zhconf.ac.cn | Web: http://www.icame.org/

Semuil Tjiharjadi <semuiltj@gmail.com> Thu, May 23, 2019 at 9:17 AM
To: icameconf <icameconf@zhconf.ac.cn>

Dear Ms. Rachel Cao,
Thank you for your confirmation email. I send back my information that you requested.
By the way, can I get an advance acceptance notification before June 15, 2019, because my institution will have a lot of loading work so it will take a long time to process my conference registration?
Thank you for your help and support.
Best Regards,

Semuil Tjiharjadi

Position can be chosen from: **Prof. / Assoc. Prof. / Asst. Prof. / Lect. / Dr. / Ph. D Candidate / Postgraduate / others**					
Paper ID +organization	Full Name	Email address	Position	Research Interests	Personal website (if any)
E009	Semuil Tjiharjadi	semuiltj@gmail.com	Asst. Prof.	Robot	

[Quoted text hidden]

---

icameconf@zhconf.ac.cn <icameconf@zhconf.ac.cn> Thu, May 23, 2019 at 9:28 AM
To: semuiltj@gmail.com

Dear Sir/Madam,

Greetings and thanks for your email.

The organizing committee will respond to your enquiry within two working days.

If you have any questions regarding the conference, please visit the website:

http://www.icame.com

Kind Regards

ICAME 2018

---

icameconf <icameconf@zhconf.ac.cn> Fri, May 24, 2019 at 10:06 AM
To: Semuil Tjiharjadi <semuiltj@gmail.com>

Dear Asst. Prof. Semuil Tjiharjadi,
Greetings.
We are well noted and due to keep the quality of all papers, we need enough time to review paper.
We will try to send you the notification before Jun 15, but we can not completely assured.
Please keep in touch.
If you have any question, please contact me.
Have a nice day.

Thanks & Best Regards

Ms. Rachel Cao | Conference Secretary

E-mail: icameconf@zhconf.ac.cn | Web: http://www.icame.org/

======= 2019-05-23 10:05:18 semuiltj@gmail.com在来信中写道： =======
---------- 源邮件 ----------
主题: Re: ICAME2019-paper received-E009
发件人: Semuil Tjiharjadi <semuiltj@gmail.com>
日期: Thu, May 23, 2019 10:17 am
收件人: ICAME <icameconf@zhconf.ac.cn>

[Quoted text hidden]

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 labyrinth, the robot must have ability to decide the direction taken at the intersection inside the labyrinth. Robot will map route and try to reach the destination in the fastest time and shortest distance. Robot will use two algorithms for path finding 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 destination 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 labyrinth. 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 — **path finding, 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 labyrinth robot. There are several algorithms that have been developed for this purpose and each algorithm has its own advantages and disadvantages [1].

As part of its autonomous ability, the Path Finding Robot uses structured algorithms to control the autonomous navigation it has [2]. In this study two combinations of algorithms were used to achieve the shortest and fastest target. The two combination algorithms are Flood fill algorithm - Wall follower algorithm as the first combination, while the second combination is 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 labyrinth. The main task is to find a path to complete the labyrinth in the shortest possible time and use the shortest way. The robot must start navigation from the corner of the labyrinth to the target as quickly as possible.

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 labyrinth 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 labyrinth 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 labyrinth. The search terms are only one cell that is opened to pass. The design of the labyrinth wall size and supporting platforms uses the IEEE standard.

II. LITERATURE REVIEW

2.1. 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 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 [3].

2.2. Depth First Search

The Depth First Search is an algorithm for searching based on tree data structures that uses 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 large computing power. A small increase in a path can result in a runtime increasing exponentially [4].

2.3. Heuristic Function

Heuristic Function plays vital role in optimization problem. It is a function that uses all mapping information to help the search process towards the right direction to achieve goals effectively [3].

2.4. 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 very large 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 of the entire population are improving [5].

2.5. A* algorithm

As one of the most popular methods for finding the shortest path in the labyrinth area, A* develops a combination heuristic approach. This approach is also used by the Best-First-Search (BFS) algorithm and the Dijkstra algorithm. Algorithm A* calculate the costs that associated with each used node. Such as the application of BFS, A* will follow its path with the lowest heuristic cost. Both them require large memory to store information, because all nodes that have been tested must be stored [6].

The A* algorithms can, during searching, judge the movement of 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 [7].

2.6. Flood Fill Algorithm

Flood fill algorithm that also known as 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 [8].

The Flood fill algorithm gives values to each node that represents the distance of the node from the center. It floods the labyrinth when it reaches a new cell or node. This algorithm requires continue update [9].

2.7. 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 fast algorithm and uses no extra memory. But this method will not necessarily find the shortest solution, and this algorithm has weakness when the labyrinth is not connected, it can back at the start point of the labyrinth [10].

2.8. Pledge Algorithm

The Pledge algorithm is designed to solve wall follower weakness. It can avoid obstacles and requires an arbitrarily chosen direction to go toward. At the beginning of algorithm, Pledge algorithm sets up direction and follows this direction. 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 original direction [10].

I. HARDWARE DESIGN

This research is tested using mobile robot. It have robot base construction by miniQ 2WD robot chassis, it is shown at Figure 1. It has a 122 mm diameter robot chassis, a couple wheels, a piece of ball caster and a couple Direct Current (DC) motors which have gear box and DC motor bracket.

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

Figure 1. 1*2WD miniQ robot chassis*.

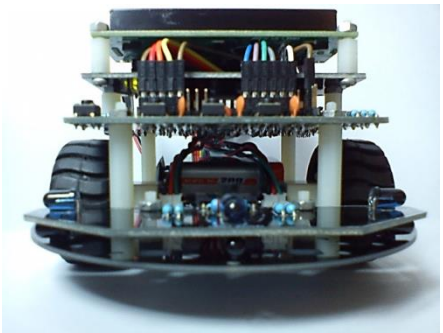

Figure 2. Mobile Robot from side view.

The robot has three infrared sensors to detect the front, right and left positions of the labyrinth 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 backwards. This labyrinth robot has an AT Mega 324 microcontroller 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 Figure 3.

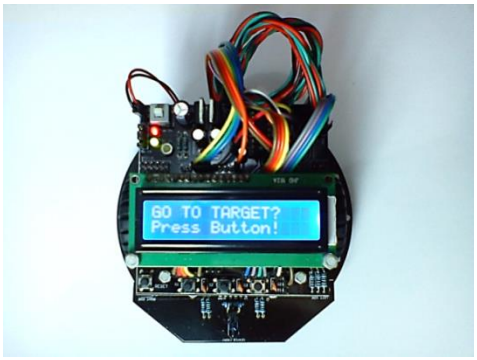

Figure 3. Mobile Robot from above view.

The block diagram of design of whole hardware system and the flowchart of main program can be seen at Figure 4 and Figure 5 [11].

The labyrinth designed for the robot to solve is of the size of 5×5 cells as shown in Figure 6. The actual labyrinth constructed, as shown in Figure 7, has a physical size of about 1.32 m^2. The labyrinth was designed so that it will have two paths for it to be solved. One of the paths is longer than the other. The robot (Figure 2) must decide which one of the paths is shorter and solve the labyrinth through that path [12].

The labyrinth designed to be solved by robots is 5 × 5 cells as shown in Figure 6. The actual labyrinth that was built, as shown in Figure 7, has a physical size of about 1.32 m2. The labyrinth 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 labyrinth through that path.

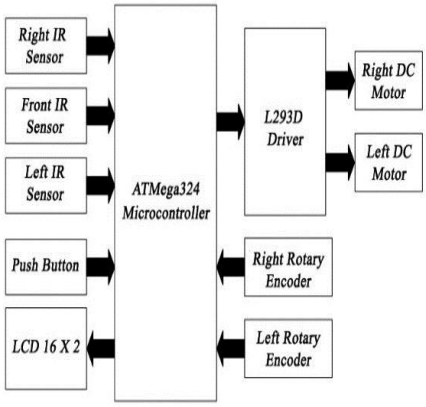

Figure 4. Maze Robot's Block Diagram.

II. ALGORITHM

In this study, three types of algorithms were used. 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.

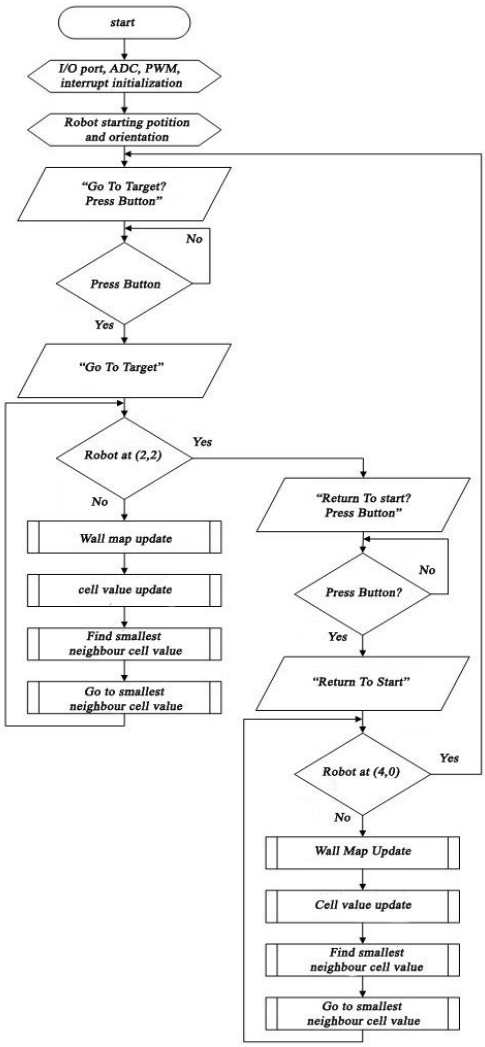

Figure 5. Flowchart of the main program.

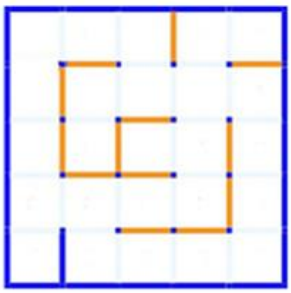

Figure 6. The layout of labyrinth.

Figure 7. The labyrinth 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 priority value based on the Flood Fill algorithm [9]. 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 goal by prioritizing the smallest possible 'Turn' variable value. Every time 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.

The Artificial Intelligence program has a two-dimensional array of memory to map the 5x5 labyrinth 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 movement of the robot in the array is done to position the robot as shown in Figure 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 decrease 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, labyrinth size, and location of the outer wall of the labyrinth.

Flood fill algorithm has four main steps: 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.

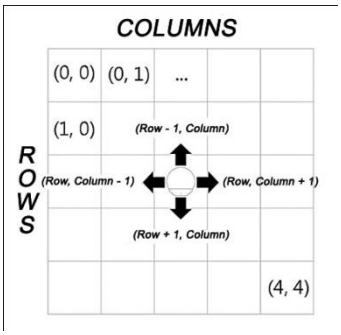

Figure 8. Robot's Array Movement

4.1 Wall data update

Robot will check its environment, any walls in its three directions: right, left and front directions. The robot will also detect the distance of any obstacle of its three directions. Anyone exceed 20 cm is updated as "wall" on its respective side. Flowchart in the Figure 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 Figure 9 explains the mechanism for updating wall data.

The maze robot also needs to know which direction it is facing so it knows where to go. Table 1 describes the relation of robot orientation and wall sensor detection. The robot has an initial orientation when it starts at the beginning 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 1.

Table 1. 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

4.2 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. The cells that will be updated are the current level array while the neighboring cells will be entered in the next level array. After the value filling process is complete, the cells in the next level array will be moved to the current level array to do the next value. The update process will be completed if the next level array cell is empty.

4.3 The smallest neigbour cell calculation

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

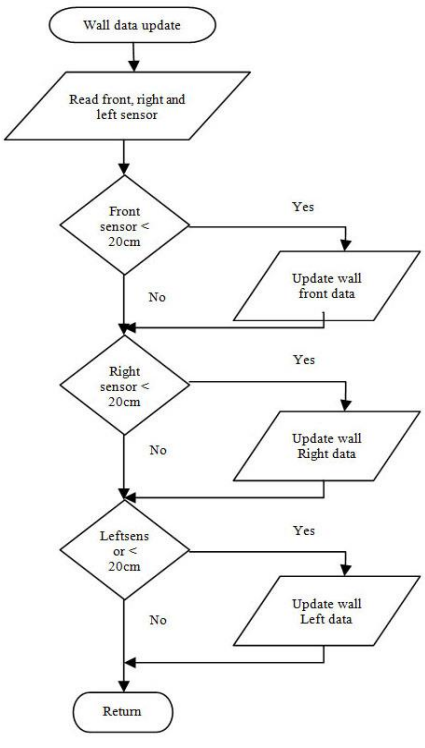

Figure 9. Flowchart for updating wall location at each cell

Priority is set based on the movement of robots that move forward one cell has priority, the second priority is to move one cell to the right, while the third priority is to move one cell to the left, 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 priority third North and the West cell has the fourth priority as in Figure 10. If 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.

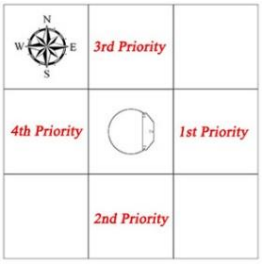

Figure 10. Priority of Neighbour cell

4.4. Moving to the smallest neighbour cell

Program subroutines move the robot to the smallest neighboring cells, then the robot will move to the cell by observing orientation. For example, if the South cell is the smallest cell and the orientation of the robot is facing west, then moves to the position of the cell, the robot must turn left, then move forward as in Figure 11.

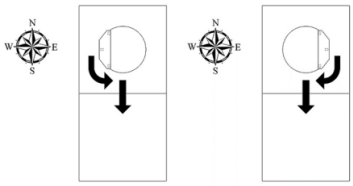

Figure 11. Moving to smallest neighbour cell.

III. RESULTS AND DISCUSSION

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

The maze program aims to facilitate observations about how the flood filling algorithm is. Figure 12 is a maze display simulator program. The labyrinth blue wall is a wall whose position is known by robots. Whereas the wall of the labyrinth is colored in an orange wall where the robot is unknown.

3.1 First Experiment

The first experiment, the Robot will look for the initial cell line (4.0) to the destination cell (2, 2). The results of the wall follower algorithm and pledge algorithm are shown in table 2 and 3. The results of 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 Pledge - Flood Fill algorithm is shown in table 5.

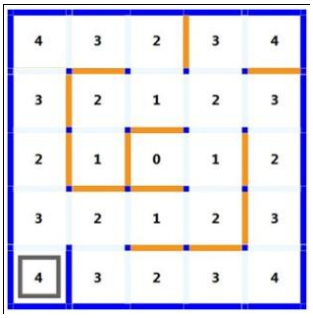

Figure 12. Simulation search path to cell (2,2), Turn = 0

Table 2. 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 3. 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 4. 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 5. 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 wall follower algorithm to achieve target point. But it also shows that synergistic Wall follower – Flood Fill algorithm or Pledge – Flood Fill algorithm have better results than search

applied only by using a wall follower algorithm or just a pledge algorithm.

This experiment also shows that in 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 wall follower algorithm or second run of pledge algorithm still have the same steps as first run, because they do not record their experience in first run.

After the robot updates the wall data while running a search on the first run in the first experiment and travels home in the second run, the robot that using combination algorithm, has enough data to find 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.

3.2 Second Experiment

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

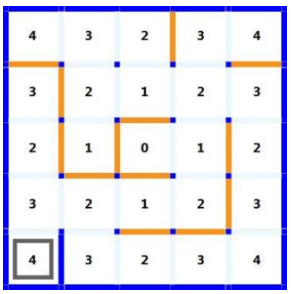

Figure 13. Simulation search path to cell (2,2) for second experiment

Figure 14. The maze for second experiment.

Table 6. 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 7. 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 8. 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 9. 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 test have same results. But for second run, all tests of the combination methods still have better results than the wall follower algorithm or the pledge algorithm.

3.3 Third Experiment

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

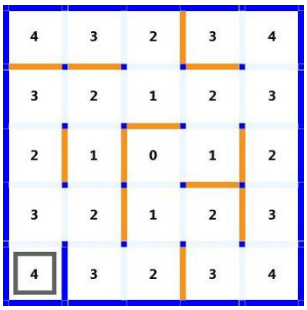

Figure 15. Simulation search path to cell (2,2) for third experiment

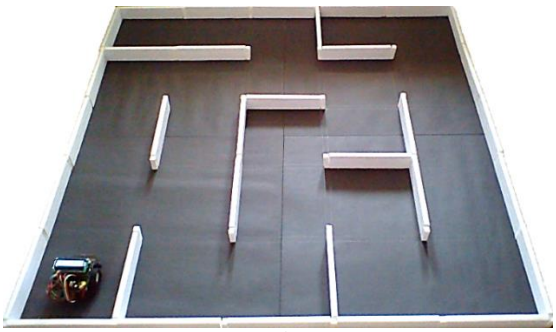

Figure 16. The maze for second experiment.

Table 10. 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 11. 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 12. 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 13. 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. If there are more than one neighboring cell that has the smallest value, then cell selection will be based on priority. Go foward has the first priority, turn right has the second priority, turn left has the third priority, and move backwards 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.

IV. 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 mapped 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 is able to 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.

REFERENCES

[1] 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.

[2] E. &. C. K. Kivelevitch, "Multi-Agent Maze Exploration," *Journal of Aerospace Computing, Information, and Communication*, vol. 7, no. 12, pp. 391-405, 2010.

[3] R. &. H. E. Zhou, "Breadth-First Heuristic Search," *Journal Artificial Intelligence*, vol. 170, no. 4-5, pp. 385-408, April 2006.

[4] S. &. M. S. Khan, "Depth First Search in the Semi-streaming Model," *The Computing Research Repository (CoRR)*, January 2019.

[5] 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.

[6] A. &. R. K. Kumaravel, "Algorithm for Automaton Specification for Exploring Dynamic Labyrinths," *Indian Journal of Science and Technology*, vol. 6, no. 5, pp. 4554-4559, 2013.

[7] &. 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.

[8] Elshamarka, I. & Saman, A.B.S, "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.

[9] Tjiharjadi, S. & Setiawan, S., "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.

[10] Babula, M., "Simulated Maze Solving Algorithms through Unknown Mazes," in *XVIIIth Concurrency, Specification and Programming (CS&P) Workshop*, Krakow-Przegorzaly, 2009.

[11] Tjiharjadi, S., Wijaya, M. C., and Wijaya, E., "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.

[12] 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.

Semuil Tjiharjadi is currently serves as vice rector of capital human management, assets and development. He is also Lectures in Computer Engineering Department. His major 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.

2. Bukti Konfirmasi lolos hasil preliminary review (29 Mei 2019)

ICAME 2019-Preliminary review result-E009

3 messages

icameconf <icameconf@zhconf.ac.cn> Wed, May 29, 2019 at 9:53 AM
To: semuiltj <semuiltj@gmail.com>

Dear Asst. Prof. Semuil Tjiharjadi,

Greentings from ICAME 2019.
We here glad to inform you that our paper "Performance Comparison Robot Path Finding uses Flood Fill - Wall Follower Algorithm and Flood Fill - Pledge Algorithm" has passed the first review. It follows the general rules of the paper format, and has passed the first primary review. It looks prepared sufficiently and nicely. But then, it will enter further peer review process which is performed by the reviewers who are experts and professors in related field, so the final review result will be sent by notification date June 15, 2019.
If you have any inquiries, please feel free to contact me anytime.

Thanks & Best Regards

Ms. Rachel Cao | Conference Secretary

E-mail: icameconf@zhconf.ac.cn | Web: http://www.icame.org/

Semuil Tjiharjadi <semuiltj@gmail.com> Wed, May 29, 2019 at 12:59 PM
To: icameconf <icameconf@zhconf.ac.cn>

Dear Ms Rachel.

Thank you for your confirmation email. I am looking forward my next progress acceptance.

Semuil Tjiharjadi
[Quoted text hidden]

---

icameconf@zhconf.ac.cn <icameconf@zhconf.ac.cn> Wed, May 29, 2019 at 1:01 PM
To: semuiltj@gmail.com

Dear Sir/Madam,

Greetings and thanks for your email.

The organizing committee will respond to your enquiry within two working days.

If you have any questions regarding the conference, please visit the website:

http://www.icame.com

Kind Regards

ICAME 2018

3. Bukti Konfirmasi Review Dan Hasil Review Kedua (14 Juni 2019)

ICAME2019-Notification-E009

21 messages

icameconf <icameconf@zhconf.ac.cn> Fri, Jun 14, 2019 at 2:05 PM
To: semuiltj <semuiltj@gmail.com>

Dear Asst. ProfSemuil Tjiharjadi,

Congratulations from ICAME conference office!

We are pleased to inform you that your paper (ID: E009), entitled *Performance Comparison Robot Path Finding uses Flood Fill - Wall Follower Algorithm and Flood Fill - Pledge Algorithm* has been accepted by ICAME 2019 for oral presentation and publication.

Accepted papers of ICAME 2019 will be published in International Journal of Mechanical Engineering and Robotics Research (IJMERR:ISSN: 2278-0149) , which will be indexed byIndex Corpernicus, ProQuest, UDL, Google Scholar, Open J-Gate, Scopus (since 2016) etc.

The attachment covers the notification and paper review form. Please complete the registration before **July 20, 2019.**

We are looking forward to meeting you in Phuket.

Thanks & Best Regards

Ms. Rachel Cao | Conference Secretary

E-mail: icameconf@zhconf.ac.cn | Web: http://www.icame.org/

2019 3rd International Conference on Automation and Mechatronics Engineering ***ICAME 2019***

Please consider the environment before printing this email.

2 attachments

ICAME2019-Notification-E009.pdf
335K

ICAME2019 Review Form-E009.pdf
92K

Notification of Acceptance of ICAME 2019

October 25-27, 2019 ‖ Phuket, Thailand

http://www.icame.org/

Dear Semuil Tjiharjadi,

Paper ID : E009

Paper Title : *Performance Comparison Robot Path Finding uses Flood Fill - Wall Follower Algorithm and Flood Fill - Pledge Algorithm*

Congratulations! The review processes for 2019 the 3rd International Conference on Automation and Mechatronics Engineering (ICAME 2019) has been completed. The conference committees received submissions from nearly 10 countries and regions. Based on the recommendations of the reviewers and the technical committees, we are pleased to inform you that your paper identified above has been accepted for publication and oral presentation. You are cordially invited to present your paper orally at ICAME 2019, during Oct. 25-27, 2019, in Phuket, Thailand.

All accepted papers after proper registration and presentation, will be included in International Journal of Mechanical Engineering and Robotics Research (ISSN: 2278-0149), which will be indexed by Index Corpernicus, ProQuest, UDL, Google Scholar, Open J-Gate, **Scopus** (since 2016) etc.

Please follow the five steps to finish registration.

i. Format your paper according to Review Form and Template carefully.

http://www.ijmerr.com/uploadfile/2015/0819/20150819064635301.doc

ii. Finish the Copyright Form.

http://www.ijmerr.com/uploadfile/2015/0326/20150326112301988.pdf

iii. Download and complete the Registration Form.

http://www.icame.org/author_reg.doc English version

iv. Finish the payment of Registration fee by Credit Card. (The detailed information can be found in the Registration form)

v. Send your final papers (both .doc and .pdf format), copyright form (.jpg or .pdf format) and filled registration form (.doc format) to us at **icameconf@zhconf.ac.cn** before **July 20, 2019**.

For the most updated information on the conference, please check the conference website at **http://www.icame.org/**. The Conference schedule will be available in late September, 2019. Please e-mail **icameconf@zhconf.ac.cn** for any queries concerning ICAME 2019.

Finally, we would like to further extend our congratulations to you and we are looking forward to meeting you in Phuket!

ICAME 2019 Organizing Committees

http://www.icame.org/

Phuket, Thailand

Review Form of ICAME 2019

http://www.icame.org/

Paper Title: Performance Comparison Robot Path Finding uses Flood Fill - Wall Follower Algorithm and Flood Fill - Pledge Algorithm

Evaluation(X where appropriate)

	Exceptional	Very Good	Good	Fair	Poor
Originality			X		
Innovation				X	
Technical Merit			X		
Applicability			X		
Presentation				X	
Relevance to the Conference			X		

Recommendation to Editors(X where appropriate)

	Strongly Accept	Accept	Marginally Accept	Reject	Strongly Reject
Recommendation		X			

Comments and Instructions

This paper 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 labyrinth. This paper was well organized. The abstract demonstrated the paper theme very well and the references are good to prove the strong background research for this paper. However, there are some suggestions if the author could consider revising this paper.

-Some references are aged. Please add more references in recent 5 years.
- It is recommended that the authors suggest possible directions for further studies beyond the work that has been completed in this manuscript in the conclusion section.

Basically this paper is good. Its topic also falls well into the category of the conference interests. So after some revision, it can be accepted for publication.

4. Bukti Konfirmasi Submit Revisi Kedua, Respon Kepada Reviewer Dan Artikel Yang DiSubmit Ulang (19-25 Juni 2019)

Semuil Tjiharjadi <semuiltj@gmail.com> Wed, Jun 19, 2019 at 10:41 AM
To: icameconf <icameconf@zhconf.ac.cn>

Dear Ms. Rachel Cao,

Thank you for your fast response to my request for the reviewer process. It is very nice to hear that my paper has been accepted. I am revising my paper now.
In fact, I have a long preparation for this paper since last year with the standard template of ICAME 2018 (that I have submitted a paper too), but I just realized that in ICAME 2019 the standard quantity of paper is not 7 pages anymore and become 5 pages.
The question is how much I should pay for my paper, should I reduce pages of my paper so I can fulfill the standard price? (By the way, I have ASR memberships)
Please help me with this problem so I can submit and pay my registration as soon as possible.
Thank you and best regards.

Semuil Tjiharjadi
[Quoted text hidden]

icameconf <icameconf@zhconf.ac.cn> Fri, Jun 21, 2019 at 11:03 AM
To: Semuil Tjiharjadi <semuiltj@gmail.com>

Dear Asst. ProfSemuil Tjiharjadi,
If your paper does not exceed 10 pages, you dont need pay the additional page fee.
Hope I can help you.
We are looking forward to your good news.

Thanks & Best Regards

Ms. Rachel Cao | Conference Secretary

E-mail: icameconf@zhconf.ac.cn | Web: http://www.icame.org/

2019 3rd International Conference on Automation and Mechatronics Engineering ***ICAME 2019***

Please consider the environment before printing this email.

======= 2019-06-19 11:06:38 semuiltj@gmail.com在来信中写道: =======
---------- 源邮件 ----------
主题: Re: ICAME2019-Notification-E009
发件人: Semuil Tjiharjadi <semuiltj@gmail.com>
日期: Wed, June 19, 2019 11:41 am
收件人: ICAME <icameconf@zhconf.ac.cn>

[Quoted text hidden]

Semuil Tjiharjadi <semuiltj@gmail.com> Tue, Jun 25, 2019 at 11:49 PM
To: icameconf <icameconf@zhconf.ac.cn>

Dear Ms. Rachel Cao,

I have updated my paper on the website and I don't know it is only for new paper submission.
I send my revision paper, copyright form and my online registration fee form payment.
Please send me the scanned receipt because my institution needs it.
Thank you and best regards.

Semuil Tjiharjadi
[Quoted text hidden]

4 attachments

copyright transfer agreement E009.pdf
288K

Payment ICAME 2019 E009.pdf
42K

Performance Comparison Path Finding Robot 2019 Rev.doc
3652K

Performance Comparison Path Finding Robot 2019 Rev.pdf
603K

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 labyrinth, the robot must have ability to decide the direction taken at the intersection inside the labyrinth. Robot will map route and try to reach the destination in the fastest time and shortest distance. Robot will use two algorithms for path finding 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 destination 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 labyrinth. 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 — **path finding, 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 labyrinth robot [1]. There are several algorithms that have been developed for this purpose and each algorithm has its own 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 Flood fill algorithm - Wall follower algorithm as the first combination, while the second combination is 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 labyrinth. The main task is to find a path to complete the labyrinth in the shortest possible time and use the shortest way. The robot must start navigation from the corner of the labyrinth 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 labyrinth 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 labyrinth 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 labyrinth. The search terms are only one cell that is opened to pass. The design of the labyrinth wall size and supporting platforms uses the IEEE standard.

II. LITERATURE REVIEW

2.1. 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 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].

2.2. Depth First Search

The Depth First Search is an algorithm for searching based on tree data structures that uses 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 large computing power. A small increase in a path can result in a runtime increasing exponentially [6].

2.3. Heuristic Function

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

2.4. 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 very large 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 of the entire population are improving [7].

2.5. A* algorithm

As one of the most popular methods for finding the shortest path in the labyrinth area, A* develops a combination heuristic approach. This approach is also used by the Best-First-Search (BFS) algorithm and the Dijkstra algorithm. Algorithm A* calculate the costs that associated with each used node. Such as the application of BFS, A* will follow its path with the lowest heuristic cost. Both 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 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].

2.6. Flood Fill Algorithm

Flood fill algorithm that also known as 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 gives values to each node that represents the distance of the node from the center. It floods the labyrinth when it reaches a new cell or node. This algorithm requires continue update [11].

2.7. 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 fast algorithm and uses no extra memory. But this method will not necessarily find the shortest solution, and this algorithm has weakness when the labyrinth is not connected, it can back at the start point of the labyrinth [12].

2.8. Pledge Algorithm

The Pledge algorithm is designed to solve wall follower weakness. It can avoid obstacles and requires an arbitrarily chosen direction to go toward. At the beginning of algorithm, Pledge algorithm sets up direction and follows this direction. 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 original direction [13].

I. HARDWARE DESIGN

This research is tested using mobile robot. It has robot base construction by miniQ 2WD robot chassis, it is shown at Figure 1. It has a 122 mm diameter robot chassis, a couple wheels, a piece of ball caster and a couple Direct Current (DC) motors which have gear box and DC motor bracket.

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

Figure 1. 1*2WD miniQ robot chassis*.

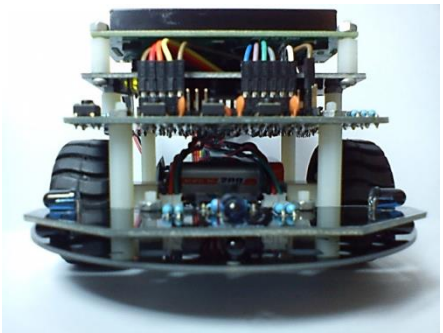

Figure 2. Mobile Robot from side view.

The robot has three infrared sensors to detect the front, right and left positions of the labyrinth 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 backwards. This labyrinth robot has an AT Mega 324 microcontroller 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 Figure 3.

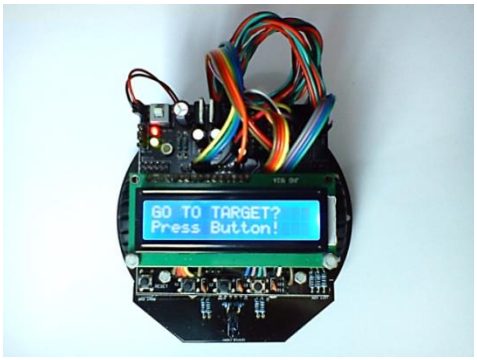

Figure 3. Mobile Robot from above view.

The block diagram of design of whole hardware system and the flowchart of main program can be seen at Figure 4 and Figure 5 [14].

The labyrinth designed to be solved by robots is 5 × 5 cells as shown in Figure 6. The actual labyrinth that was built, as shown in Figure 7, has a physical size of about 1.32 m2. The labyrinth 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 labyrinth through that path [15].

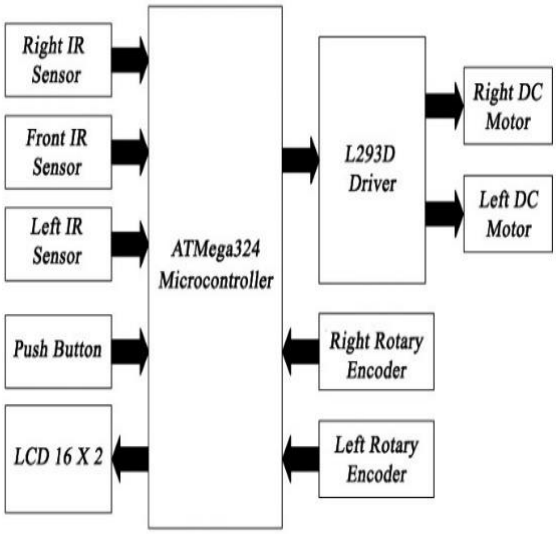

Figure 4. Maze Robot's Block Diagram.

II. ALGORITHM

In this study, three types of algorithms were used. 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.

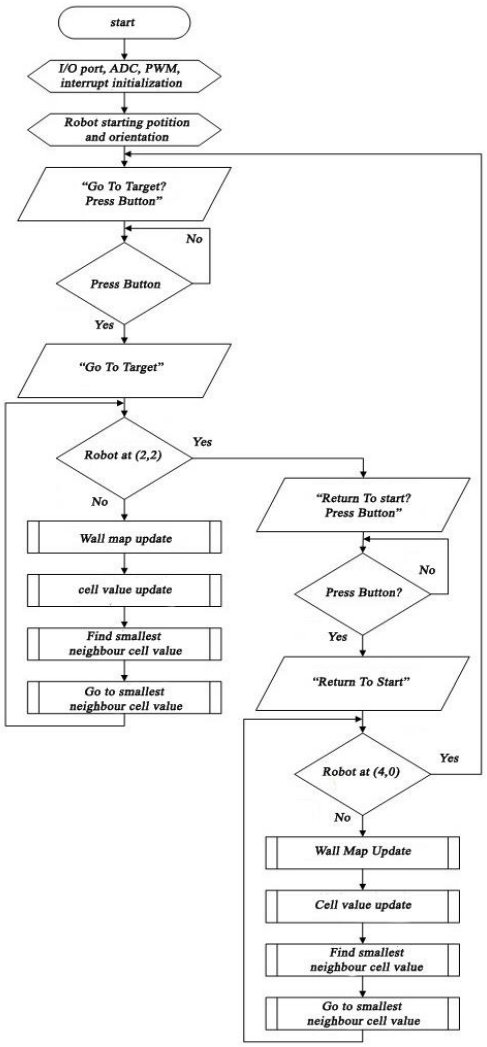

Figure 5. Flowchart of the main program.

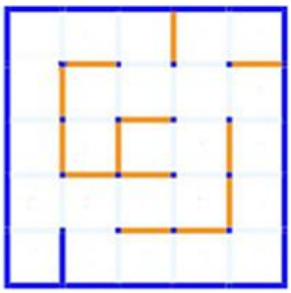

Figure 6. The layout of labyrinth.

Figure 7. The labyrinth 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 priority value 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 goal by prioritizing the smallest possible 'Turn' variable value. Every time 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.

The Artificial Intelligence program has a two-dimensional array of memory to map the 5x5 labyrinth 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 movement of the robot in the array is done to position the robot as shown in Figure 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 decrease 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, labyrinth size, and location of the outer wall of the labyrinth.

Flood fill algorithm has four main steps: 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.

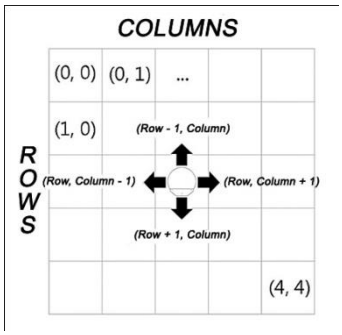

Figure 8. Robot's Array Movement

4.1 Wall data update

Robot will check its environment, any walls in its three directions: right, left and front directions. The robot will also detect the distance of any obstacle of its three directions. Anyone exceed 20 cm is updated as "wall" on its respective side. Flowchart in the Figure 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 Figure 9 explains the mechanism for updating wall data.

The maze robot also needs to know which direction it is facing so it knows where to go. Table 1 describes the relation of robot orientation and wall sensor detection. The robot has an initial orientation when it starts at the beginning 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 1.

Table 1. 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

4.2 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. The cells that will be updated are the current level array while the neighboring cells will be entered in the next level array. After the value filling process is complete, the cells in the next level array will be moved to the current level array to do the next value. The update process will be completed if the next level array cell is empty.

4.3 The smallest neigbour cell calculation

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

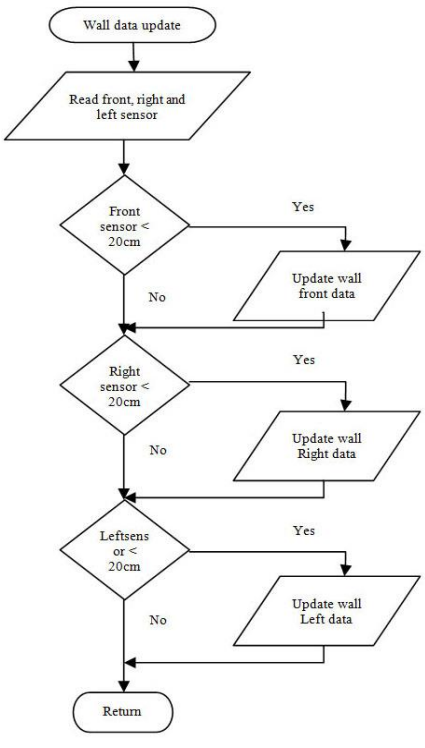

Figure 9. Flowchart for updating wall location at each cell

Priority is set based on the movement of robots that move forward one cell has priority, the second priority is to move one cell to the right, while the third priority is to move one cell to the left, 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 priority third North and the West cell has the fourth priority as in Figure 10. If 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.

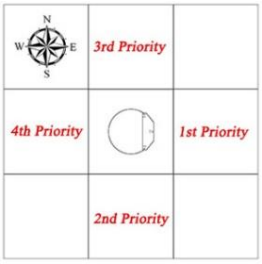

Figure 10. Priority of Neighbour cell

4.4. Moving to the smallest neighbour cell

Program subroutines move the robot to the smallest neighboring cells, then the robot will move to the cell by observing orientation. For example, if the South cell is the smallest cell and the orientation of the robot is facing west, then moves to the position of the cell, the robot must turn left, then move forward as in Figure 11.

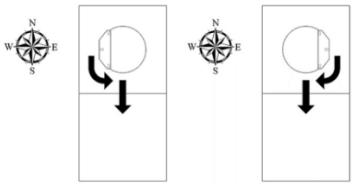

Figure 11. Moving to smallest neighbour cell.

III. RESULTS AND DISCUSSION

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

The maze program aims to facilitate observations about how the flood filling algorithm is. Figure 12 is a maze display simulator program. The labyrinth blue wall is a wall whose position is known by robots. Whereas the wall of the labyrinth is colored in an orange wall where the robot is unknown.

3.1 First Experiment

The first experiment, the Robot will look for the initial cell line (4.0) to the destination cell (2, 2). The results of the wall follower algorithm and pledge algorithm are shown in table 2 and 3. The results of 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 Pledge - Flood Fill algorithm is shown in table 5.

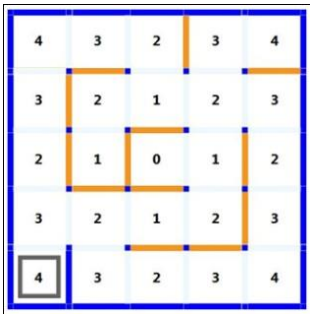

Figure 12. Simulation search path to cell (2,2), Turn = 0

Table 2. 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 3. 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 4. 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 5. 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 wall follower algorithm to achieve target point. But it also shows that synergistic Wall follower – Flood Fill algorithm or Pledge – Flood Fill algorithm have better results than search

applied only by using a wall follower algorithm or just a pledge algorithm.

This experiment also shows that in 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 wall follower algorithm or second run of pledge algorithm still have the same steps as first run, because they do not record their experience in first run.

After the robot updates the wall data while running a search on the first run in the first experiment and travels home in the second run, the robot that using combination algorithm, has enough data to find 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.

3.2 Second Experiment

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

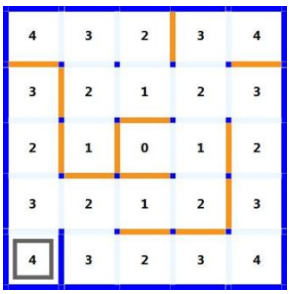

Figure 13. Simulation search path to cell (2,2) for second experiment

Figure 14. The maze for second experiment.

Table 6. 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 7. 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 8. 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 9. 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 test have same results. But for second run, all tests of the combination methods still have better results than the wall follower algorithm or the pledge algorithm.

3.3 Third Experiment

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

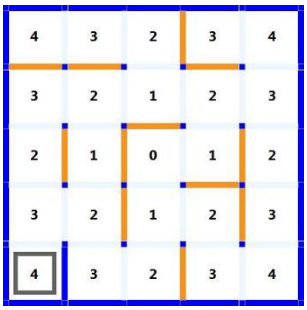

Figure 15. Simulation search path to cell (2,2) for third experiment

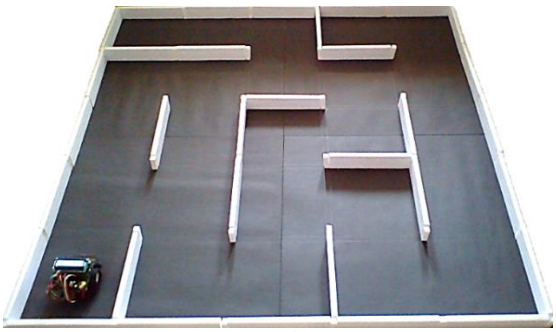

Figure 16. The maze for third experiment.

Table 10. 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 11. 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 12. 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 13. 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. If there are more than one neighboring cell that has the smallest value, then cell selection will be based on priority. Go foward has the first priority, turn right has the second priority, turn left has the third priority, and move backwards 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.

IV. 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 mapped 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 is able to 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 complete in a bigger and more complex maze.

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 Labyrinths," *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] Elshamarka, I. & Saman, A.B.S, "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] Tjiharjadi, S. & Setiawan, S., "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] Babula, M., "Simulated Maze Solving Algorithms through Unknown Mazes," in *XVIIIth Concurrency, Specification and Programming (CS&P) Workshop*, Krakow-Przegorzaly, 2009.

[14] Tjiharjadi, S., Wijaya, M. C., and Wijaya, E., "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.

[15] 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.

[16] 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.

[17] 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.

[18] L. L. Kai and F. Annaz, "Implementation of the Tremaux Maze Solving Algorithm to an Omnidirectional Mobile Robot," in *13th International Conference on Electronics, Information, and Communication*, Kinabalu, 2014.

[19] N. Z. Yew, K. M. Tiong and S. T. Yong, "Recursive Path-finding in a Dynamic Maze with Modified Tremaux's Algorithm," *International Journal of Mathematical and Computational Sciences*, vol. 5, no. 12, pp. 2102-2104, 2011.

[20] 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.

[21] K. L. Lim and F. Annaz, "Implementation of the Tremaux Maze Solving Algorithm to an Omnidirectional Mobile Robot," in *13th International Conference on Electronics, Information and Communication*, Kinabalu, 2014.

Semuil Tjiharjadi is currently serves as vice rector of capital human management, assets and development. He is also Lectures in Computer Engineering Department. His major 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.

5. Bukti Konfirmasi Permintaan Revisi Tambahan (22 Mei 2019)

icameconf <icameconf@zhconf.ac.cn>
to me

Jun 26, 2019, 1:53 PM

Dear Asst. Prof. Semuil Tjiharjadi,
Thanks for the registration.
Please revise your paper as below:
-Reference [16-21] are not mentioned in the text clearly.
Thanks and we are looking forward to your reply.

Thanks & Best Regards

Ms. Rachel Cao | Conference Secretary

E-mail: icameconf@zhconf.ac.cn | Web: http://www.icame.org/

2019 3rd International Conference on Automation and Mechatronics Engineering ***ICAME 2019***

6. Bukti Konfirmasi Submit Tambahan Dan Artikel Yang Disubmit Ulang (26 Juni 2019)

Semuil Tjiharjadi <semuiltj@gmail.com>
to icameconf

Jun 26, 2019, 4:09 PM

Dear Ms. Rachel Cao,

Thank you for your email. I missed removing some references.
I send my paper again. By the way, I really need **the receipt,** it will determine whether my institution will allow my participation in the next ASR conferences.
Is there any problem to send me the receipt after I made the payment?
Best regards

Semuil Tjiharjadi

•••

2 Attachments • Scanned by Gmail

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 labyrinth, the robot must have ability to decide the direction taken at the intersection inside the labyrinth. Robot will map route and try to reach the destination in the fastest time and shortest distance. Robot will use two algorithms for path finding 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 destination 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 labyrinth. 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 — **path finding, 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 labyrinth robot [1]. There are several algorithms that have been developed for this purpose and each algorithm has its own 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 Flood fill algorithm - Wall follower algorithm as the first combination, while the second combination is 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 labyrinth. The main task is to find a path to complete the labyrinth in the shortest possible time and use the shortest way. The robot must start navigation from the corner of the labyrinth 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 labyrinth 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 labyrinth 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 labyrinth. The search terms are only one cell that is opened to pass. The design of the labyrinth wall size and supporting platforms uses the IEEE standard.

II. LITERATURE REVIEW

2.1. 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 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].

2.2. Depth First Search

The Depth First Search is an algorithm for searching based on tree data structures that uses 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 large computing power. A small increase in a path can result in a runtime increasing exponentially [6].

2.3. Heuristic Function

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

2.4. 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 very large 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 of the entire population are improving [7].

2.5. A* algorithm

As one of the most popular methods for finding the shortest path in the labyrinth area, A* develops a combination heuristic approach. This approach is also used by the Best-First-Search (BFS) algorithm and the Dijkstra algorithm. Algorithm A* calculate the costs that associated with each used node. Such as the application of BFS, A* will follow its path with the lowest heuristic cost. Both 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 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].

2.6. Flood Fill Algorithm

Flood fill algorithm that also known as 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 gives values to each node that represents the distance of the node from the center. It floods the labyrinth when it reaches a new cell or node. This algorithm requires continue update [11].

2.7. 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 fast algorithm and uses no extra memory. But this method will not necessarily find the shortest solution, and this algorithm has weakness when the labyrinth is not connected, it can back at the start point of the labyrinth [12].

2.8. Pledge Algorithm

The Pledge algorithm is designed to solve wall follower weakness. It can avoid obstacles and requires an arbitrarily chosen direction to go toward. At the beginning of algorithm, Pledge algorithm sets up direction and follows this direction. 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 original direction [13].

I. HARDWARE DESIGN

This research is tested using mobile robot. It has robot base construction by miniQ 2WD robot chassis, it is shown at Figure 1. It has a 122 mm diameter robot chassis, a couple wheels, a piece of ball caster and a couple Direct Current (DC) motors which have gear box and DC motor bracket.

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

Figure 1. 1*2WD miniQ robot chassis*.

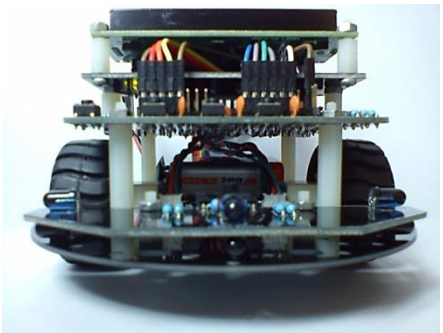

Figure 2. Mobile Robot from side view.

The robot has three infrared sensors to detect the front, right and left positions of the labyrinth 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 backwards. This labyrinth robot has an AT Mega 324 microcontroller 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 Figure 3.

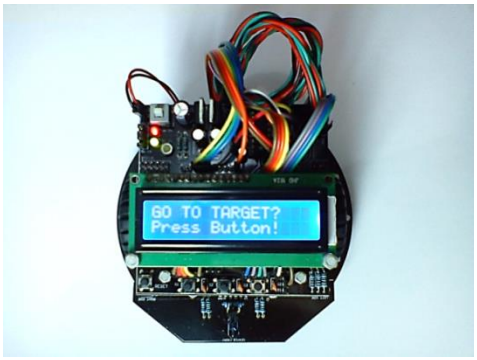

Figure 3. Mobile Robot from above view.

The block diagram of design of whole hardware system and the flowchart of main program can be seen at Figure 4 and Figure 5 [14].

The labyrinth designed to be solved by robots is 5 × 5 cells as shown in Figure 6. The actual labyrinth that was built, as shown in Figure 7, has a physical size of about 1.32 m2. The labyrinth 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 labyrinth through that path [15].

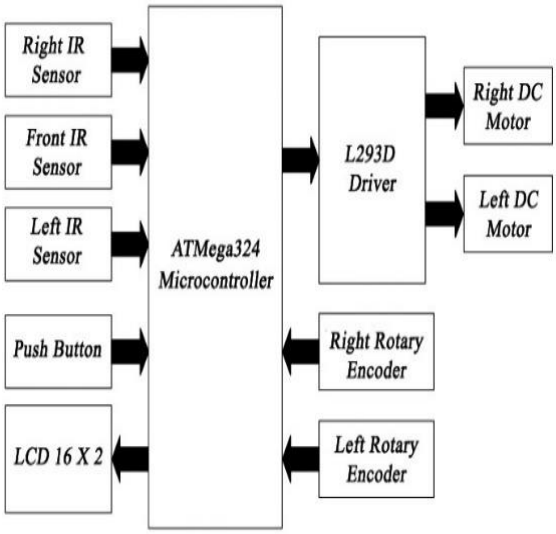

Figure 4. Maze Robot's Block Diagram.

II. ALGORITHM

In this study, three types of algorithms were used. 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.

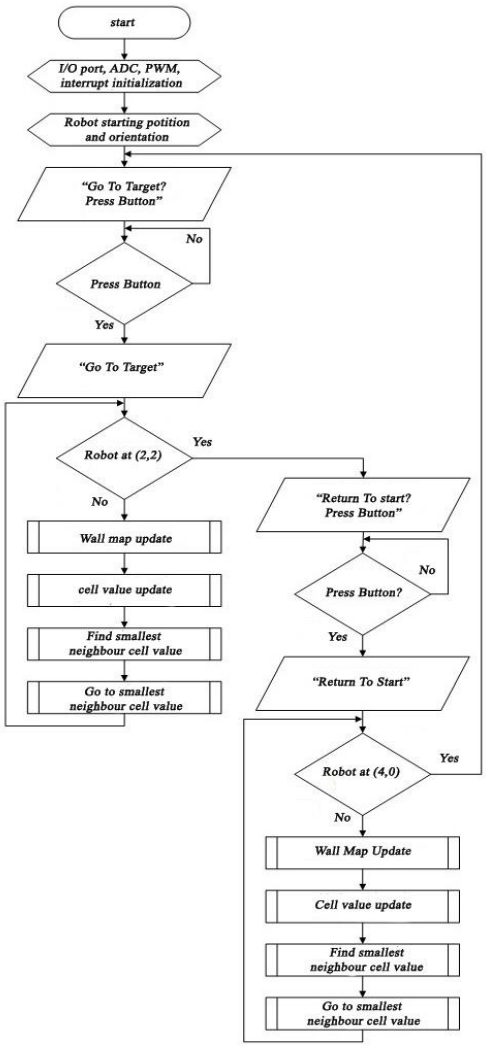

Figure 5. Flowchart of the main program.

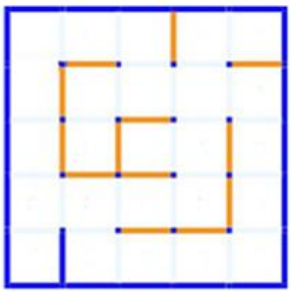

Figure 6. The layout of labyrinth.

Figure 7. The labyrinth 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 priority value 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 goal by prioritizing the smallest possible 'Turn' variable value. Every time 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 [16].

The Artificial Intelligence program has a two-dimensional array of memory to map the 5x5 labyrinth 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 movement of the robot in the array is done to position the robot as shown in Figure 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 decrease 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, labyrinth size, and location of the outer wall of the labyrinth [17].

Flood fill algorithm has four main steps: 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.

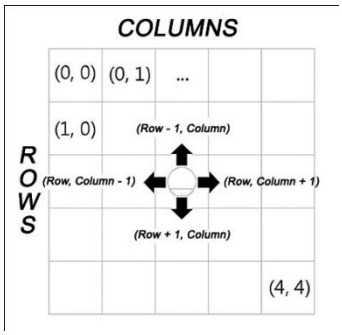

Figure 8. Robot's Array Movement

4.1 Wall data update

Robot will check its environment, any walls in its three directions: right, left and front directions. The robot will also detect the distance of any obstacle of its three directions. Anyone exceed 20 cm is updated as "wall" on its respective side. Flowchart in the Figure 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 Figure 9 explains the mechanism for updating wall data.

The maze robot also needs to know which direction it is facing so it knows where to go. Table 1 describes the relation of robot orientation and wall sensor detection. The robot has an initial orientation when it starts at the beginning 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 1.

Table 1. 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

4.2 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. The cells that will be updated are the current level array while the neighboring cells will be entered in the next level array. After the value filling process is complete, the cells in the next level array will be moved to the current level array to do the next value. The update process will be completed if the next level array cell is empty.

4.3 The smallest neigbour cell calculation

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

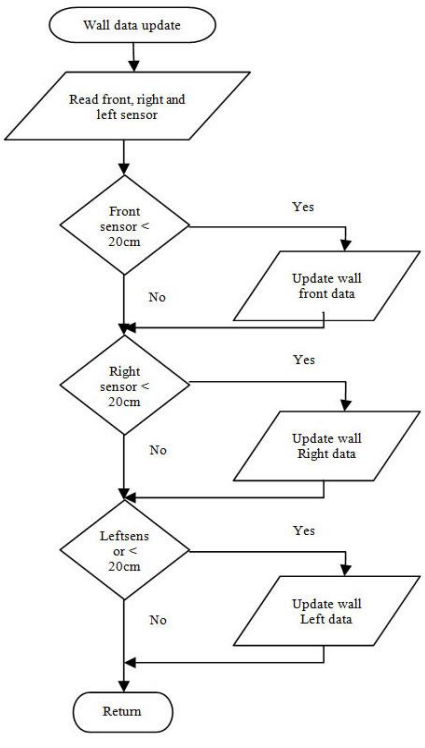

Figure 9. Flowchart for updating wall location at each cell

Priority is set based on the movement of robots that move forward one cell has priority, the second priority is to move one cell to the right, while the third priority is to move one cell to the left, 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 priority third North and the West cell has the fourth priority as in Figure 10. If 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.

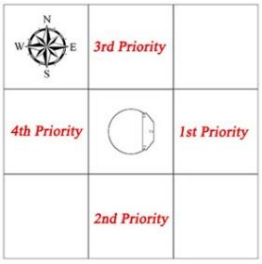

Figure 10. Priority of Neighbour cell

4.4. Moving to the smallest neighbour cell

Program subroutines move the robot to the smallest neighboring cells, then the robot will move to the cell by observing orientation. For example, if the South cell is the smallest cell and the orientation of the robot is facing west, then moves to the position of the cell, the robot must turn left, then move forward as in Figure 11.

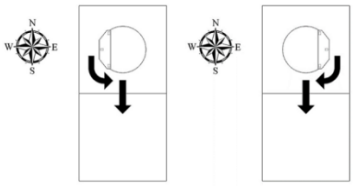

Figure 11. Moving to smallest neighbour cell.

III. RESULTS AND DISCUSSION

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

The maze program aims to facilitate observations about how the flood filling algorithm is. Figure 12 is a maze display simulator program. The labyrinth blue wall is a wall whose position is known by robots. Whereas the wall of the labyrinth is colored in an orange wall where the robot is unknown.

3.1 First Experiment

The first experiment, the Robot will look for the initial cell line (4.0) to the destination cell (2, 2). The results of the wall follower algorithm and pledge algorithm are shown in table 2 and 3. The results of 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 Pledge - Flood Fill algorithm is shown in table 5.

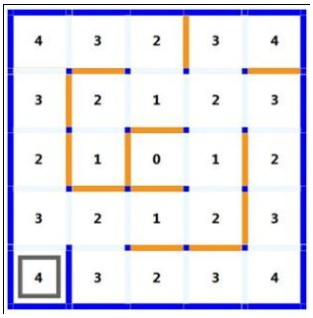

Figure 12. Simulation search path to cell (2,2), Turn = 0

Table 2. 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 3. 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 4. 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 5. 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 wall follower algorithm to achieve target point. But it also shows that synergistic Wall follower – Flood Fill algorithm or Pledge – Flood Fill algorithm have better results than search

applied only by using a wall follower algorithm or just a pledge algorithm.

This experiment also shows that in 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 wall follower algorithm or second run of pledge algorithm still have the same steps as first run, because they do not record their experience in first run.

After the robot updates the wall data while running a search on the first run in the first experiment and travels home in the second run, the robot that using combination algorithm, has enough data to find 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.

3.2 Second Experiment

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

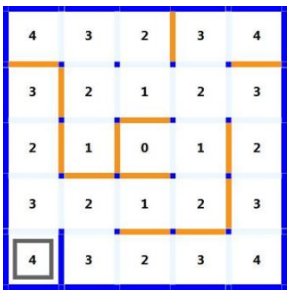

Figure 13. Simulation search path to cell (2,2) for second experiment

Figure 14. The maze for second experiment.

Table 6. 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 7. 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 8. 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 9. 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 test have same results. But for second run, all tests of the combination methods still have better results than the wall follower algorithm or the pledge algorithm.

3.3 Third Experiment

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

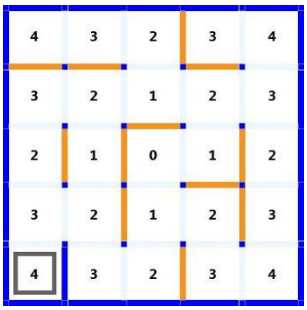

Figure 15. Simulation search path to cell (2,2) for third experiment

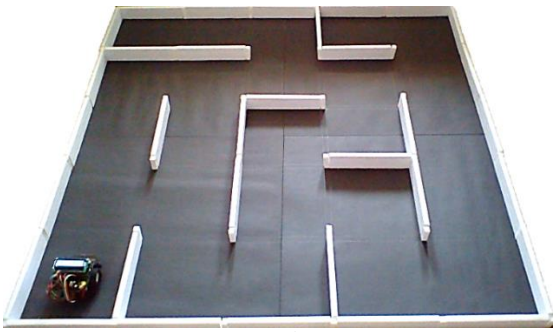

Figure 16. The maze for third experiment.

Table 10. 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 11. 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 12. 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 13. 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. If there are more than one neighboring cell that has the smallest value, then cell selection will be based on priority. Go foward has the first priority, turn right has the second priority, turn left has the third priority, and move backwards 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.

IV. 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 mapped 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 is able to 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 complete in a bigger and more complex maze [18].

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 Labyrinths," *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] Elshamarka, I. & Saman, A.B.S, "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] Tjiharjadi, S. & Setiawan, S., "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] Babula, M., "Simulated Maze Solving Algorithms through Unknown Mazes," in *XVIIIth Concurrency, Specification and Programming (CS&P) Workshop*, Krakow-Przegorzaly, 2009.

[14] Tjiharjadi, S., Wijaya, M. C., and Wijaya, E., "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.

[15] 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.

[16] 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.

[17] 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.

[18] 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 is currently serves as vice rector of capital human management, assets and development. He is also Lectures in Computer Engineering Department. His major 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.

7. Bukti Korespondensi Artikel Akan Segera Diterbitkan Dan Butuh Konfirmasi Terakhir, Submit Artikel Terakhir Dan Perbaikannya (7 Nopember 2019)

Gmail Semuil Tjiharjadi <semuiltj@gmail.com>

ICAME 2019-final paper confirmation before publication-IJMERR-E009

icameconf <icameconf@zhconf.ac.cn> Thu, Nov 7, 2019 at 1:27 PM
To: semuiltj <semuiltj@gmail.com>

Dear Semuil Tjiharjadi,

Greetings from Rachel and ICAME 2019.

We have revised your paper （E009） format with the IJMERR template as the press demand. Please confirm whether the content is wrong or not and whether the information and order of authors are wrong or not. If yes, please revise your paper based on the attached version I sent to you and then send it back to us before **Nov. 15, 2019**. （both doc and pdf version needed）

Please also add the highnighted part as below mentioned.

Please open it with "office" only.

Thanks for your cooperation.

Looking forward to your reply.

Thanks & Best Regards

Ms. Rachel Cao | Conference Secretary

E-mail: icameconf@zhconf.ac.cn | Web: http://www.icame.org/

3 attachments

E009-IJMERR.doc
3657K

E009-IJMERR.pdf
589K

IJMERR.doc
104K

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 June 14, 2019; accepted June 28, 2019.
corresponding author: Semuil Tjiharjadi, semuiltj@gmail.com

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 Figure 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.

Figure 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.

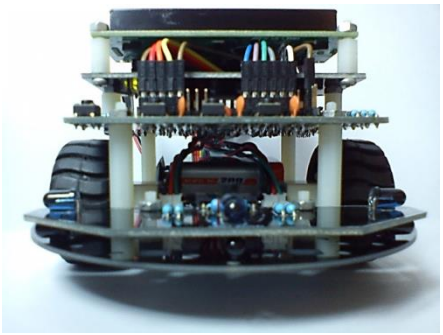

Figure 2. Mobile Robot from side view.

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 Figure 3.

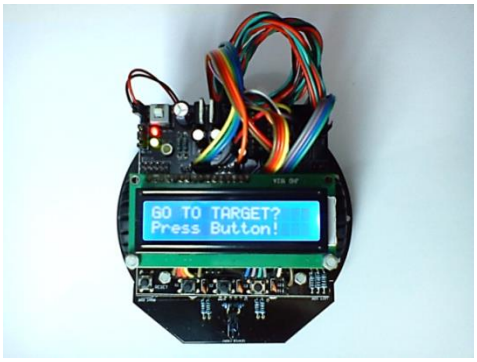

Figure 3. Display of Mobile Robot from the above.

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

The maze designed to be solved by robots is 5 × 5 cells, as shown in Figure 6. The actual maze that was built, as shown in Figure 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].

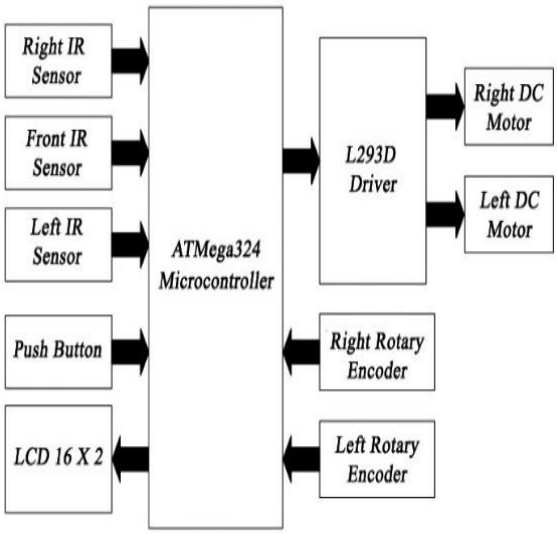

Figure 4. Maze Robot's Block Diagram.

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.

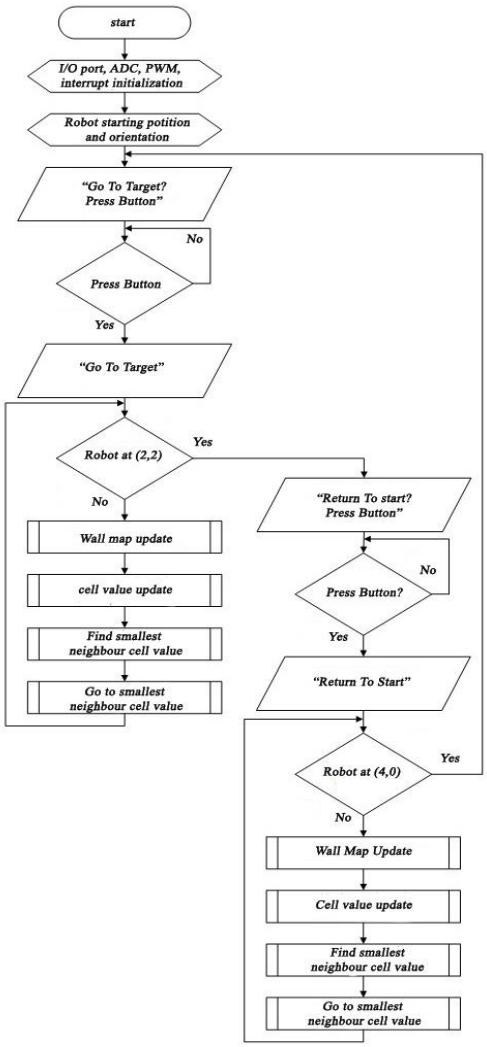

Figure 5. Flowchart of the main program.

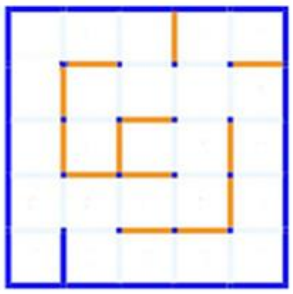

Figure 6. The layout of the maze.

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 Figure 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.

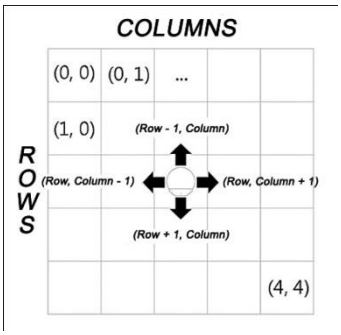

Figure 8. Robot's Array Movement

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 Figure 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 Figure 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 1 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 1.

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.

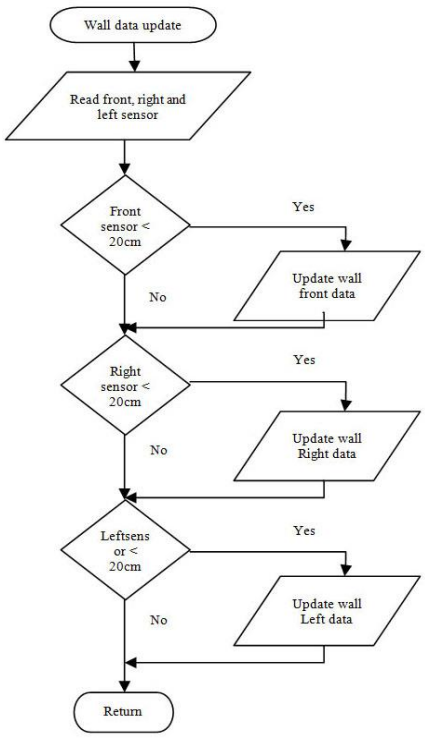

Figure 9. Wall location update flowchart.

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 Figure 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.

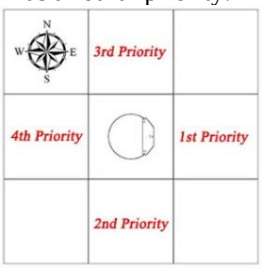

Figure 10. Priority of Neighbour cell

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 Figure 11.

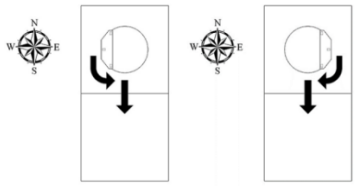

Figure 11. Moving to the smallest neighbor cell.

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. Figure 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 2 and 3. 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 5.

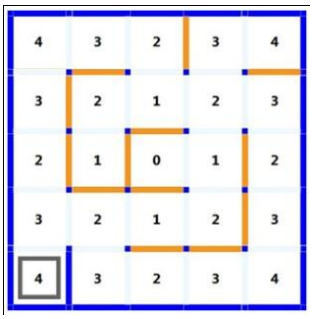

Figure 12. Simulation search route to cell (2,2), Turn = 0

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 figures 13 and 14. The results of this second experiment can be seen in tables 6 to 9.

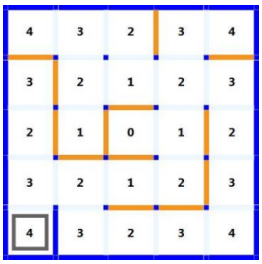

Figure 13. Simulation search path to cell (2,2) for the second experiment

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 figures 15 and 16. The results of this second experiment can be seen in tables 10 to 13.

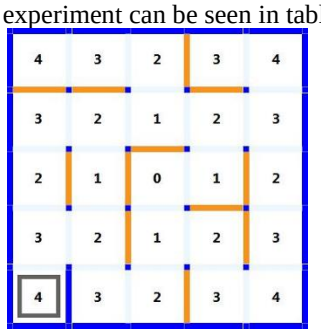

Figure 15. Simulation search path to cell (2,2) for the third experiment

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] Elshamarka, I. & Saman, A.B.S, "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] Tjiharjadi, S. & Setiawan, S., "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] Tjiharjadi, S., "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] Babula, M., "Simulated Maze Solving Algorithms through Unknown Mazes," in *XVIIIth Concurrency, Specification and Programming (CS&P) Workshop*, Krakow-Przegorzaly, 2009.

[15] Tjiharjadi, S., Wijaya, M. C., and Wijaya, E., "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.

8. Bukti Konfirmasi Artikel Published Online (19 Mei 2020)

ICAME 2019-online in IJMERR-E009

Inbox ×

icameconf@zhconf.ac.cn
to me

Tue, May 19, 2020, 7:47 PM

Dear Semuil Tjiharjadi,
Greetings from ICAME 2019.
Good news!
Your paper-"Performance Comparison Robot Path Finding uses Flood Fill - Wall Follower Algorithm and Flood Fill - Pledge Algorithm" has been online in IJMERR.
http://www.ijmerr.com/list-178-1.html
If you have any questions, please contact me.
Have a good day.

Regards

Ms. Chris Zhang
Conference Secretary of ICAME2020

Website: http://www.icame.org/
Email: icameconf@zhconf.ac.cn
Working Hours: 9:30-18:00(Monday-Friday)