Optimization Maze Robot Using A* and Flood Fill Algorithm

Semuil Tjiharjadi, Marvin Chandra Wijaya, and Erwin Setiawan
Maranatha Christian University, Bandung, Indonesia
Email: semuiltj@gmail.com

***Abstract*— As the development of Robotics technology is expanding rapidly, the developments in the world of artificial intelligence is also growing very fast. The challenge is how to make a robot to become cleverer in deciding actions based on the circumstances that exist. It has become a distinct branch of science that offers many things that need to be investigated further. This research will be developed an intelligent robot that is able to solve the shortest possible distance to reach the destination in a maze arena. The robot will be "smart" in finding the nearest road to re-discover the goal. It is applying artificial intelligence method uses 2 algorithms to be compared and optimized at the same time to be together, so it is hoped to produce optimal results. Obviously, the two algorithms used are algorithms that will be the starting point for the next development of further research in artificial intelligence research of smart robot. The algorithm is applied to these smart robot is A * Algorithm and Flood Fill Algorithm.**

***Index Terms*— artificial intelligence, maze robot, A* Algorithm, Flood Fill Algorithm**

I. Introduction

Along with the development of technology, the function of the robots is also growing. The robots are now programmed to be more intelligent and capable decided to take certain measures in accordance with existing conditions. Even through the applied artificial intelligence, robots can take a new decision based on consideration of the growing conditions. That's why the science of artificial intelligence is also growing and complex.

There are several general categories in artificial intelligence, namely Breadth First Search is developing a search based on thorough testing of all the possibilities. Depth First Search is trying to search in depth the possibility of an option available, before trying other options. Or a combination of both, known as the Best First Search. The third major category is spawned a wide range of algorithms that develop and increase the speed of the robot in finding the solution of a problem.

The use of artificial intelligence methods on a problem finding the location of the maze, currently growing. Various algorithm was developed to solve the problem of robot searches the existing maze. This is the forerunner to the manufacture of automated robots capable of searching the nearest road by a particular location.

Autonomous navigation is an important feature of automated robotics. It allows the robot to independently move from a place to target location without a tele-operator. The robot is using a structured technique and controlled implementation of autonomous navigation which is sometimes preferable in studying specific aspect of the problem [1]. This paper discusses two methods of a small size mobile robot designed to solve a maze based on A* and flood-fill algorithm [2].

This maze robot tries to solve a maze in the least time possible and using the most efficient way. Robot must navigate from a corner of a maze to the center as quickly as possible [3]. It knows where the starting location is and where the target location is, but it does not have any information about the obstacles between the two. The maze is normally composed of 256 square cells, where the size each cell is about 18 cm × 18cm. The cells are arranged to form a 16 row × 16 column maze. The starting location of the maze is on one of the cells at its corners, and the target location is formed by four cells at the center of the maze. Only one cell is opened for entrance. The requirements of maze walls and support platform are provided in the IEEE standard.

II. Literature Review

A. A* Algorithm

A* combines feature of uniform-cost search and heuristic search. It is BFS in which cost associated with each node is calculated using admissible heuristic. For graph traversal, it follows path with lowest known heuristic cost. The time complexity of this algorithm depends on heuristic used. Since it is Breadth First Search drawback of A* is large memory requirement because entire open-list is to be saved.

B. Flood Fill Algorithm

Robot maze problems are an important field of robotics and it is based on decision making algorithm [4]. It requires complete analysis of workspace or maze and proper planning [5]. Flood fill algorithm and modified flood fill are used widely for robot maze problem [6]. Flood fill algorithm assigns the value to each node which is represents the distance of that node from center. The

Manuscript received December 28, 2016; revised April 28, 2017.

doi: 10.18178/ijmerr.6.5.366-372

flood fill algorithm floods the maze when mouse reaches new cell or node. Thus it requires high cost updates. These flooding are avoided in modified flood fill.

Flood fill algorithm is an algorithm that determines the areas that are connected to a node in a multidimensional array. Flood fill algorithm is widely used in the bitmap image editor program for coloring a limited area with a specific color (boundary fill). Flood fill algorithm can be adapted to solve the problems maze solving.

Flood fill algorithm itself is analogous flooded with water maze. Water shedding process be centralized in only one cell is a cell of interest. Water continues to flow to flood the whole maze. Path traversed by the first water drops until it reaches the start location is the shortest path to reach that goal

How it works flood fill algorithm is to start giving value to each cell in the maze. The process of scoring was done by observing the position of the existing walls of the maze. The first water-filled cells are the cells of interest and these cells are given a value of 0. The water then flows into the surrounding area which is not blocked by the wall. The next cell that has been filled with water will be assigned a value of 1, then this value will continue to grow to the next cell to the entire cell occupied by water maze. The robot cannot move diagonally and the robots have learned some of the positions of the existing wall.

The values of these cells represents the distance of each cell to the destination cell. If the robot is in a cell that is worth 2, the robot is located as far as 2 cells from the cells of interest. Assume cells that are at the bottom left of the initial cell, then searched the cell, which has a smaller value than the value of the cell that is being occupied. The path is the shortest path is formed which can be reached from the initial cell leading to the destination cell

With the flood fill algorithm, each time the robot reaches a new cell, the robot needs to update the mapping of the walls, refill each cell with the new values, determine neighboring cells which have the smallest value, and continue moving towards neighboring cells which have the smallest value.

III. Hardware Design

Mobile robot base construction was made using miniQ 2WD robot chassis as shown in Fig. 1. It has a robot chassis with a diameter of 122mm. It has 2 wheels with a diameter of 42mm, 1 piece ball caster and 2 DC motors which have been furnished by the gearbox as well as two pieces of the DC motor bracket to pair on the chassis. The robot also had 2 pieces' rotary encoder. They attached to the DC motor to calculate the rotation of the wheel as shown in Fig. 1. [7]

The hardware system of this maze mobile robot can be seen in the block diagram at Fig. 2 . Fig. 3 shows the main program. This maze mobile robot used three infrared sensors to detect maze wall at right, left and front position. Driver L293D controled the direction of rotation and also speed of a DC motor [8]. Rotary encoder calculated the rotation of the right and left wheels. Push button was used to instruct the robot to start, then the system output would drive two DC motors that served as actuators to move both wheels to move forward, turn, and rotates reverse [9]. ATmega324 microcontroller process the input signal, process the algorithms, and generates output signals to control the robot [10]. Fig. 3 is shown that LCD will display the information about all actions that had been taken by the robot.

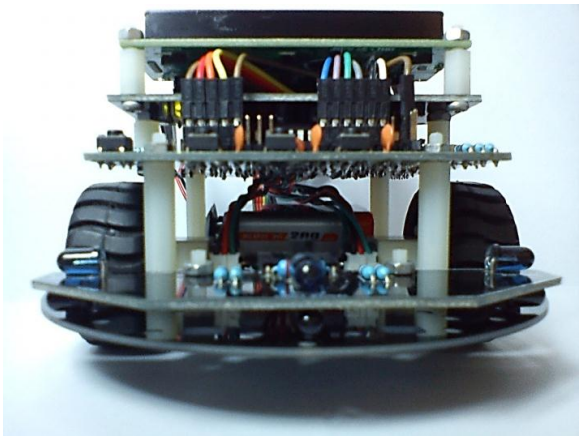

Figure 1. Mobile Robot from side view.

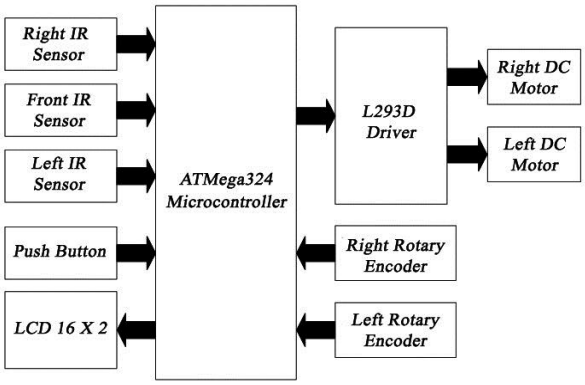

Figure 2. Block Diagram of Mobile Robot.

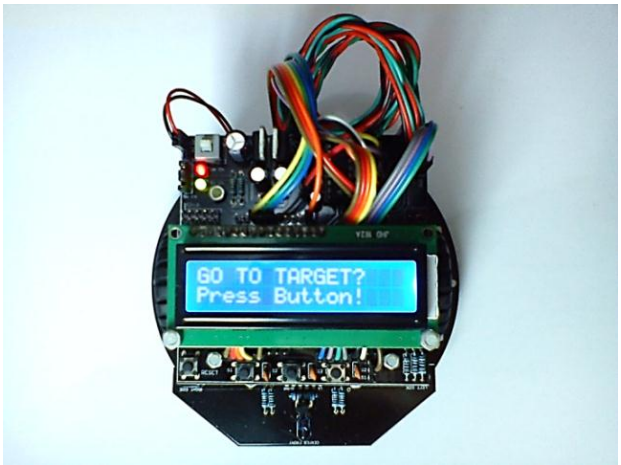

Figure 3. Mobile Robot from above view.

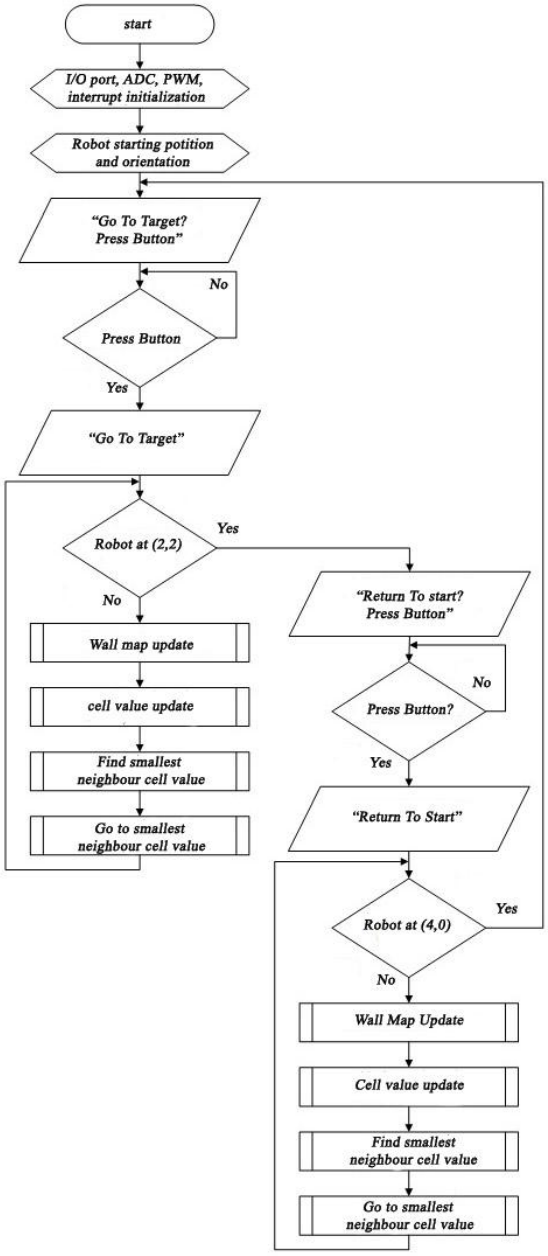

Figure 4. Flowchart of the main program.

The maze designed for the robot to solve is of the size of 5 ×5 cells.

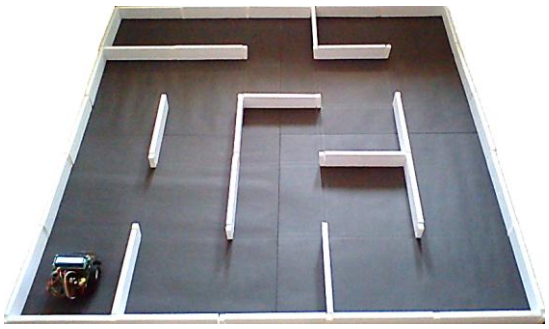

Figure 5. The maze.

IV. TESTING

The test of artificial intelligence carried out on a maze which has a size of 5x5 cells as shown in Fig. 6. The robot will conduct a search to find the shortest path from the starting cell (line 4, column 0) to the destination cell (line 2, column -2) and then back again to the initial cell. The initial orientation of the robot facing the North.

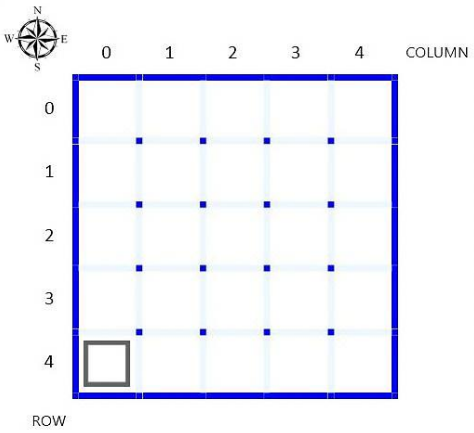

Figure 6. 5x5 cell Maze

A. Flood Fill Testing

Robot will perform a search of the initial cell lines (4, 0) to the destination cell (2, 2). The results of the search process of cell lines (4, 0) to the cell (2, 2) are shown in Fig. 7-15.

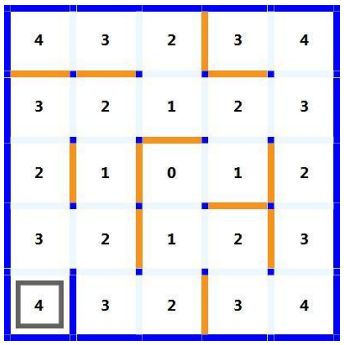

Figure 7. Simulation search path to cell (2,2)

Seen in Fig. 7, the robot is in the cell (4, 0). The robot finds a new wall on the east side. Then the robot will update the value of a cell by placing cells (2, 2) as the destination cell, so the search is done on cell lines (4, 0) to the cell (2, 2). Then the robot will move to a neighboring cell that has the smallest value that the cells (3, 0).

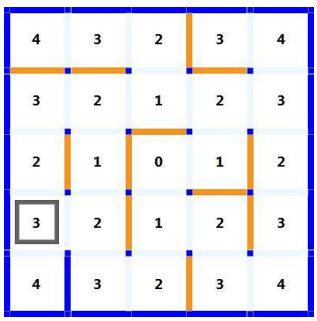

Figure 8. Simulation search path to cell (2,2)

In Fig. 8, the robot is now in the cells (3, 0). In these cells, the robot does not find a new wall. So the update cell values do not cause any changes in the value of the cell. Then the robot doing the movement to neighboring cells that have the smallest value, is the value of the selected neighboring cells are cells (2, 0).

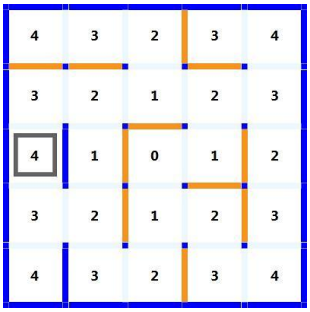

Figure 9. Simulation search path to cell (2,2)

In Fig. 9, the robot is now in a cell (2, 0). In this position, the robot finds a new wall on the east side. After the robot to update the value of the cell, then the cell (2, 0) will change the value. So that the robot will move to a neighboring cell that has the smallest value, is cell (1, 0).

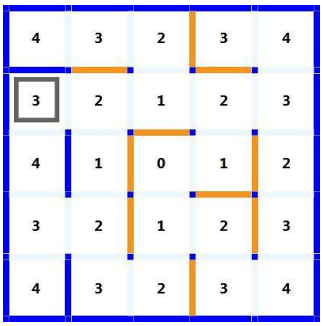

Figure 10. Simulation search path to cell (2,2)

In Fig. 10, the robot is now in a cell (1, 0). In this cell position, the robot finds a new wall on the north side. Update the value of the cells did not cause any changes in the value of the cell. So that the robot will move to a neighboring cell that has the smallest value that the cell (1, 1).

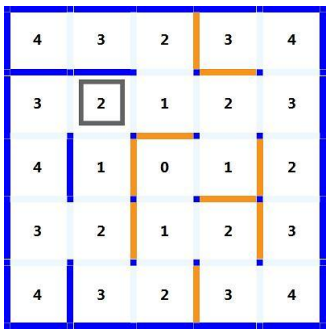

Figure 11. Simulation search path to cell (2,2)

In Fig. 11, the robot is now in a cell (1, 1). In this position, the robot finds a new wall on the north side. So the update cell values do not cause any changes in the value of the cell. Then the robot will move to a neighboring cell that has the smallest value. Neighboring cells are selected that the cell (1, 2).

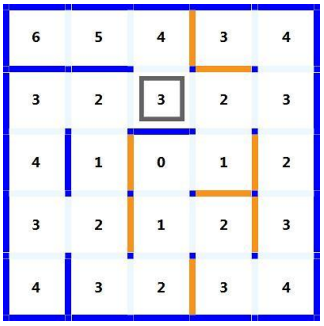

Figure 12. Simulation search path to cell (2,2)

In Fig. 12, the robot is now in a cell (1, 2). In this position, the robot finds a new wall on the south side. After the robot updating the value of the cell, then the cell (0, 0), (0, 1), (0, 2), and (1, 2) to change the value. Then the robot will move to a neighboring cell that has the smallest value. Neighboring cells are selected that the cell (1, 3).

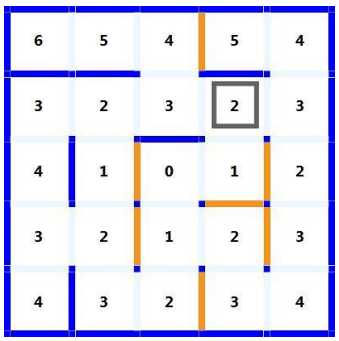

Figure 13. Simulation search path to cell (2,2)

In Fig. 13, the robot is in the position of the cell (1, 3). In this position, the robot finds a new wall on the north side. After the robot to update the value of the cell, then the cell (0, 3) will change the value. Then the robot will move to a neighboring cell that has the smallest value that the cells (2, 3).

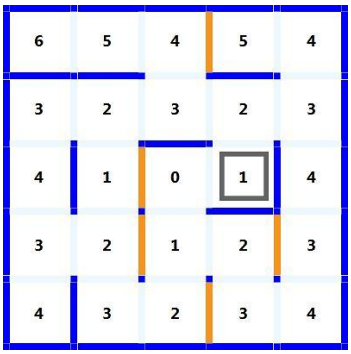

Figure 14. Simulation search path to cell (2,2)

In Fig. 14, the robot has been in the position of the cell (2, 3). In this cell position, the robot finds a new wall on the side of the South and East. After the robot to update the value of the cell, then the cell (2, 4) will change the

value. Furthermore, the robot will move to a neighboring cell that has the smallest value that the cells (2, 2). In Fig. 4.11, the robot has arrived at the location of the destination cell (2, 2).

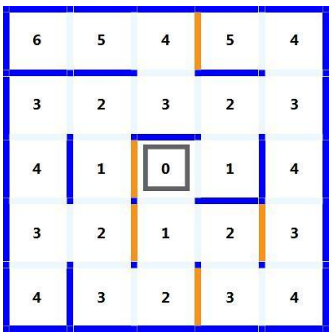

Figure 15. Simulation search path to cell (2,2)

After the robot to position the cells (2, 2), the robot will search the path back to the cell (4, 0). The results of the trip artificial intelligence program at the time of the search of the cell lines (2, 2) return to the cell (4, 0) is shown in next Figures.

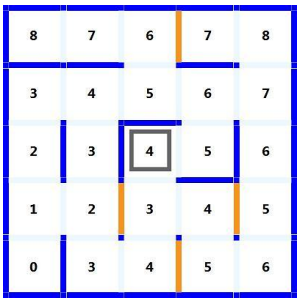

Figure 16. Simulation search path to cell (2,2)

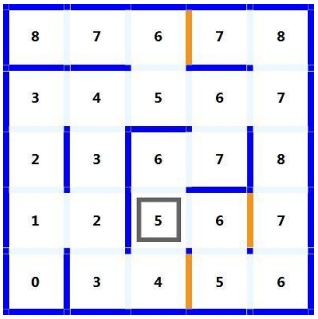

Figure 17. Simulation search path to cell (2,2)

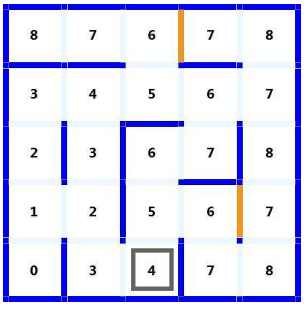

Figure 18. Simulation search path to cell (2,2)

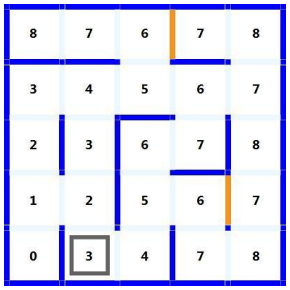

Figure 19. Simulation search path to cell (2,2)

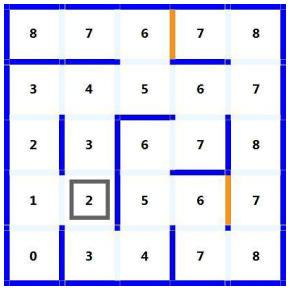

Figure 20. Simulation search path to cell (2,2)

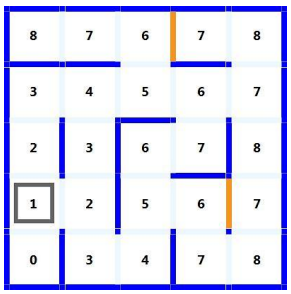

Figure 21. Simulation search path to cell (2,2)

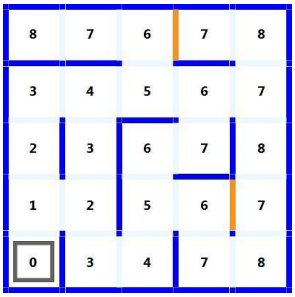

Figure 22. Simulation search path to cell (2,2)

After the robot movement by choosing neighboring cells which have the smallest value, then the robot is now in the destination cell (2, 2) and the path is a shortest path from the starting cell to get to the destination cell.

In Table 1, the first departing requires the movement of as much as 8 cells to reach the destination cell. While in the second process in table 2, the robot requires just as much as 6 cell movement. This happens because the robot has mapped the location of the position of the wall, so the robots can map where a shorter path to get to the destination cell. So that the path through the process of setting off the second is the shortest path.

TABLE I. FIRST AND SECOND ROUTES OF ROBOT EXPERIMENT

Testing	Routes	Number of 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) → (3,1) → (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

B. *A* Algorithm Testing*

On the use of robot movements using the A * algorithm obtained the situation that the robot must know the position of the wall in advance so that the new can be calculated using the A * algorithm. So that when mapped and then use the maze on a trial run, Robot obtained using the path as follows:

TABLE II. A * ALGORITHM ON THE CONDITION OF THE KNOWN WALL LOCATION

Routes	Number of steps
(4,0) → (3,0) → (3,1) → (4,1) → (4,2) → (3,2) → (2,2)	6

Meanwhile, when the robot tested without knowing the position of the wall first, obtains the same results with experiments in Table 3.

TABLE III: A * ALGORITHM OF THE UNKNOWN WALL LOCATION

Routes	Number of steps
(4,0) → (3,0) → (2,0) → (1,0) → (1,1) → (1,2) → (1,3) → (2,3) → (2,2)	8

C. *Testing Results on A* and the Flood Fill algorithm comparison program*

On the use of robot movements in table 4, that try to optimize the use of the A * algorithm and the Flood Fill, obtained similar results with each experiment.

TABLE IV: THE MOVEMENT OF THE ROBOT IN THE MAZE USING THE A * ALGORITHM TO OPTIMIZE EXPERIMENT WITH THE FLOOD FILL

Testing	Routes	Number of 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) → (3,1) → (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

D. *Additional Testing with Different Layout*

Other experiments with different pattern of board, to make a better comparison for both methods. Fig. 23 describes the second layout of board to test the both methods.

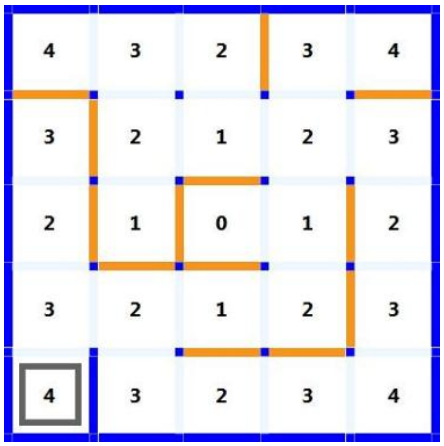

Figure 23. Simulation search path to cell (2,2)

Robot will perform a search of the initial cell lines (4.0) to the destination cell (2, 2). Flood fill algorithm simulation results when a search of the cell lines (4, 0) to the cell (2, 2) are shown in Table V.

TABLE V. THE MOVEMENT OF THE ROBOT IN THE MAZE USING FLOOD FILL ALGORITHM

Testing	Routes	Number of 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

Robot will perform other experiments using A* algorithm in table VI for known wll location and table VII for unknown wall location.

TABLE VI. A * ALGORITHM ON THE CONDITION OF THE KNOWN WALL LOCATION

Routes	Number of steps
(4,0) →(3,0) → (3,1) → (3,2) → (3,3) → (2,3) → (2,2)	6

Meanwhile, when the robot tested without knowing the position of the wall first, obtains the same results with experiments in Table 3.

TABLE VII. A * ALGORITHM OF THE UNKNOWN WALL LOCATION

Routes	Number of steps
(4,0) →(3,0) → (2,0) → (1,0) → (2,0) → (3,0) → (3,1) → (3,2) → (3,3) → (2,3) → (2,2)	8

Table Vto VII describe the same results using Flood Fill Algorithm and A* Algorithm.

V. CONCLUSION

From the results of experiments that have been conducted in this study, some conclusions as follows:

1. The robot can find the shortest travel path after successfully mapping the maze arena.
2. Use of arena size of 5 x 5 have not been able to compare differences in the A * algorithm and Flood Fill. They need wider arena size to get more detail comparison for both of them.
3. The test optimization program, still lead to improvements in the search results, it can be caused both methods own the shortest path.

ACKNOWLEDGEMENT

This work was supported by Maranatha Christian University, Indonesia

REFERENCES

[1] Elshamarka, Ibrahim and Abu Bakar Sayuti Saman. "Design and Implementation of a Robot for Maze-Solving using Flood-Fill Algorithm", *Universiti Teknologi Petronas*, 2012.

[2] Elshamarka, I. and A.B.S. Saman, "Design and Implementation of a Robot for Maze-Solving Using Flood-Fill Algorithm", *in International Journal of Computer Applications*, Volume 56-No.5, pp.8-13, October 2012.

[3] Tjiharjadi, Semuil and Erwin Setiawan, "Design and Implementation of Path Finding Robot Using Flood Fill Algorithm", *International Journal of Mechanical Engineering and Robotics Research*, Volume 5, No. 3, July 2016, pp 180-185.

[4] Ansari, A., M.A. Sayyed, K. Ratlamwala and P. Shaikh, "An Optimized Hybrid Approach For Path Finding", *in International Journal in Foundations of Computer Science & Technology (IJFCST)*, Vol. 5 No. 2, pp. 47-58, March 2015.

[5] Sharma, K. and C. Munshi, "A Comprehensive and Comparative Study of Maze-Solving Techniques by Implementing Graph Theory", *in IOSR Journal of Computer Engineering*, Vol. 17, Issue 1, Ver. IV, pp. 24-29, 2015.

[6] Sreekanth, R.K., "Artificial Intelligence Algorithms", *IOSR Journal of Computer Engineering (IOSRJCE)*, volume 6, issue 3 September-October, 2012.

[7] Cook, David, "Intermediate Robot Building", *New York: Apress*, 2010.

[8] Scherz, Paul, "Practical Electronics for Inventors", *New York: McGraw-Hill*, 2000.

[9] Braunl, Thomas, "Embedded Robotics", *Berlin: Springer*, 2006.

[10] Mazidi, Muhammad Ali, Sarmad Niami, and Sepehr Niami, "The AVR Microcontroller and Embedded System", *New Jersey: Prentice Hall*, 2011.

Semuil Tjiharjadi is currently serves as vice rector of capital human management, assets and development. He is also Assistant Professor 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, Senate of Engineering Faculty, Member: APTIKOM, Member: MSDN Connection, Member: AAJI.