Received: 20 March 2016 | Revised: 29 April 2016 | Accepted: 10 May 2016
DOI: 10.37917/ijeee.12.1.8 | Vol. 12| Issue 1 | June 2016
Open Access
Iraqi Journal for Electrical and Electronic Engineering
Original Article
IJEEE University of Basrh College of Engineering

Autonomous Navigation of Mobile Robot Based on Flood Fill Algorithm

Ayad Mohammed Jabbar
Computer Science Department
Shatt Al-Arab University College
Basra, Iraq

Abstract: *The autonomous navigation of robots is an important area of research. It can intelligently navigate itself from source to target within an environment without human interaction. Recently, algorithms and techniques have been made and developed to improve the performance of robots. It's more effective and has high precision tasks than before. This work proposed to solve a maze using a Flood fill algorithm based on real time camera monitoring the movement on its environment. Live video streaming sends an obtained data to be processed by the server. The server sends back the information to the robot via wireless radio. The robot works as a client device moves from point to point depends on server information. Using camera in this work allows voiding great time that needs it to indicate the route by the robot.*

Index Terms—Mobile robot, Wi-Fi, OpenCV, EV3, LEGO Mindstorm.

I. Introduction

An intelligent robot is a smart machine with high ability in leading itself based on the information that is available in its environment. Autonomous navigation system is one of the most important research fields nowadays. It allows moving from source to destination without any human operator interaction. Finding the target in optimal time is the most challenging issue faced by researchers in this area, leading to the development of algorithms and techniques, to improve performance [1]. This work proposes to solve a maze by using a Flood fill algorithm with real time camera monitoring movement in the environment. Live video streaming sends data to the server, which processes the obtained images while simultaneously sending information back to the robot via wireless (Wi-Fi) radio. The robot works as a client and executes commands via Wi-Fi, while the camera detects the location for each movement. The goal of using a camera is to increase the robot performance by finding the shortest route in the shortest time. Here the technique employs the target to detect the starting point, which differs from the usual way. The technique begins at the final point searching for all possible points giving them a special number depending on the distance of the nearest neighbor point. Live video streaming analyzed by a server using Java language with an open source computer vision (OpenCV) library for image processing. The robot acts as a client receives and executes tasks based on the server. The maze that must be solved contains 16 cells, with a matrix size of 4×4 and a cell size of approximately 30 cm. The starting point is located in one corner, and the final target is at the center of the maze. Results were tested with related works using 16x16 solve maze. The evaluated results show that the algorithm is better and high efficiency and generates the result in perfect time.

II. Related Works

Elshamarka and Saman used the Flood fill algorithm, which includes the following steps: update the walls, flood the maze, determine the turn, and move to the next cell. Although the robot can easily obtain the shortest route using the Flood fill algorithm, it still takes too much time to complete the process [2].

Dang and Song proposed a technique based on the Flood fill algorithm to reduce time costs by skipping some an unnecessary steps not in certain cases. Their work realized that a number of channels could exist inside the maze, which means when the robot enters the maze, it can only move to the cell in front. All four steps can be performed using this technique [3].

III. Basic Theory of Flood fill Algorithm

Choosing an algorithm for the maze robot is critical in solving the maze. The researcher selected Flood fill algorithm to solve the maze because of its efficient balance. Flood fill algorithm is one of the best mazes solving algorithms. The algorithm is used to discover walls and obtain the best available route. Each cell in the maze provides a value that indicates its distance from the target.

The algorithm begins to address the distance of each cell from its neighbors while no wall exists to separate the cells. The mechanism starts from the final point by giving it an initial value and then moves forward to other neighboring points by giving them a higher value. Sometimes the algorithm identifies more than one route with the same weight. However, it should determine the shortest route and the most direct route to the destination. Fig. 1, shows the flowchart of this algorithm which mainly includes four parts [4].

- Update the walls.
  Before the robot decides where it wants to go, it has to check for adjacent walls in any of the three directions: right, left, and front.

- Flood the maze.
  After updating the wall information for the current cell, the robot starts to flood the matrix to obtain the shortest route to the goal.

- Determine the turn.
  After the robot decides which direction it will go next, it returns the number of degrees of turn to go to the target cell.

- Move to the next cell.
  Move the robot from its location to the next, depending on the above information.

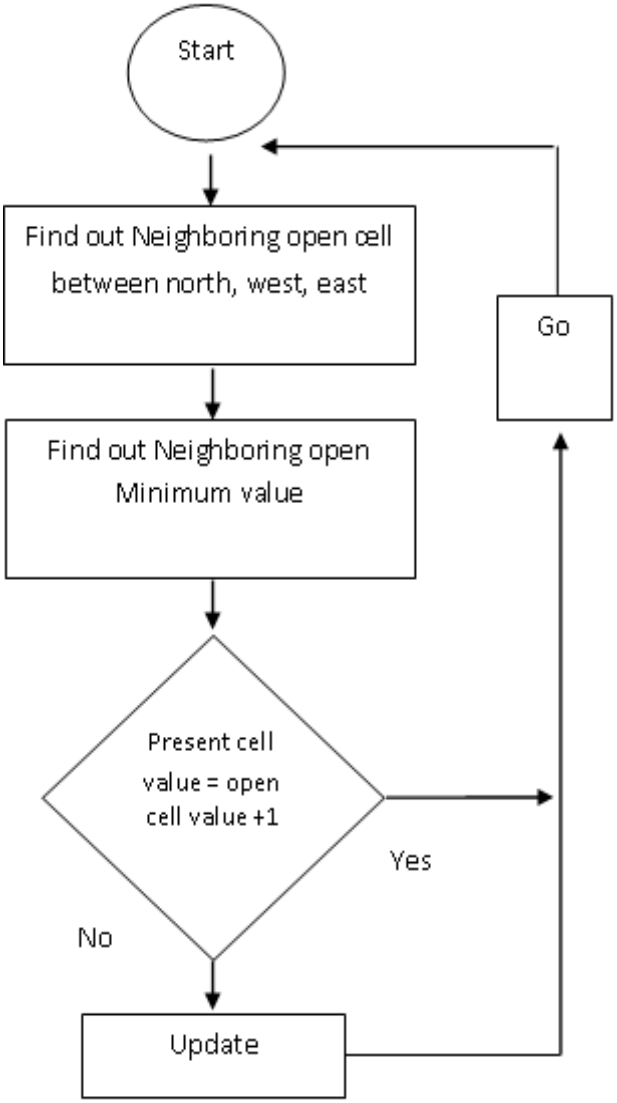

Fig. 1 Flowchart for Flood fill algorithm

As shown in the next scenario Fig. 2, the idea of the algorithm starts at the destination (the goal which is the center of the maze) by giving an initial value. The algorithm fills the maze with higher values, which represent the distance between each cell and the goal. The mechanism keeps tracking until reach the starting cell. The algorithm stops at the start cell and downhill to the goal by following the values. In the real scenario, the robot works differently, it starts from a starting cell follows the values downhill to the goal. When it's time comes to move, the robot uses the four steps that mentioned before to perform the task based on the algorithm.

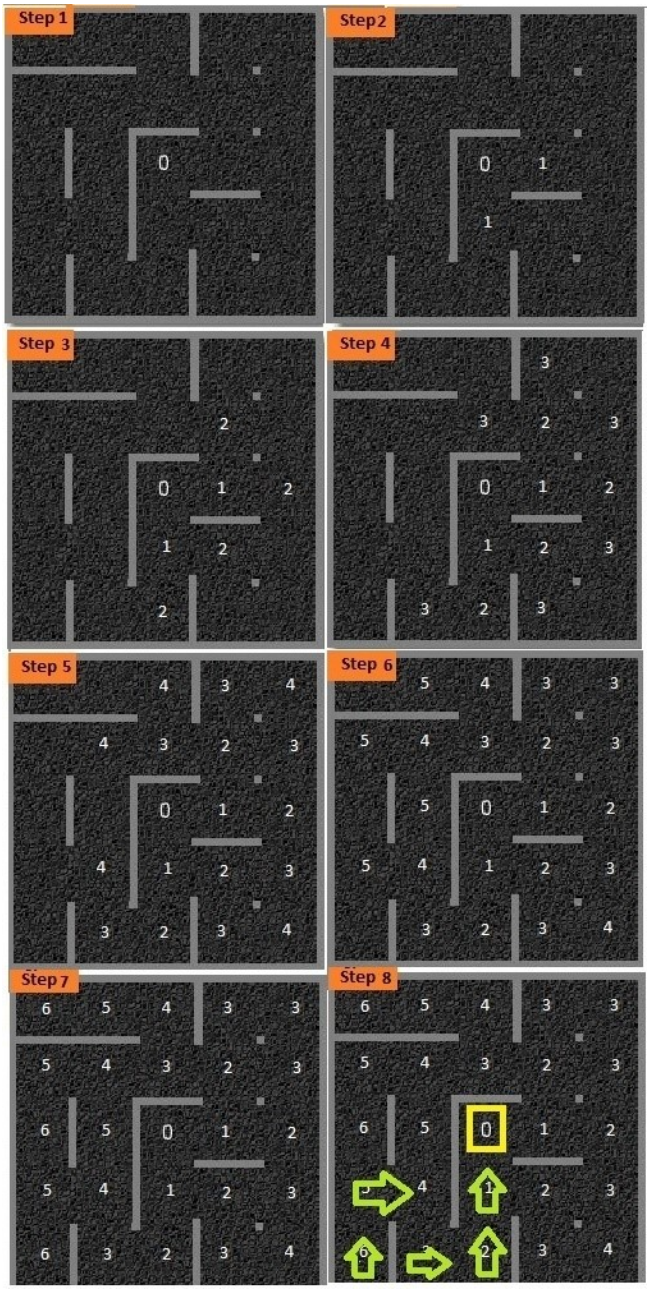

Fig. 2 Sequence of filling the maze

The robot identifies the adjacent cells, which are not separated by walls, then select the lowest value. As show in the Fig. 3, the robot establishes from the starting point number 4, ignores west wall because the robot lockup for the lowest values that are not separated by walls in the north, east and south of cell. In this scenario the robot moves forward to the north of the cell to number 3. In this step east and north cells have equal value, while the south has a value 4. However the priority in this case moves forward to the north cell because of turning would take too much time.

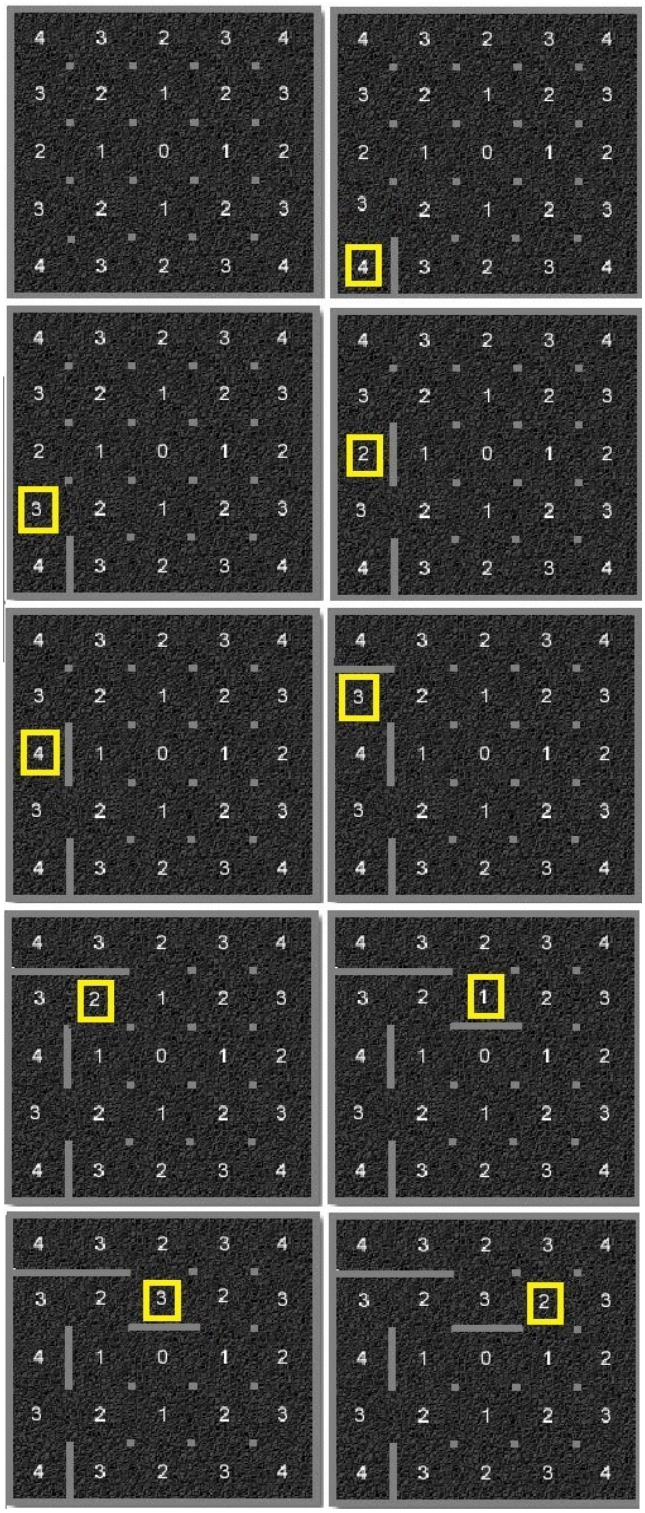

Fig.3 Updating maze walls

The robot reached step four at the cell number 2. In this step the robot identifies a wall on the east cell. Thus the robot updates the cell by a new value as a result of updating the wall. The robot keeps tracking fills the maze with new values until it reaches the goal. The same case in the night step where robot found a new wall and distance value had to change.

IV. Problem And Approach

Although the robot using the Flood fill algorithm can obtain the shortest route to its destination, problems are still encountered when using the algorithm in the maze solving process as follows:

• First, for each cell the robot reaches, it must execute four steps: update the walls, flood the maze, determine the turn, and move to the next cell. The search speed of the robot is one of the problems because time is a key factor in this competition [5].

• Second, to rapidly locate the destination, the robot must determine the direction it should take when it reaches across cell that has two or more directions having the same value, so the decision costs gets much time in this scenario.

Consequently, these problems take too much time. However, most of the steps can be avoided by using a camera here, a computer processes the data, the processed data are sent to the robot thus reduces the processing time that has to be concerned by a robot.

V. Architecture

The architecture of this study is divided into three parts camera, server and robot. The study tried to avoid most problems that are mentioned earlier in related works. The camera supports the server by providing live streaming on the maze environment. The server uses an OpenCV library for image processing to find all routes of environment, and then detects the shortest final distance based Euclidean distance.

The server analyzes images that come from a camera then, converted directly to binary matrix. And finally, the obtained matrix is evaluated by Flood fill algorithm. Before sending extracted information by Wi-Fi, the server calculates the required steps and the trends to be executed by the robot. For example the instruction (2, 3+1, E) means the robot moving one step from its location to the new point which is it column number four (East). The robot will execute the sequence of instructions to reach its destination. The overall functions of this process are summarized in Fig. 4.

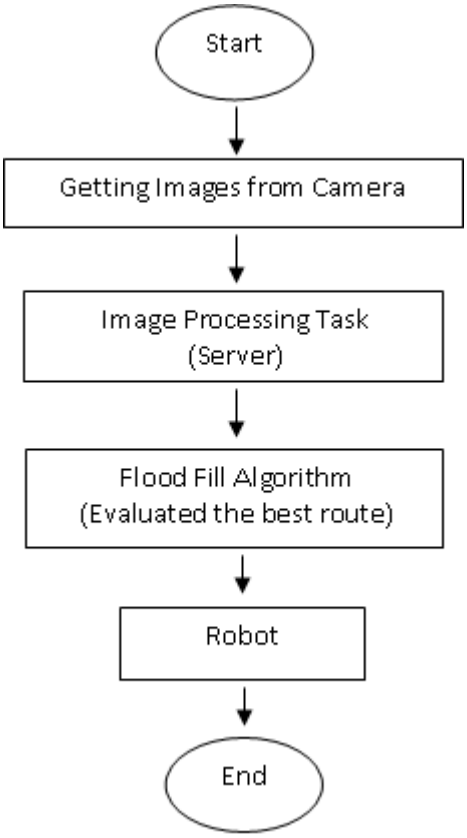

Fig. 4 Architecture flowchart

VI. Development

The prototype developed using the Java language, with Windows operating system as its platform. The code is an open source available for researchers who are interest in this field to modify and update the source code, depending on their needs. The prototype uses OpenCV library which is a real time computer vision for image processing. The OpenCV library is developed by Intel research center and supported by different operating system platform even computers or mobiles. The library is free to be used for researchers who are interested in this field. The camera transfers the image into the server. Image processing task converts the obtained RGB image to another formula, which, is a grayscale image. The grayscale image is a binary image. In the binary image, each pixel contains one of only two values: either 1 or 0. A binary image is stored as a logical array represents a black color for logic 0 and white color for logic 1 [6]. Finally, objects begin to appear as shown below in Fig. 5.

In this figure the maze was detected by subtracting other colors. Image contains white color for 1 binary and the black background as 0 binary. However, by detecting both colors, it

becomes easy for the robot to flow the black color while the white color represents the border of the maze.

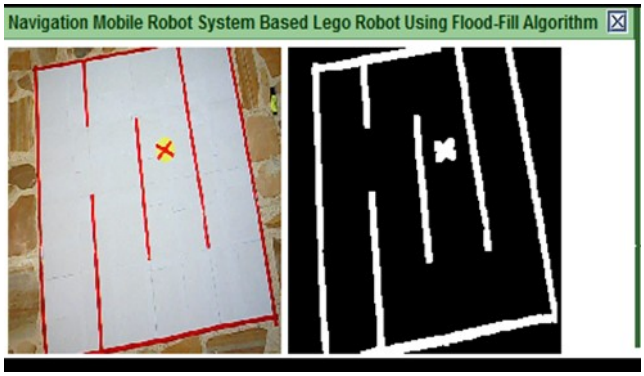

Fig. 5 Objects tracking

VII. Robot Technical Features

EV3 Lego Mindstorm is the third generation robotics kit that is released in 2013[7]. It makes researcher able to build a powerful program and find their solutions based on real life robotics technology. The EV3 is an intelligent kit, which is a small computer that makes it possible to control motors and collect data feedback from a sensor. It contains 32-bit microprocessor, a large matrix display, 4 input and 3 output ports, Wi-Fi, USB modem and Bluetooth to communicate with other communication points. In this paper, we upgrade the operating system with a new (leJOS) which is an open Source Java based operating system as a replacement for the original Lego operating system [8]. It currently includes a Java virtual machine, which lets researchers to program Lego robots in Java language rather than using the traditional Lego environment, which contains some limitations. Fig. 6 shows the Lego robots.

Fig. 6 LEGO Mindstorm EV3

VIII. Analysis And Comparisons

In order to evaluate the results, a comparison is done to compare the researchers' result in term of time with related work for 16x16 solve a maze as represented in the Fig. 8. The results of three tests are shown in Table I. In this test the researcher's symbolize T1 as a result of Elshamarka and Saman their work, and T2 for Dang and Song work while T3 the researches' result.

TABLE I. Comparison Experiments Data

Time Table	16×16maze
T1	266
T2	221
T3	173

It should be noted that the three experiments are shown in Table I, have been conducted and evaluated all on the same robot. The results showed that enhancing algorithm by a camera with image processing is better in time searching and much shorter than the time spent in both related works T1 and T2. In the first experiment T1, the time was taken by the robot about 266 sec for reaching the destination while the second experiment T2 did the task with less time about 221 sec. The third experiment T3 showed the minim time of the experiments was about 173 sec for reaching destination. From the results it can be noted in the experiment of 16x16 cells maze that, the work needs about quarter of time to reach the goal. Graphical comparison data results showed in the Fig. 7.

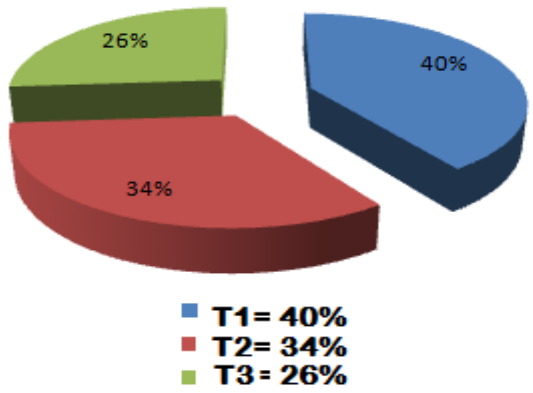

Fig. 7 Graphical comparison experiment

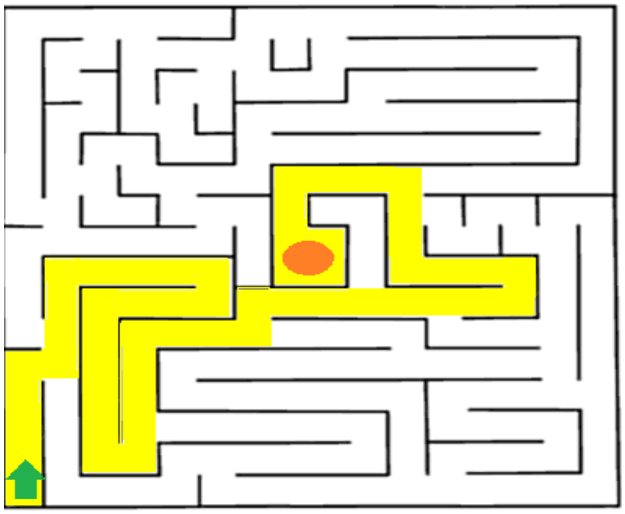

Fig. 8 Final route for 16 x 16 solve maze

IX. Conclusion And Future Work

Maze solving problem is one of the common issues discussed in the few past years. Researchers used different algorithms to solve this problem. Using a good algorithm can get a high performance led to find a best shortest route in short time. The proposed technique based on the Flood fill algorithm works better and with less searching time. Using camera in this work avoids the algorithm step by executing them in the computer using image processing technique rather than using them by a robot. The results evaluated in a maze of 16x16 cells. Results showed that the algorithm enhanced by camera, the performance improved and the result in term of time is much shorter. This work was tested using an EV3 robot based java language for both the server and the client robot. Therefore, the next stage towards making the robot connected directly to camera contains wireless modem. This step makes the robot more efficient through its ability to process and decision making by itself, instead of using the computer for this task.

References

[1] L. Wyard-Scott and Q. H. M. Meng, "A potential maze solving algorithm for a micromouse robot," *Commun. Comput. Signal Process. 1995. Proceedings. IEEE Pacific Rim Conf.*, vol. 55, pp. 614–618, 1995.

[2] I. Elshamarka and A. Bakar Sayuti Saman, "Design and Implementation of a Robot for Maze-Solving using Flood-Fill Algorithm," *Int. J. Comput. Appl.*, vol. 56, pp. 8–13, 2012.

[3] H. Dang, J. Song, and Q. Guo, "An efficient algorithm for robot maze-solving," *Proc. - 2010 2nd Int. Conf. Intell. Human-Machine Syst. Cybern.*, vol. 2, pp. 79–82, 2010.

[4] J. A. Pandian, R. Karthick, and B. Karthikeyan, "Maze Solving Robot Using Freeduino and LSRB Algorithm," *International Journal of Modern Engineering Research.*, vol. 56, pp. 92-100, 2012.

[5] B. H. Kazerouni, M. B. Moradi, and P. H. Kazerouni, "Variable Priority in Maze-Solving Algorithms for Robot Movement," *International Association For Automation And Robotics In Construction.*, pp. 147–152, 2003.

[6] M. T. Rashid, H. A. Zaki, and R. J. Mohammed,"Simulation of Autonomous Navigation Mobile Robot System," *Journal of Engineering and Development.*, vol. 18, pp. 25-38, 2014.

[7] S. R. Perez, C. Gold-Veerkamp, J. Abke, and K. Borgeest, "A new didactic method for programming in C for freshmen students using LEGO mindstorms EV3," *2015 Int. Conf. Interact. Collab. Learn.*, pp. 911–914, 2015.

[8] M. W. Lew, T. B. Horton, and M. S. Sherriff, "Using Lego Mindstorms NXT and LeJOS in an advanced software engineering course," *Softw. Eng. Educ. Conf. Proc.*, pp. 121–128, 2010.